Udacity - Machine Learning Engineer Nanodegree

Dhanaji More
06/23/2017

Capstone Project

Predict Future Sales

Table of Contents

Introduction

Accurately forecasting product sales and building a sales plan is a common problem all enterprises face. Accurate forecasting helps companies.

- avoid inventory buildup problems
- avoid unforeseen cash flow problems and
- effectively manage production, staff and financing needs.
- maximize profit.

Forecasting is an essential activity for managing a business of any size. It is a month-by-month forecast of the level of product or service sales a company expects to achieve.

Project Overview

Traditional sales forecast methods mainly use time series analysis techniques.

[G. Keller and N. Gaciu. Managerial statistics. South-Western Cengage Learning, 2012.]

[1 W.-I. Lee, B.-Y. Shih, and C.-Y. Chen. A hybrid artificial intelligence sales-forecasting system in the convenience store industry. Human Factors and Ergonomics in Manufacturing & Service Industries, 22(3):188–196, 2012.].

Classical time series techniques include the autoregressive models (AR), integrated models (I), and moving average models (MA). These models predict future sales using a linear function of the historical sales data. More recent models, such as autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA), are more general and are shown to achieve better performance.

These time series analysis models take the historical sales data as the input and are suitable for sales data with stable or seasonal sales trends. To better handle irregular sales patterns, some new methods attempt to exploit more information in sales forecast as an increasing amount of data are becoming available in E-commence.

Many of the decision support systems in today's ecommerce systems [Using online search data to forecast new product sales. Decision Support Systems, 52(3):604–611, 2012.] use online search data to forecast new product sales by using search term volume as a marketing metric. Attempts have been made to improve the forecast accuracy in promotional sales by incorporating product specific demand factors using multiple linear regression analysis. Many other models predict product sales by identifying customers purchase purpose from their browsing behavior. These methods are generally case-by-case developed for specific commercial scenarios and are limited in their applicability. They rely on specific domain knowledge to extract relevant features from the data, which is labor intensive and exhibits their inability to extract and organize the discriminative information from the data.

In this project, a Kaggle competition, we are challenged to predict sales of a product in a store during the next month in sequence. It has a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company.

We are provided with daily historical sales data. Our objective is to forecast the total amount of products sold in shops for items in the 35th month in the test set.

https://www.kaggle.com/c/competitive-data-science-predict-future-sales#description

We are provided 34 months of daily sales data from 60 stores that have sold over ~2.9 million items. The lists of shops and products changes slightly every month. We have to create a robust model that can handle such situations is part of the challenge.

Problem Statement

We here describe the problem in a formal way. Given the sales data of a product (item_id) at a certain store (shop_id) at a certain price (item_price) of certain count (item_cnt), we intend to forecast its (shop_id) total sales of its product (item_id) in next time sequence.

Our goal is to build a mapping function to predict y (item_cnt) at time t+1 with X (historical sales data) as the input at time t.

$$y_{t+1} = f(X_{t)}$$

Metrics

For classification problems popular metrics such as precision, recall are common ways to check the accuracy of the model. Since we are predicting a continuous variable (item_count) we will use metrics mostly used for regression problems such as RMSE and MAE.

RMSE (Root Mean Square Error)

It represents the same standard deviation of the differences between predicted values and observed values (called residuals). The formula we will use.

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Yhat – is the predicted value of item sales count.
Yi. - is the actual value of item sales count.
n - total number of observations.

MAE (Mean Absolute Error)

It is the average of the absolute difference between the predicted values and obsev4d value. The MAE is a linear score which means that all the individual differences are weighted equally.
MAE is calculated using the following formula:

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

MAE metric is easy to understand and to interpret because it directly takes the average of offsets whereas RMSE penalizes the higher difference (square error) more than MAE.

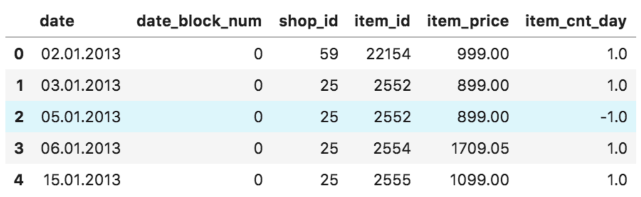
MAE and RMSE both should help us better evaluate and compare our neural network models.

Data Analysis

Data Exploration

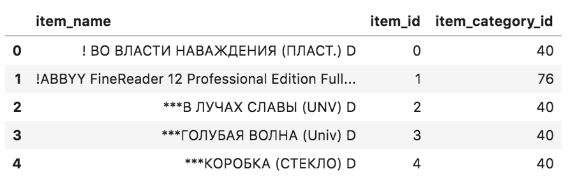
Dataset consists of four relational tables as standard csv files:

Data Filename	Description
sales_train.csv	Training set. Daily historical data from Jan 2013 to Oct 2015.
items.csv	Supplemental information about the items/products.
item_categories.csv	Supplemental information about the item categories.
shops.csv	Supplemental information about the shops.



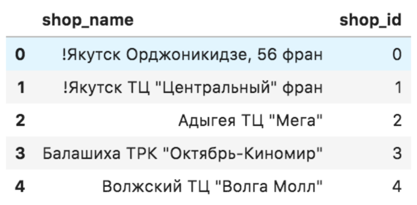
	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

sales_train



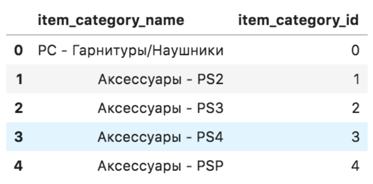
	item_name	item_id	item_category_id
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40
1	!ABBYY FineReader 12 Professional Edition Full...	1	76
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40
4	***КОРОБКА (СТЕКЛО) D	4	40

items



	shop_name	shop_id
0	!Якутск Орджоникидзе, 56 фран	0
1	!Якутск ТЦ "Центральный" фран	1
2	Адыгея ТЦ "Мега"	2
3	Балашиха ТРК "Октябрь-Киномир"	3
4	Волжский ТЦ "Волга Молл"	4

shops



	item_category_name	item_category_id
0	PC - Гарнитуры/Наушники	0
1	Аксессуары - PS2	1
2	Аксессуары - PS3	2
3	Аксессуары - PS4	3
4	Аксессуары - PSP	4

item_categories

Dataset observation and analysis:

a. Dataset stats

▪ Shops	- 60
▪ Item Categories	- 84
▪ Items	- 22,170
▪ Median item price	- 399
▪ Min item price	- (-1.0)
▪ Max item price	- 307,980
▪ Total Rows	- 2,935,849

b. Data has no missing values.

c. Price has negative values

d. item_sales_count field in sales_train has negative values indicating this maybe returns.

