Predict Future Sales

Group 3
Dongmei Yin | Sandhya Santhanam | Qiong Wu | Sindhana Krishnamurthy | Manas Abhyankar

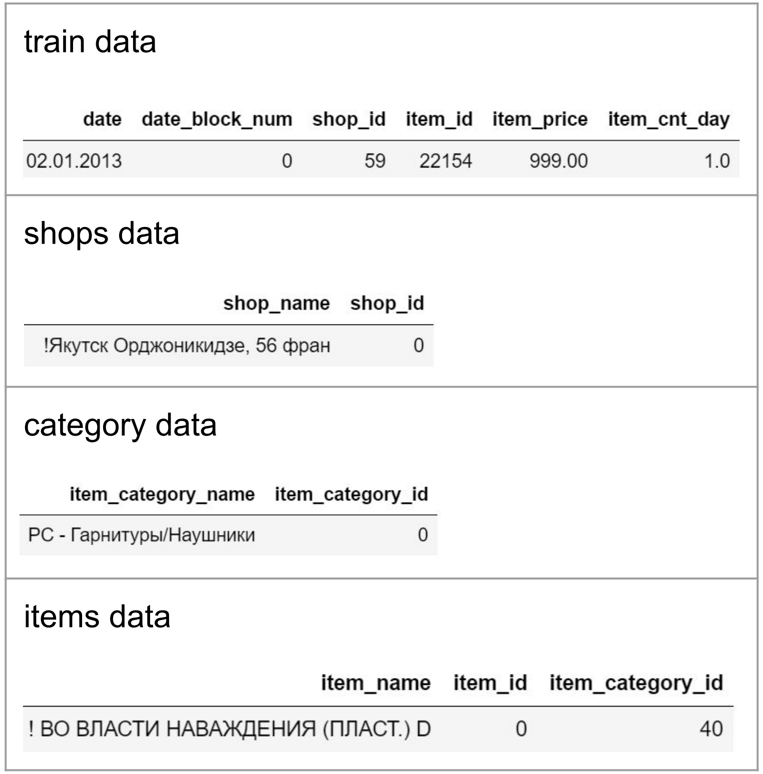
Introduction

- In this project we are working with a challenging time-series dataset that consists of monthly sales data provided by a Russian software firm (1C Company).
- Performed lots of data cleaning, pre-processing, and feature engineering to extract the maximally important features from the existing dataset.
- Implementation of various ML algorithms, including:
  - LightGBM
  - CatBoost
  - XGBoost
  - DNN
  - Naive Bayesian
  - LSTM
- Usage of utility algorithms like ARIMA and PCA for dimensionality reduction and time series data analysis.
- The goal of this project is to predict future sales for the month of November 2015.

Dataset

The dataset consists of real world historical sales data with the following datafields:

- ID - an Id that represents a (Shop, Item) tuple within the test set
- shop_id - unique identifier of a shop
- item_id - unique identifier of a product
- item_category_id - unique identifier of item category
- item_cnt_day - number of products sold. You are predicting a monthly amount of this measure
- item_price - current price of an item
- date - date in format dd/mm/yyyy
- date_block_num - a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1,..., October 2015 is 33
- item_name - name of item
- shop_name - name of shop
- item_category_name - name of item category



train data

date	date_block_num	shop_id	item_id	item_price	item_cnt_day
02.01.2013	0	59	22154	999.00	1.0