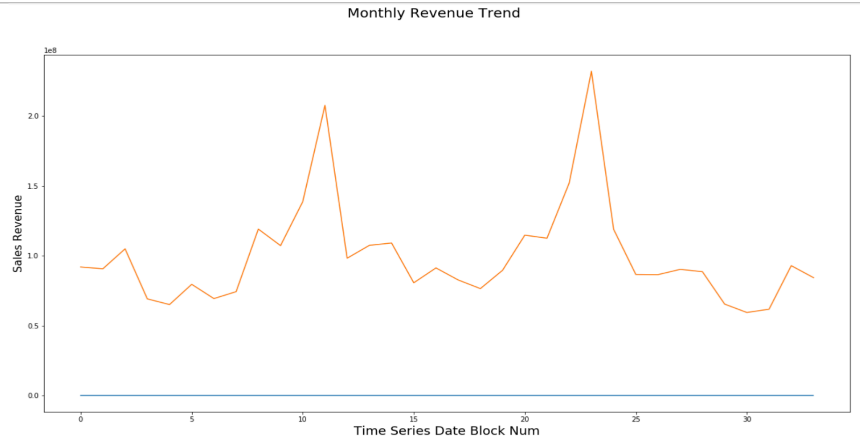
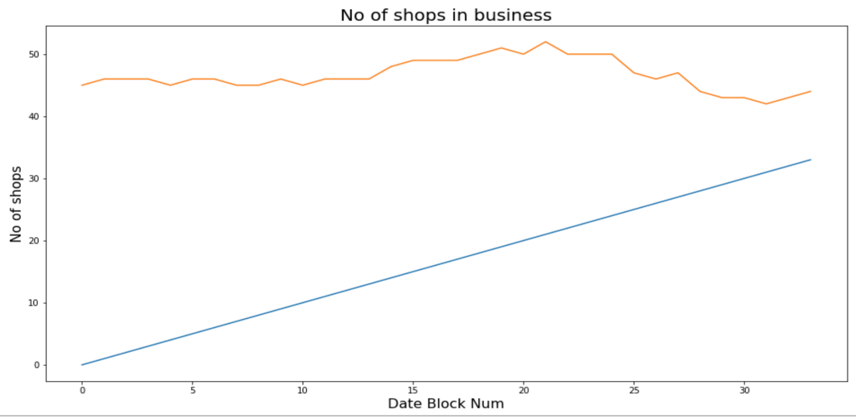
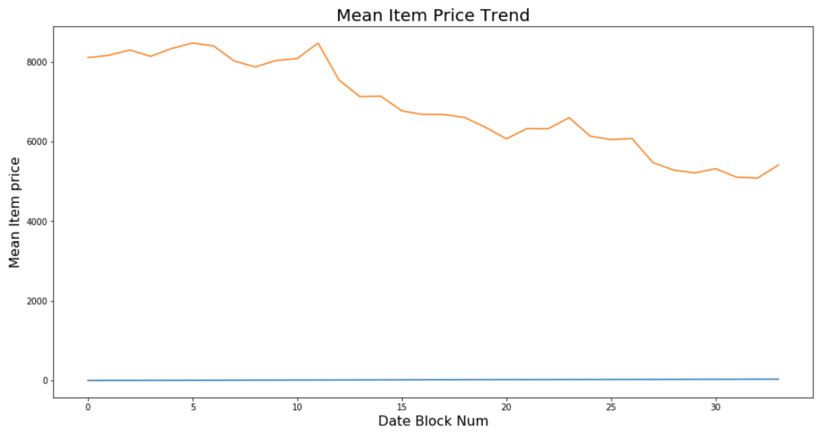
Some interesting information from test set analysis:

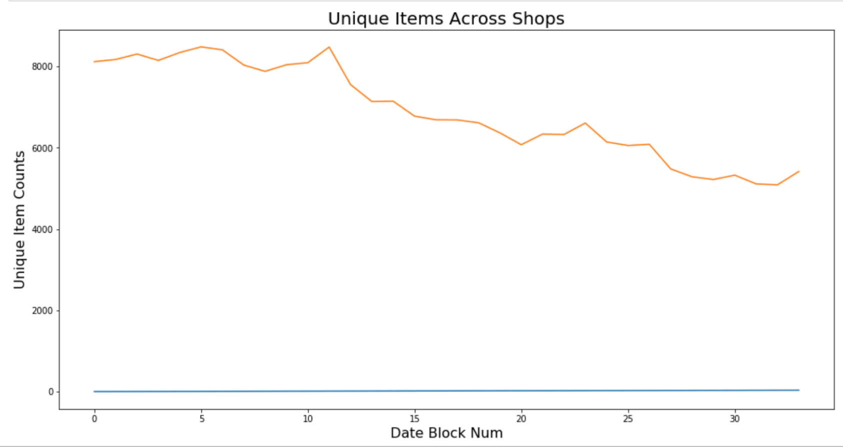
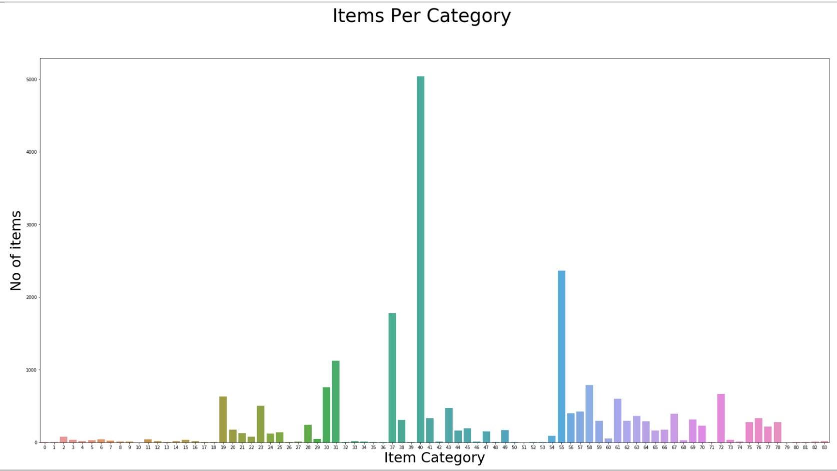
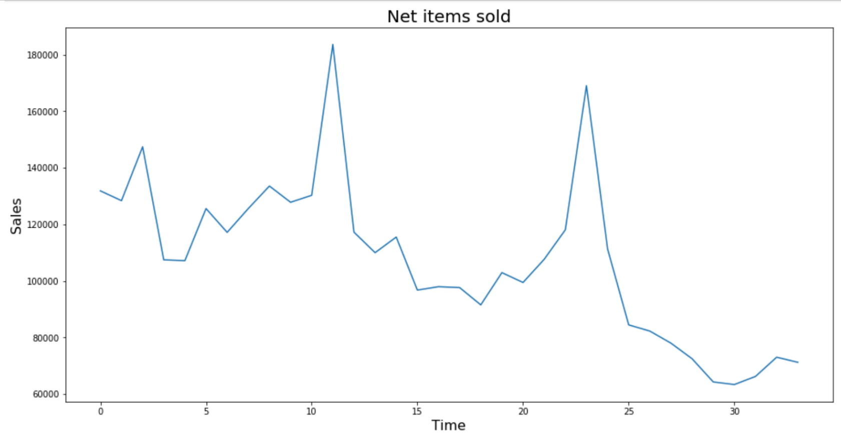
a. Not all shop_id in training set are used in test set. Test set excludes following shops (but not vice versa): [0, 1, 8, 9, 11, 13, 17, 20, 23, 27, 29, 30, 32, 33, 40, 43, 51, 54]

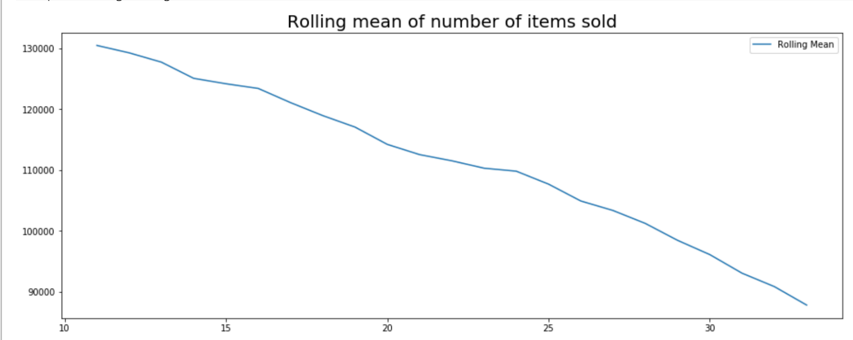
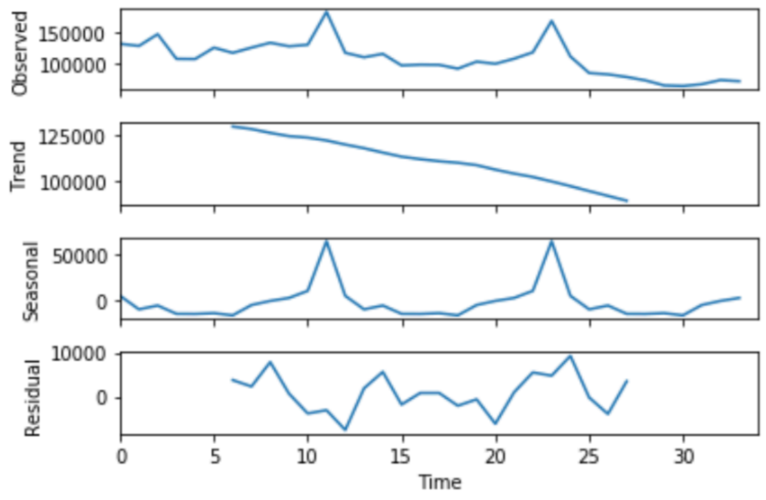
b. Not all item in train set are in test set and vice versa

c. In test set, a fixed set of items (5100) are used for each shop_id, and each item only appears one per each shop. This possibly means that items are picked from a generator, which will result in lots of 0 for item count.