shops data

shop_name	shop_id
!Якутск Орджоникидзе, 56 фран	0

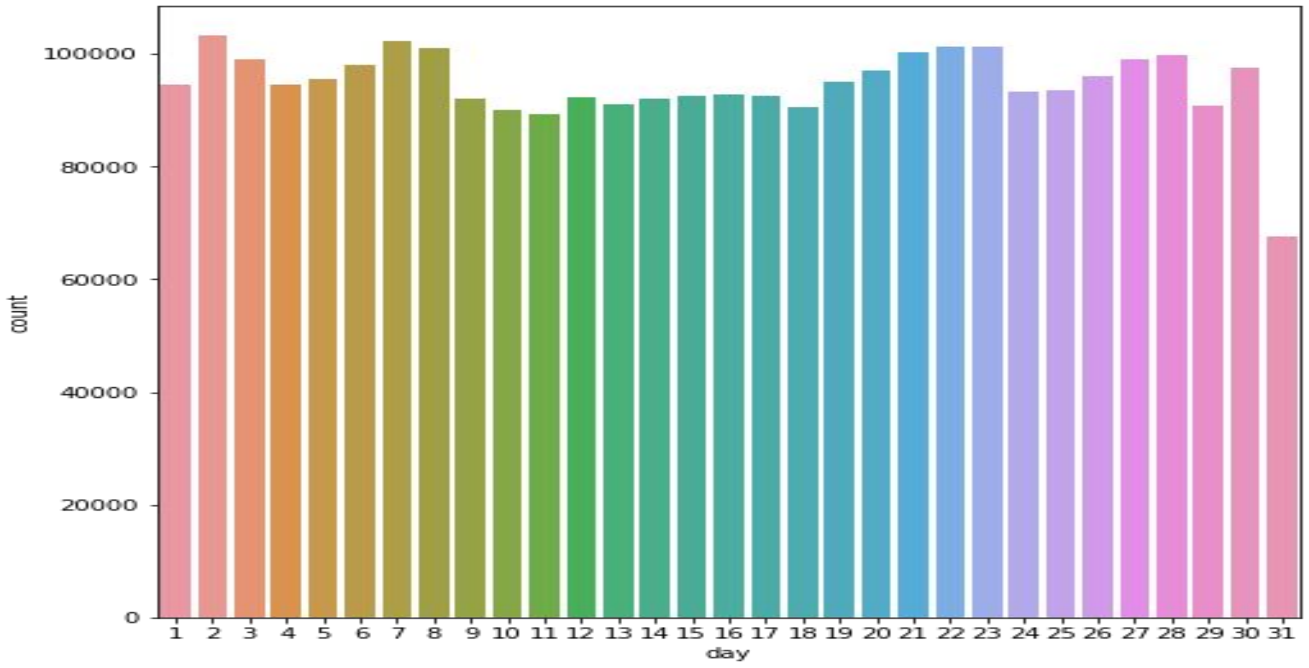
category data

item_category_name	item_category_id
PC - Гарнитуры/Наушники	0

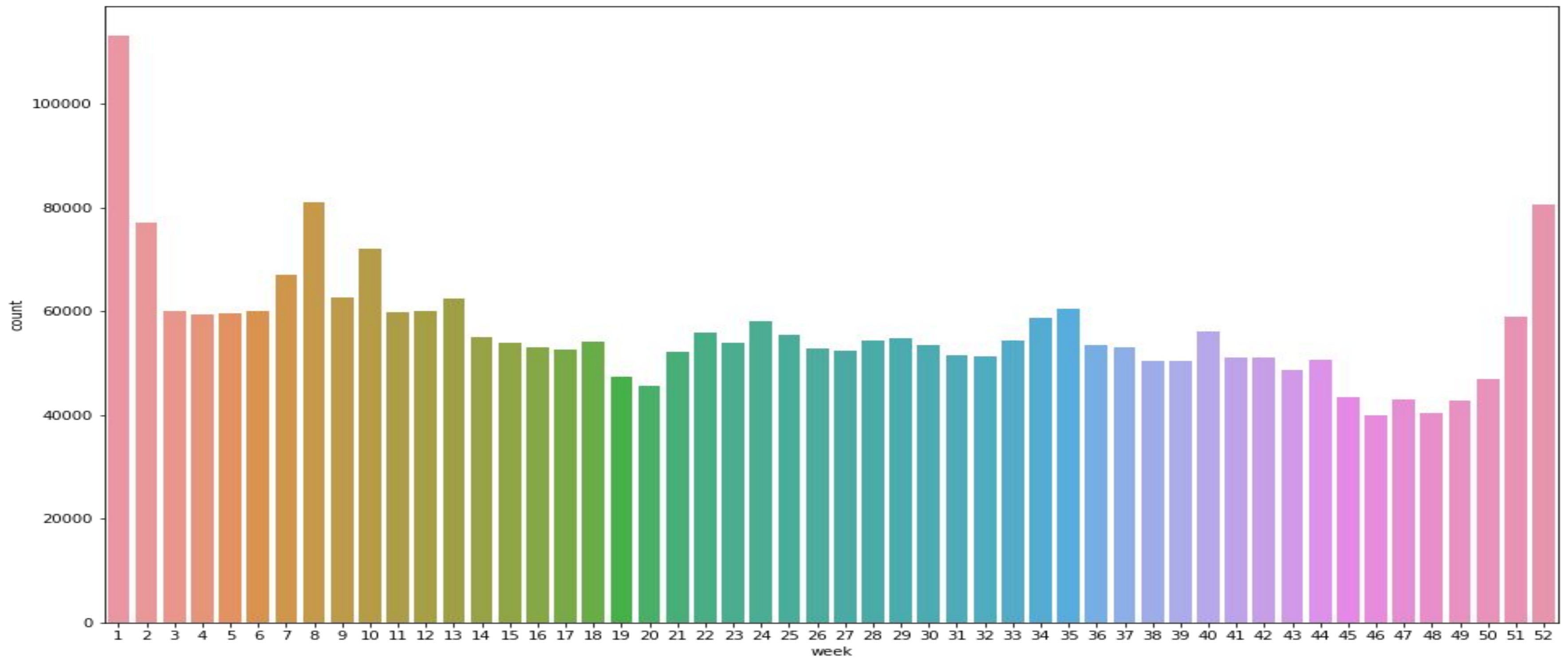
items data

item_name	item_id	item_category_id
! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40