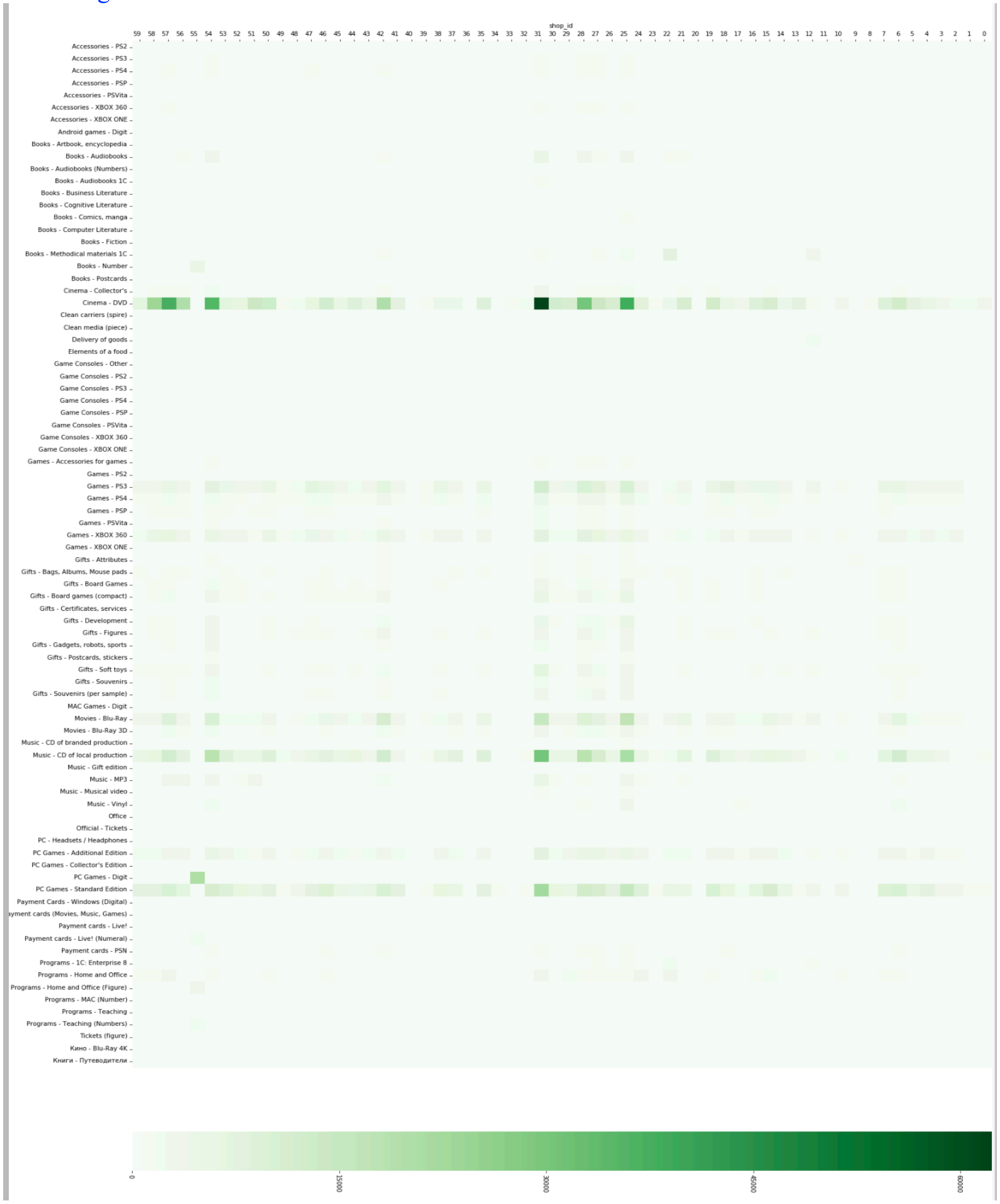
Exploratory Visualization

Monthly revenue trend indicates. 1. Company has obvious seasonality in their sales cycle during Nov-Dec. 2. Sales over the years increased in first 2 years but during its third year has started to slow down.	
Number of stores in business 1. It has stayed relatively stable over the years.	
Mean item price trend 1. Declining mean price indicates `company is either reducing prices to retain customers or to move inventory or company may be selling less premium products.	

Unique Items sold across shops 1. Declining may indicate customers are losing interests and company is unable to introduce new item categories and items.	
Items per Category shows 1. Out of 84 item categories there are handful of categories that have significant number of items. There are 5000 items in category_id 40.	
Number of sold items trend shows 1. Declining sales. 2. There are peaks in Nov-Dec and some seasonal behaviors during June-July-August. Seasonality maybe likely due to some national events such as – national holidays, or some other seasonal influence during these times of the year.	

Rolling mean of number of sold items indicates. 1. Declining sales for this business over the last 3 years.	
Here I have decomposed sales data into Trend, seasonality and residuals - using Statsmodels, a python library available for statisticians. 1. We see obvious "seasonality" trend 2. Sales peak around a certain time of the year - right around October and November.	 Statsmodel provide additive as well as multiplicative model. ● The additive model is useful when the seasonal variation is relatively constant over time. ● The multiplicative model is useful when the seasonal variation increases over time.

Item categories



& shops heatmap

Benchmark

We will use following 2 benchmark models.

1. Our base benchmark model will predict next month sales based on the rolling average sales of the previous similar months from the dataset.

2. Our target benchmark model to beat will be based on Prophet, a Facebook forecasting tool available in Python.

Prophet forecasting tool is an additive regression model with four main components:

- Prophet automatically detects changes in trends by selecting changepoints from the data.
- Prophet has yearly seasonal component modeled using Fourier series.
- Prophet has weekly seasonal component using dummy variables.
- Prophet has weekly can accept user-provided list of important holidays.

Comparing our results with above 2 benchmark models will allow us to figure out how well is our prediction model.

Limitations of Prophet forecasting tool –

a. It can only perform univariate analysis.

b. It takes 2 variable inputs – a datetime series (ds) and time sequence output y (corresponding numeric output).

c. For our project, we will aggregate all the monthly sales count as y and all date_block_num into sequential months starting with 0 to 33.

Implementation of benchmark model are provided in the associated jupyter notebook – PredictSales – Baseline.ipynb

Methodology

Data consists of daily sales for 60 stores, that sell 22,140 unique items in 84 different categories. We are given 34 months of daily sales data ~2.9 million rows.

Contrary to most established forecasting models used in the industry which are based on Moving Averages such as ARIMA (Auto Regressive Integrated Moving Average), I have attempted to use an almost fully automated approach based on deep neural networks.

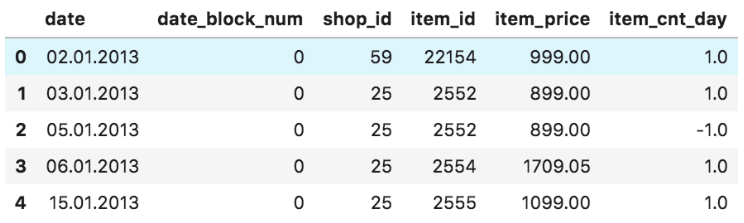
The architecture I have used is a simple deep neural network that takes in a sequence of sales data aggregated monthly for every single shop and item pair for all months and predicts the sales for every shop-item pair during the next month in sequence.

My approach could easily be adapted to other applications in which the goal is to predict a fixed-length output from a variable-length sequence.

The key to this methodology is to systematically engineer lagging sales indicator (item_sales_cnt for previous month) for each shop-item pair for every month as input data for deep neural network. Key idea here is to present neural network sales trend for every entry of shop_id and item_id pair and letting network figure out all the hidden patterns.

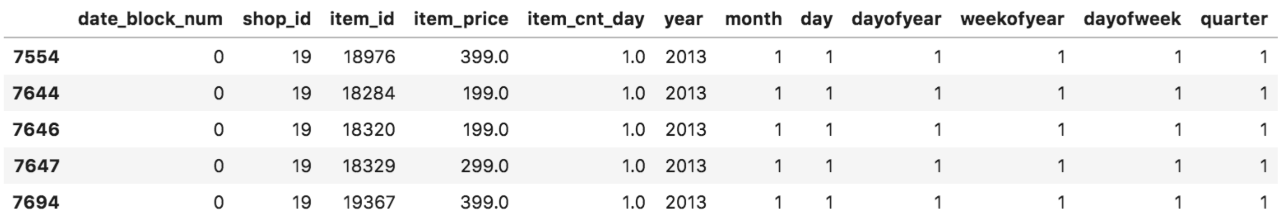
Data Preprocessing

I. Daily sales data that is made available to train our model looks as below –



	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

II. For data exploration and analysis, we will break down the date feature into individual feature components to explore daily, weekly, monthly, quarterly and yearly seasonality.



	date_block_num	shop_id	item_id	item_price	item_cnt_day	year	month	day	dayofyear	weekofyear	dayofweek	quarter
7554	0	19	18976	399.0	1.0	2013	1	1	1	1	1	1
7644	0	19	18284	199.0	1.0	2013	1	1	1	1	1	1
7646	0	19	18320	199.0	1.0	2013	1	1	1	1	1	1
7647	0	19	18329	299.0	1.0	2013	1	1	1	1	1	1
7694	0	19	19367	399.0	1.0	2013	1	1	1	1	1	1

III. Since we have to predict monthly sales numbers, we will aggregate daily sales data into monthly sales numbers before providing it as input to deep neural net model. We will also aggregate shop-item pairs for every month (date_block_num) and calculate the mean price of the item at that shop during that month(date_block_num).

	shop_id	item_id	date_block_num	month	year	item_cnt_day	item_price
103792	5	13334	33	10	2015	4.0	359.0
103791	5	13331	33	10	2015	1.0	399.0
103790	5	13331	33	7	2015	1.0	399.0
723715	25	13332	33	10	2015	6.0	1399.0
1760607	50	20742	33	10	2015	1.0	499.0

IV. We have used two separate deep neural network model implementations.

a. DNN with categorical entity embeddings.

b. DNN with monthly sales count lag features.

V. Categorical entity embeddings

In this approach we will take daily sales data which has date feature broken down into individual features – month, day, dayofyear, weekofyear, dayofweek and quarter and submit it to a sequential model with first layer being the embedding layer.

Our input will look as below and item_sales_cnt is our target.

	date_block_num	shop_id	item_id	item_price	item_cnt_day	year	month	day	dayofyear	weekofyear	dayofweek	quarter
7554	0	19	18976	399.0	1.0	2013	1	1	1	1	1	1
7644	0	19	18284	199.0	1.0	2013	1	1	1	1	1	1
7646	0	19	18320	199.0	1.0	2013	1	1	1	1	1	1
7647	0	19	18329	299.0	1.0	2013	1	1	1	1	1	1
7694	0	19	19367	399.0	1.0	2013	1	1	1	1	1	1

VI. Monthly sales count lag features

Lag features are values at prior time steps. We will generate lag features based on 'item_cnt_day' and grouped by 'shop_id' and 'item_id'. We will generate item sales count for every shop-id, item_id pair for all the months.

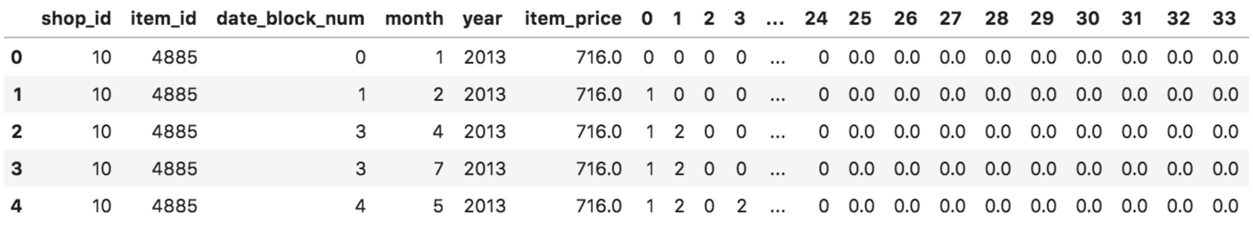
These lag features are the key engineered features we want our model to learn from.

Lag feature explanation –

For the very first month (date_block_num=0) – all the lag features (0,1,2,3,...33) will be 0 because at that point in time we will have no prior sales count information. So all rows of date_block_num=0 will have lag features set to 0. For the second month rows (date_block=1), we will populate the lag feature – '0' which indicates the previous month sales count, we will set the current month lag feature "1" and all future months (2 to 33) to 0 since we have no information about it.

Finally by 33rd month – we should have item_sales_cnt in 0,1,2,3,…32 for all rows where date_block_num=32 and column '33' lag feature will be 0.

Sample input data (X) to deep neural net should look as below –



	shop_id	item_id	date_block_num	month	year	item_price	0	1	2	3	...	24	25	26	27	28	29	30	31	32	33
0	10	4885	0	1	2013	716.0	0	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	10	4885	1	2	2013	716.0	1	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	10	4885	3	4	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	10	4885	3	7	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	10	4885	4	5	2013	716.0	1	2	0	2	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Our target variable will be item_sales_cnt - sales count for the next month in sequence.

Algorithms and Techniques

Common techniques used in the industry for demand forecasting involve techniques including both informal methods, such as educated guesses, and quantitative methods, such as the use of historical sales data and statistical techniques or current data. For the purpose of this project we will focus on quantitative method specifically using deep neural network technique to build our forecasting model.

Let's look at the 2 DNN models we are proposing in this project in detail.

1. DNN with Categorical Embeddings

Here we a build a feed-forward neural network and prepare the first layer of network with categorical variables using entity embeddings. To make a neural network work effectively on this type of problem which has many category features (item_categories, number of stores, items and days), we are proposing to use entity embedding to represent category features in a multi-dimensional space. It is inspired by semantic embedding in the natural language processing domain.

Traditionally in machine learning the two common methods for handling categorical variables are - one-hot encoding and label encoding. Entity embeddings is similar to label encoding but instead of assigning an integer to a category it comes up with a whole vector to represent the category. This vector can be of any size and has to be specified by the researcher.

There are couple of advantages of entity embeddings over one encoding and label encoding. One-hot encoding variables with many categories results in very sparse vectors, which are computationally inefficient and make it harder to reach optimization. Label encoding also solves this problem but can only be used by tree-based models.

Embeddings provide information about the distance between different categories. The beauty of using embeddings is that the vectors assigned to each category are also trained during the training of the neural network. Therefore, at the end of the training process we end up with a vector that represents each category. These trained embeddings can then be visualized to provide insights into each category.

The trained embeddings can be saved and used in non-deep learning models. For example, one could train the embeddings for categorical features each month and save the embeddings. These embeddings can then be used to train a Random Forest or a Gradient Boosted Trees model by loading the learned embeddings for the categorical features.

To build our model entity embeddings we have to

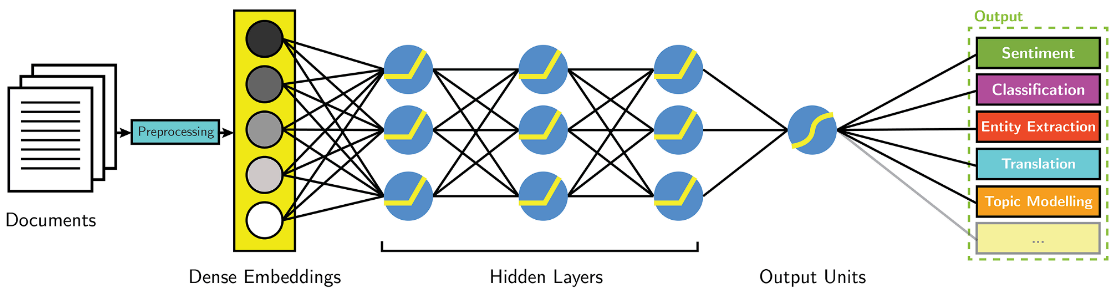
1. choose an optimal embedding size and

2. determine the maximum features in the model.