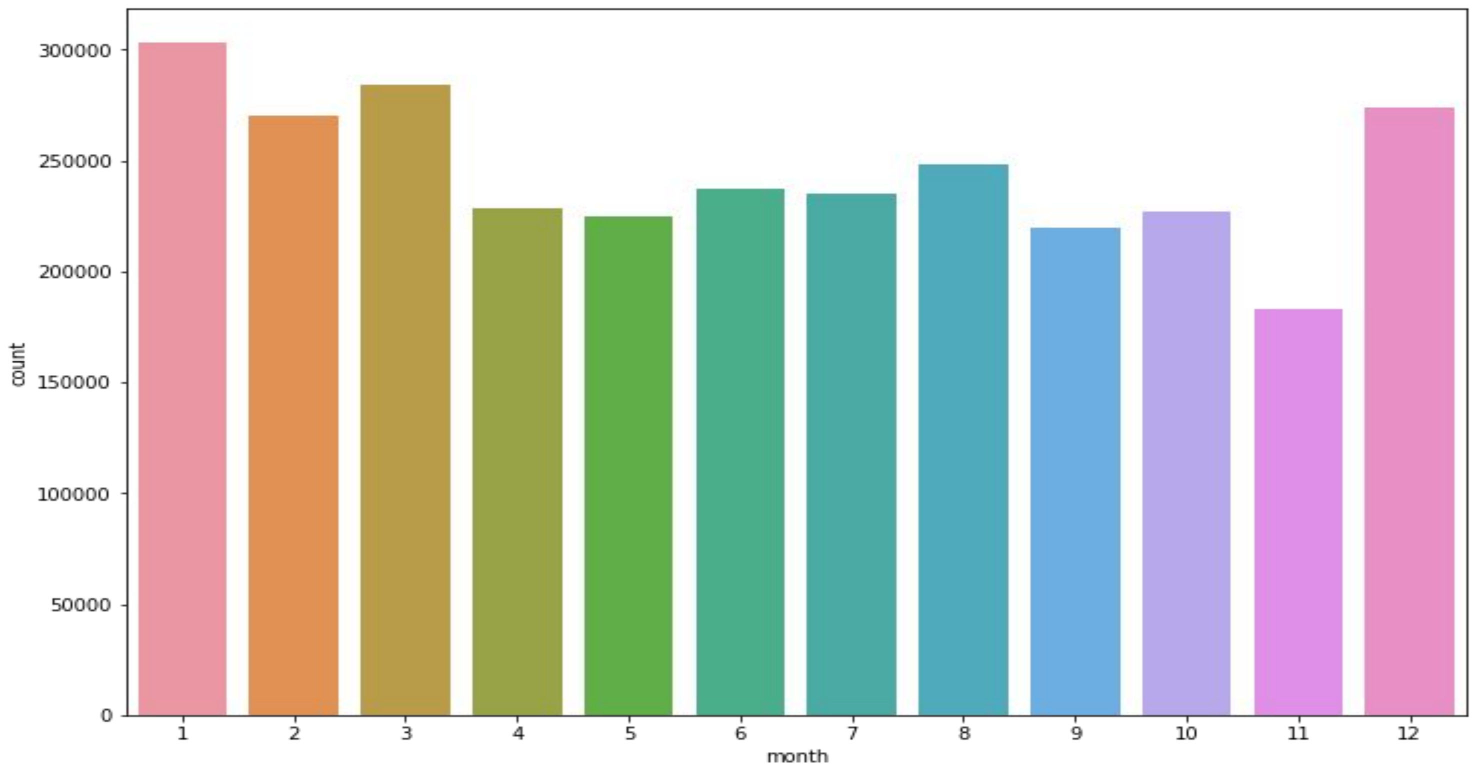
Visualization



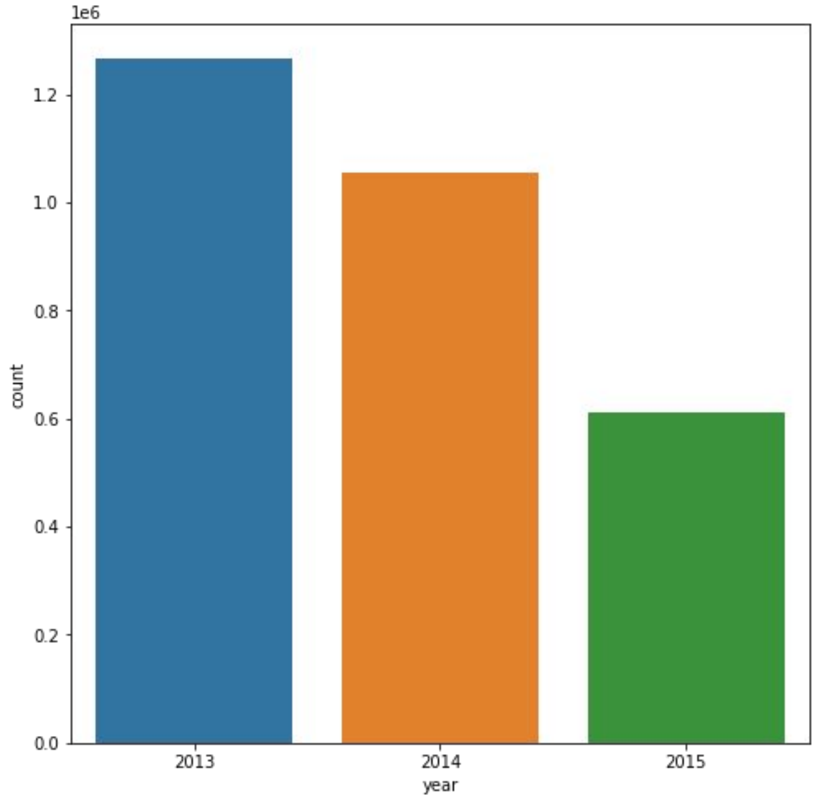
Daily sales trend



Weekly sales trend

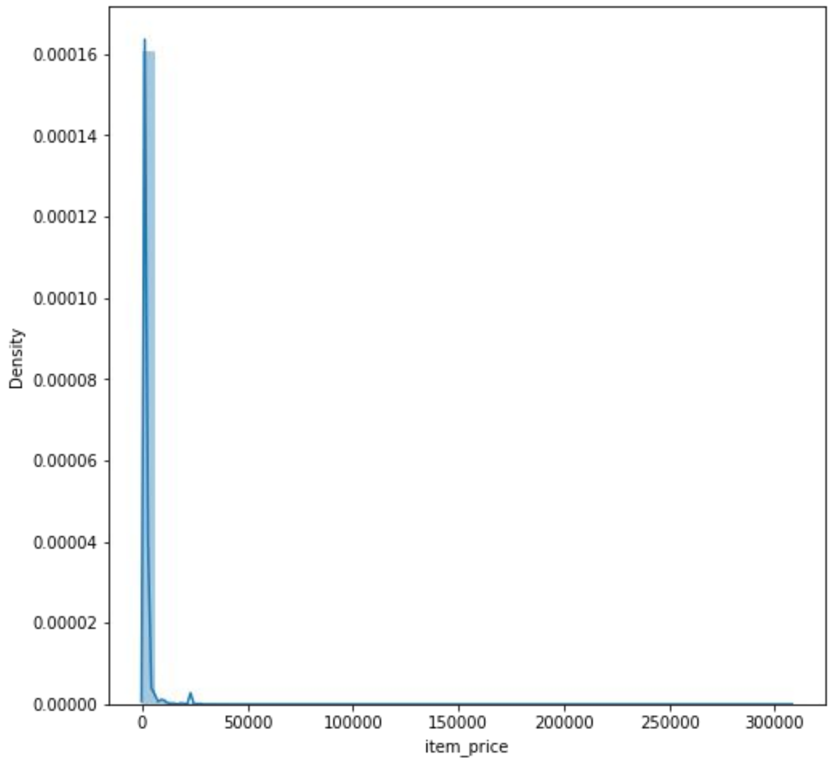


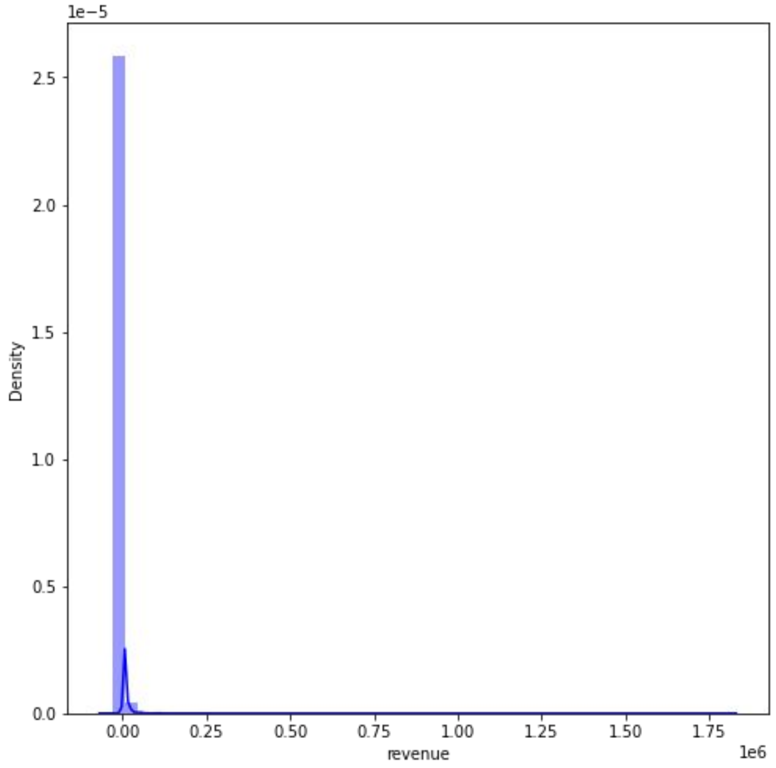
Monthly sales trend



Yearly sales trend

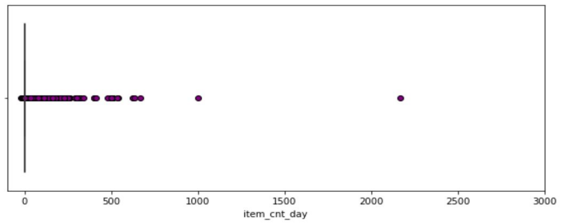
Distribution of Item price vs Revenue

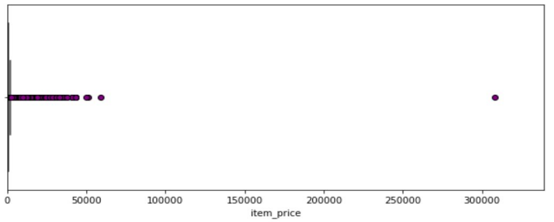




Data Cleaning

- Remove Outliers
    - Count of item sold in one day
    - Price of item
- Shop Name Cleanup
    - Combine same shops with different names
    - Derive shop city and category from shop name (first and second word in shop name)
    - Combine shop categories, with < 5 shops, into one single category
- Item Category Name Cleanup
    - Derive category type and subtype
- Item Name Cleanup
    - Lowercase, remove special characters & white spaces
    - Derive item type and name2