The embedding size refers to the length of the vector representing each category and can be set for each categorical feature. Similar to the tuning process of hyperparameters in a neural network, there are no hard rules for choosing the embedding size. Choosing the same embedding size for each category based on max distinct of all the categorical variables is an easy and transparent approach, but probably not the optimal one.

For a similar kaggle competition, Rossmann store sales prediction task, the researchers (who won the 3rd place) chose a value between 1 and M (the no. of categories) - 1 with a maximum embedding size of 10. For example, day of the week (7 values) gets an embedding size of 6, while store id (1115 values) gets an embedding size of 10. However, the authors have no clear rules of choosing the size between 1 and M-1.

Here we will try different values for embedding size and max_features to compare model convergence time and overall accuracy.



The idea of embeddings comes from learning them on words in NLP (word2vec), image taken from Aylien

Source - https://towardsdatascience.com/deep-learning-structured-data-8d6a278f3088

2. DNN with monthly sales count lag features

In this NN model we have removed the entity embedding layer and instead added engineered monthly sales count lag features for every shop_id and item_id pair.

Our objective here is let our model discover the trend for monthly sales count from historical price trend.

Implementation

We'll be using the popular data manipulation framework pandas and keras libraries to preprocess data and setup our deep learning neural network. Python library, pandas, among other things, allows you to manipulate tables/data frames in python as one would in a relational database using SQL (structured query language).

As a structured data problem, our data is already organized into dataframe but we will still have to go through all the typical cleaning and feature engineering as described in the previous Data Preprocessi and Data exploration section.

We have removed all the missing item_cnt (NaN) rows and also removed negative item_price and item_cnt, which may indicate returns in a particular shop, to keep it simple.

Our feature space will have a single dataframe that is prepared by joining the below tables.

sales train: training dataset containing daily sales data provided by competition

shops: list of stores

item categories: mapping of item ids to an item category.

Item: list of all items

We will also breakdown the date column into individual fields to help detect sales trends and seasonality.

All these new date features along with shop_id, item_id and item_category_id will be the key columns for categorical embeddings.

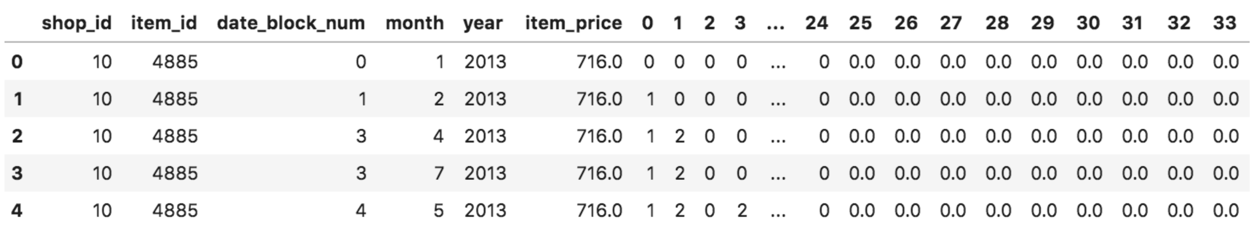
During implementation we figured the max_features value has to be any value greater than 400,000 as the model tries to map all the columns into entity embedding space.

```
# Embedding space

max_features = 400000

embedding_size = 200
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_4 (Embedding)      (None, 11, 200)           80000000  
_________________________________________________________________
flatten_4 (Flatten)          (None, 2200)              0         
_________________________________________________________________
dense_10 (Dense)             (None, 3)                 6603      
_________________________________________________________________
dropout_7 (Dropout)          (None, 3)                 0         
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 12        
_________________________________________________________________
dropout_8 (Dropout)          (None, 3)                 0         
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 4         
=================================================================
Total params: 80,006,619
Trainable params: 80,006,619
Non-trainable params: 0
_________________________________________________________________
```

3. DNN with monthly sales count lag features

These monthly lag features are values at prior time steps. We will generate lag features based on ‘item_cnt_day’ and grouped by ‘shop_id’ and ‘item_id’. We will generate item sales count for every shop-id, item_id pair for all the months.



	shop_id	item_id	date_block_num	month	year	item_price	0	1	2	3	...	24	25	26	27	28	29	30	31	32	33
0	10	4885	0	1	2013	716.0	0	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	10	4885	1	2	2013	716.0	1	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	10	4885	3	4	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	10	4885	3	7	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	10	4885	4	5	2013	716.0	1	2	0	2	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

These lag features are the key engineered features we want our model to learn from.

Lag feature explanation –

For the very first month (date_block_num=0) – all the lag features (0,1,2,3,...33) will be 0 because at that point in time we will have no prior sales count information. So all rows of date_block_num=0 will have lag features set to 0. For the second month rows (date_block=1), we will populate the lag feature – ‘0’ which indicates the previous month sales count, we will set the current month lag feature “1” and all future months (2 to 33) to 0 since we have no information about it.

Finally by 33rd month – we should have item_sales_cnt in 0,1,2,3,…32 for all rows where date_block_num=32 and column ‘33’ lag feature will be 0.

Sample input data (X) to deep neural net should look as below –

	shop_id	item_id	date_block_num	month	year	item_price	0	1	2	3	...	24	25	26	27	28	29	30	31	32	33
0	10	4885	0	1	2013	716.0	0	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	10	4885	1	2	2013	716.0	1	0	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	10	4885	3	4	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	10	4885	3	7	2013	716.0	1	2	0	0	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	10	4885	4	5	2013	716.0	1	2	0	2	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Our target variable will be item_sales_cnt - sales count for the next month in sequence.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_89 (Dense)             (None, 128)               4992      
_________________________________________________________________
dropout_31 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_90 (Dense)             (None, 64)                8256      
_________________________________________________________________
dropout_32 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_91 (Dense)             (None, 1)                 65        
=================================================================
Total params: 13,313
Trainable params: 13,313
Non-trainable params: 0
_________________________________________________________________
```

Implementation of benchmark model are provided in the associated jupyter notebook – PredictSales – DeepNeuralNetwork.ipynb

Implementation of Neural Network with Embeddings model are provided in the associated jupyter notebook – PredictSales – DNN-CategoricalEmbeddings.ipynb

Implementation of Neural Network with Embeddings model are provided in the associated jupyter notebook – PredictSales – DeepNeuralNetwork.ipynb

Refinement

To further refine the model,

1. We can add Russian holidays as additional one hot encoded feature.
2. Since new shop_id's start showing up randomly, we can try to identify that as a separate feature to see if it is important for the model. Correspondingly we can try to identify shops that have not generated any sales for more than 6 months and try to mark them as closed.
3. We can translate shop names and item names to English and perform "most favored shop" analysis based on most to least revenue contribution by shop to company's total revenue.

Results

Model Evaluation and Validation

Of the two neural network models, we tried, neural network with lagging sales indicator appeared to be comparable to benchmark. We tried optimizing our model using kernel_optimizer set to he_normal, adding batch normalization and trying out adam and rsmprop optimizer to get the best RMSE score of 6.305 with adam optimizer and learning rate of 0.001.

Below we have evaluated models based on the predicted sales count and actual sales count for last 6 months of data in the set.

Model	Root Mean Square Error	Mean Square Error	Mean Absolute Error
Moving Average	n/a	n/a	4546
Prophet	n/a	n/a	15753
DNN /w Entity Embeddings	9.75	95.16	16240
DNN /w Lag Features	6.305	39.758	740

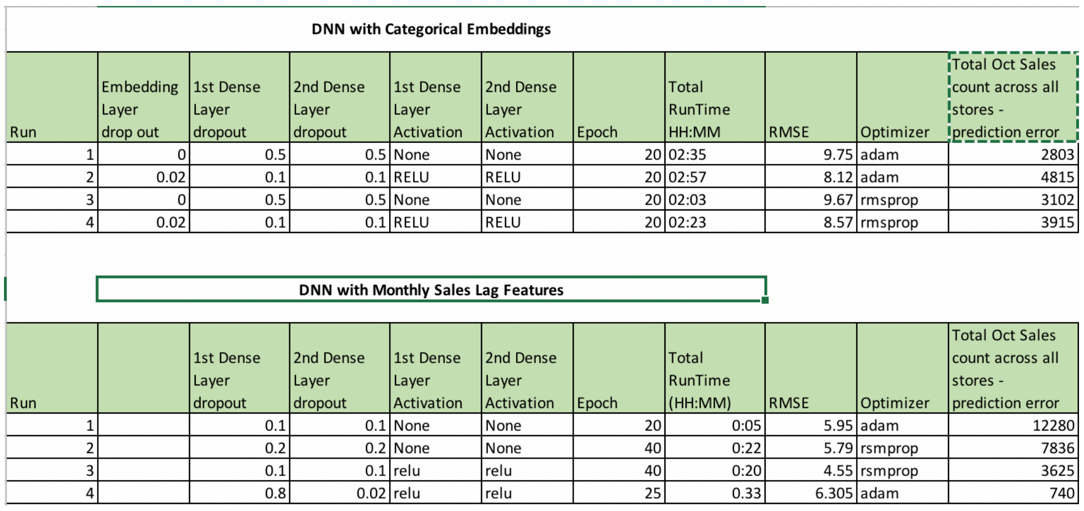
The four different models we used for testing, moving average and prophet were clearly the fast of all but for this data we could only use them to do aggregate forecast across all the stores.

DNN with entity embedding model did not fair well in terms of run time as well as prediction. I believe the feature space (shop_id,item_id and date features) we tried embedding did not very well represented the pattern I initially anticipated. For the subsequent rounds I would propose to use shop_name, item_name and item_category_names instead of their ids.

Justification

DNN with lag features fared better compared to DNN with Categorical embeddings, Moving Average and Prophet. Moving average was the quickest and least cumbersome.

The reason DNN better fared maybe due to monthly lag features captured the sales pattern for shop-item over 33 months better than just date features embedded in daily sales data.



DNN with Categorical Embeddings

Run	Embedding Layer drop out	1st Dense Layer dropout	2nd Dense Layer dropout	1st Dense Layer Activation	2nd Dense Layer Activation	Epoch	Total RunTime HH:MM	RMSE	Optimizer	Total Oct Sales count across all stores - prediction error
1	0	0.5	0.5	None	None	20	02:35	9.75	adam	2803
2	0.02	0.1	0.1	RELU	RELU	20	02:57	8.12	adam	4815
3	0	0.5	0.5	None	None	20	02:03	9.67	rmsprop	3102
4	0.02	0.1	0.1	RELU	RELU	20	02:23	8.57	rmsprop	3915

DNN with Monthly Sales Lag Features

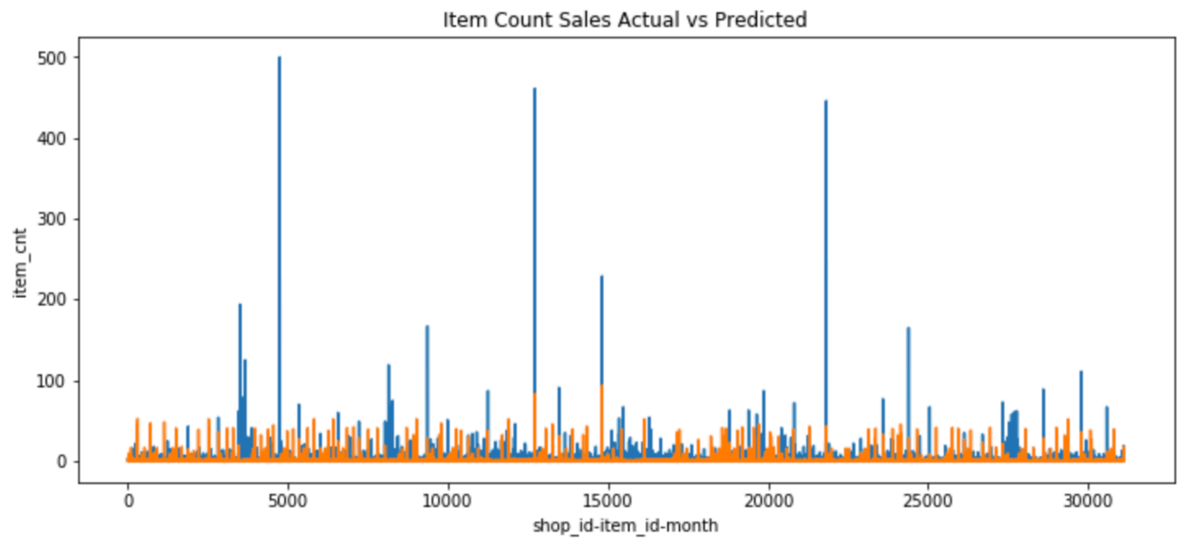
Run		1st Dense Layer dropout	2nd Dense Layer dropout	1st Dense Layer Activation	2nd Dense Layer Activation	Epoch	Total RunTime (HH:MM)	RMSE	Optimizer	Total Oct Sales count across all stores - prediction error
1		0.1	0.1	None	None	20	0:05	5.95	adam	12280
2		0.2	0.2	None	None	40	0:22	5.79	rsmprop	7836
3		0.1	0.1	relu	relu	40	0:20	4.55	rsmprop	3625
4		0.8	0.02	relu	relu	25	0.33	6.305	adam	740

Table 1.1 – DNN experiment result tabulation

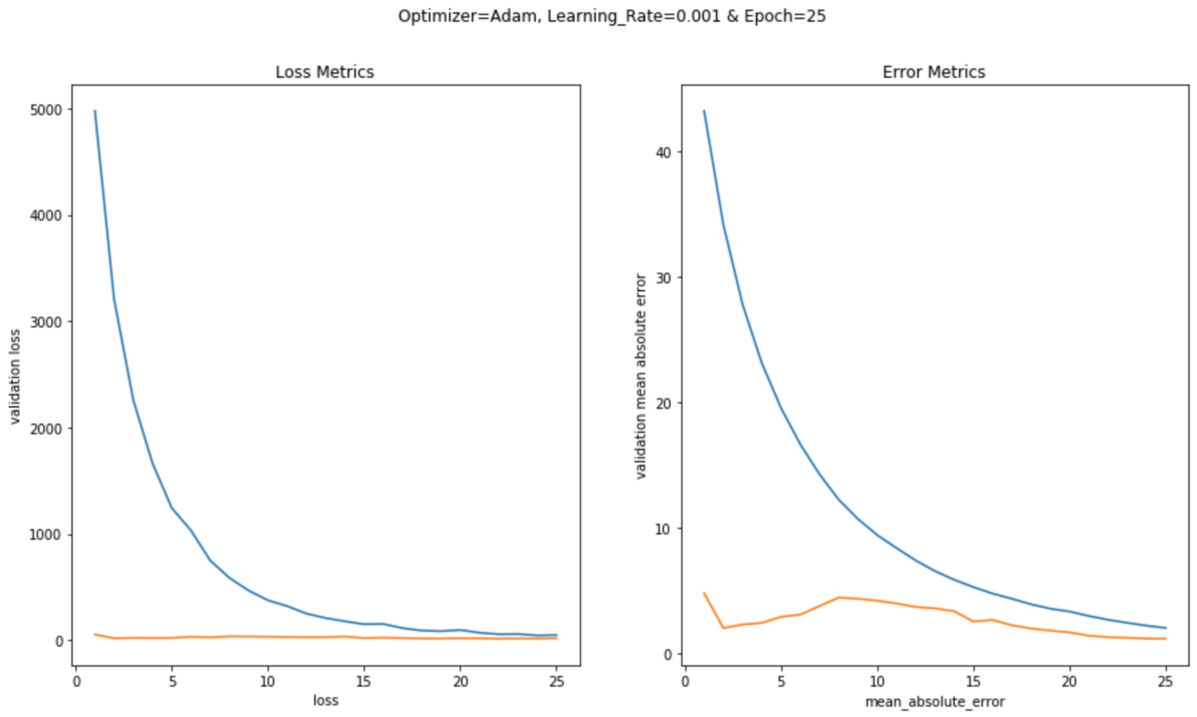
Visualization

DNN with lag features was overall best in terms of predicting monthly sales for shop_id and item_id, we tested the DNN model with number of different units, kernel_optimizer and batch_normalization. Our goal was to get training loss and validation_loss to converge but even with higher number of epochs I noticed convergence was not as clear as I would have expected.

To test models general robustness we tested the prediction accuracy per shop_id, item_id pair on 33rd month sales count. Model prediction for total sales count across sales compares well with actuals but does not capture the individual patterns of items at shops for certain seasonal months.