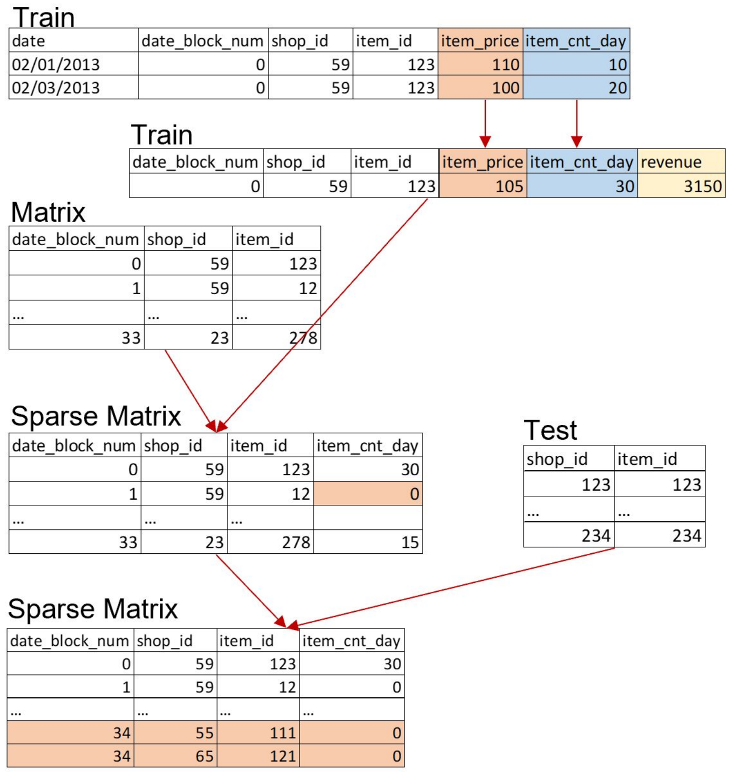


item_category_name	type	subtype
Аксессуары - PS2	Аксессуары	PS2
Game - PS2	Game	PS2

item_name	name2	type
007 Legends [Xbox 360, русская версия]	Xbox 360, русская версия Xbox 360, Russian Version	Xbox 360

Data Transformation

- Label encoding
  - City, Shop category, Item category type, Item category subtype, Item type, Item name2
- Aggregate train data from daily record to monthly record
- Create a sparse matrix with all month-shop-item combinations
- Join sparse matrix with train data
- Append test data to the matrix
- Join matrix with items, category, shops data



Train

date	date_block_num	shop_id	item_id	item_price	item_cnt_day
02/01/2013	0	59	123	110	10
02/03/2013	0	59	123	100	20

Train

date_block_num	shop_id	item_id	item_price	item_cnt_day	revenue
0	59	123	105	30	3150

Matrix

date_block_num	shop_id	item_id
0	59	123
1	59	12
…	…	…
33	23	278

Sparse Matrix

date_block_num	shop_id	item_id	item_cnt_day
0	59	123	30
1	59	12	0
…	…	…	
33	23	278	15

Test

shop_id	item_id
123	123
…	…
234	234

Sparse Matrix

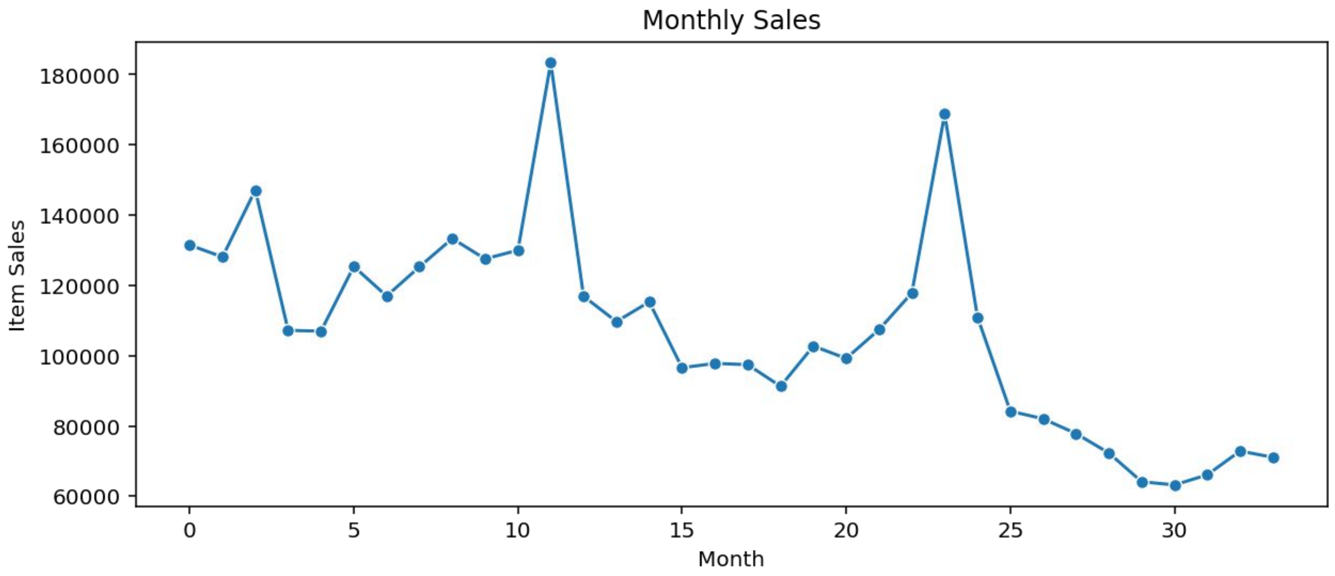
date_block_num	shop_id	item_id	item_cnt_day
0	59	123	30
1	59	12	0
…	…	…	…
34	55	111	0
34	65	121	0