```
Total sales across all shops during October 2015 is 62827.0
Deep neural network model predicted 63567.0
Mean Absolute Error of 1.4545877591728562
```



Conclusion

Reflection

In this project, my goal was to apply techniques that are generally popular in image recognition and classification problems with high dimensional feature set. In this project we tried extracting features from historical sales figures. Our goal was to come with feature to capture trends and seasonality etc. into every row that go as input to NN. Our final model fell below my expectations, but I believe there is lot of room for improvement. If we can engineer features that capture knowledge and intuition of people responsible for forecasting, we can come up with a NN based forecasting tool that will come very close to something that has both qualitative as well quantitative expertise to predict accurately.

My number one challenge was the size of the data and using pandas to perform complex join, grouping, aggregation. Framing the data, the I way I was intending to be was way more difficult than I anticipated. Coming from database and data background I found working with pandas challenging and cumbersome but at the same time it was very rewarding. I now have an additional toolset under my belt. I discovered the whole process of preparing data and feature engineering as more than half the battle for such problems.

My ultimate goal was to try emerging techniques on age old problem in the industry using structured data. Enterprises today have tons of structured data that are hiding patterns that we can uncover using powerful technique such as NN.

Improvement

There were two particular improvements we can try with this project.

1. Define a model that will concatenate categorical embedding layer with dense layer.

   With 60 shops and 22k items and various categorical date features made feature set huge and model very slow but it did not improve the model’s prediction accuracy. I believe categorical embedding layer on entire input set wasn’t as helpful as I initially anticipated.

Instead we should try to build a categorical embedding layer of just shop_id,item_id, item_price and dayofyear and then concatenate rest of the features before submitting as input to dense layers.

2. Neural network model with just lagging sales count indicator doesn't capture all the nuances, additional features such holidays, shops popularity and items popularity etc., elapsed time between when an item is sold at the same shop again may also improve the NN model.

3. LSTM/RNN based network model may better capture the time-series pattern.

References

- An introduction to Deep Learning for Tabular Data - http://www.fast.ai/2018/04/29/categorical-embeddings/
- An Introductory Study of Time-Series Modeling and Forecasting - https://arxiv.org/pdf/1302.6613
- Understanding LSTM Networks - http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Avoiding Look Ahead Bias in Time Series Modelling - https://www.datasciencecentral.com/profiles/blogs/avoiding-look-ahead-bias-in-time-series-modelling-1
- Prophet: forecasting at scale - https://research.fb.com/prophet-forecasting-at-scale/
- An Introduction to Time Series Forecasting with Prophet Package in Exploratory - https://blog.exploratory.io/an-introduction-to-time-series-forecasting-with-prophet-package-in-exploratory-129ed0c12112
- Sales Forecast in E-commerce using Convolutional Neural Network https://arxiv.org/pdf/1708.07946.pdf
- Deep Learning Structured Data https://towardsdatascience.com/deep-learning-structured-data-8d6a278f3088