Time-Series Analysis/Related Work

- Data that is collected in regular time intervals
- Components of a Time-Series:
    - Trend
    - Seasonality

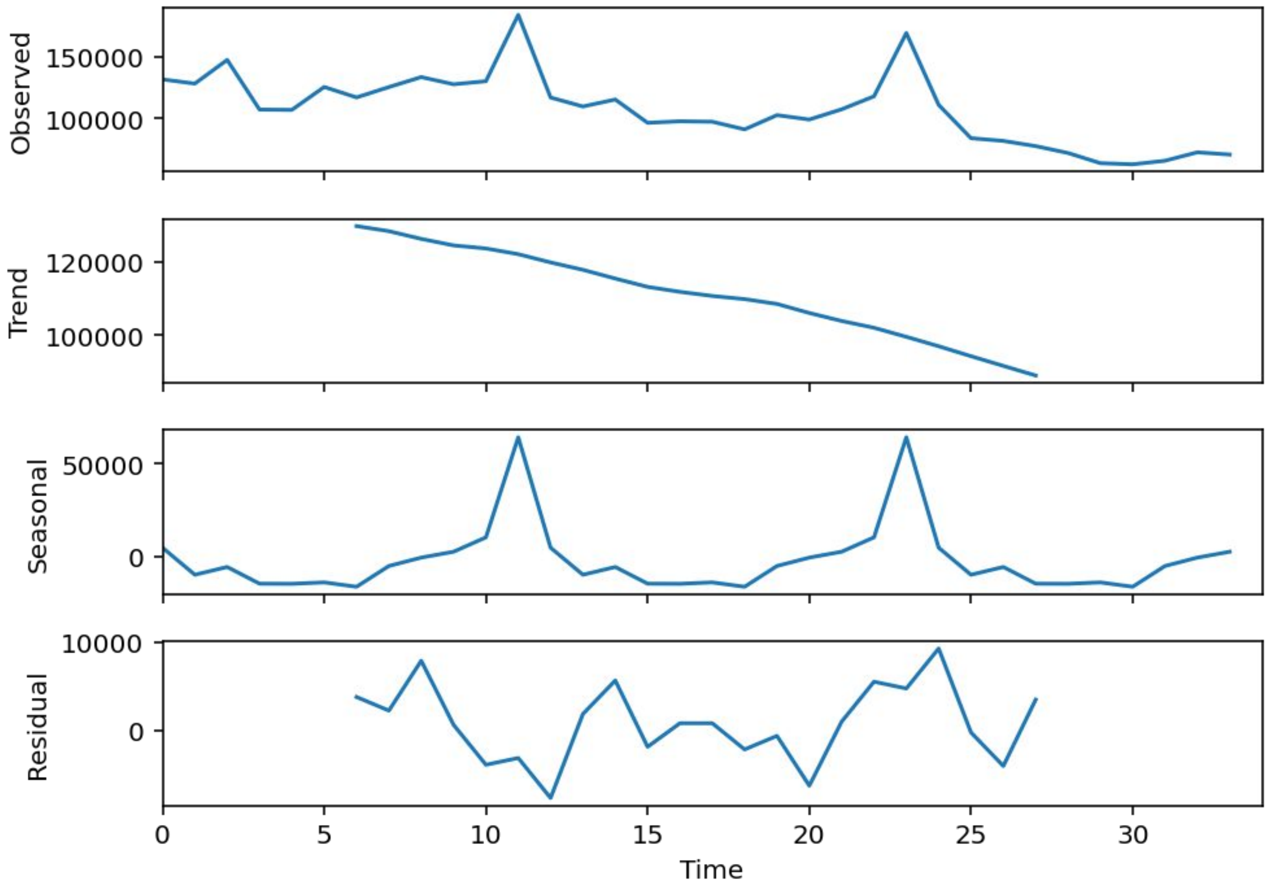
- ARIMA for time-series forecasting
    - Visualize data
    - Test if data is stationary
    - Plot ACF and PACF
    - Build ARIMA model

Visualize the data

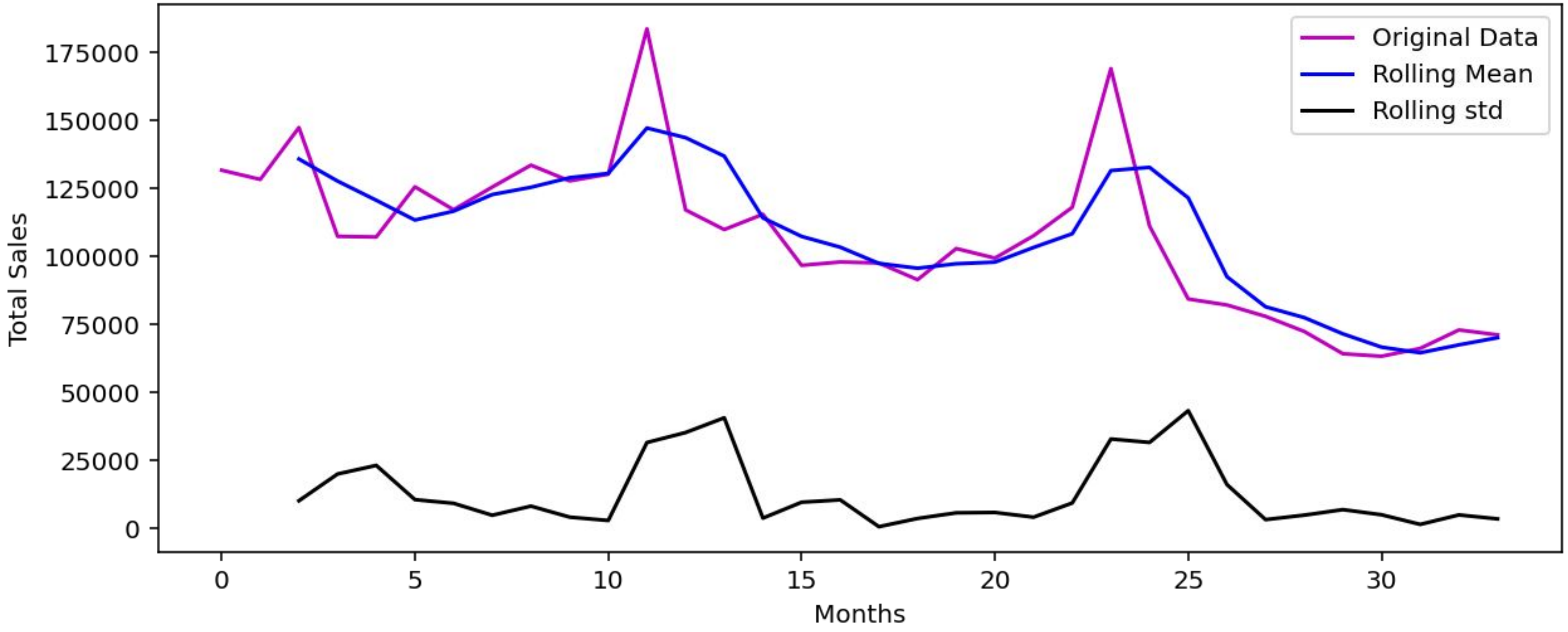


- Peak in sales around months 11 and 23
- Gradual decrease in sales over time

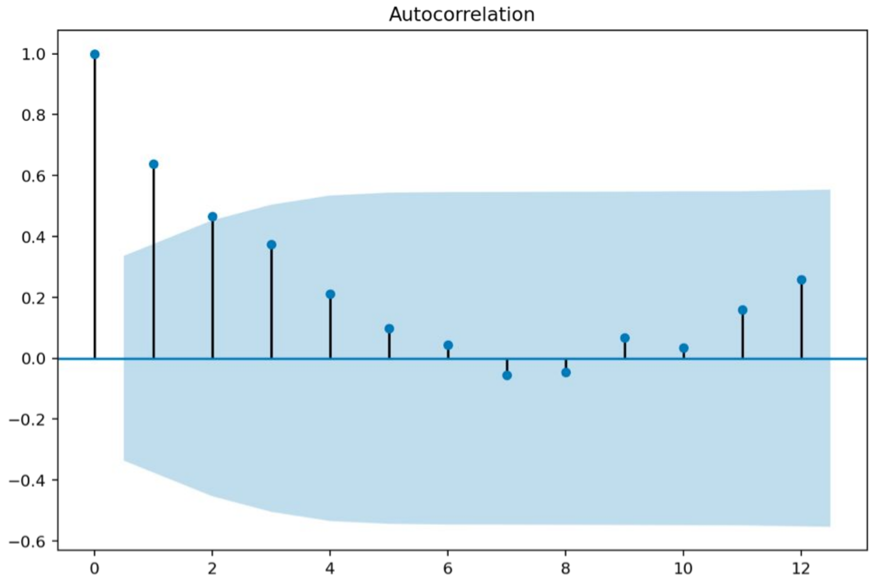
Trend and Seasonality

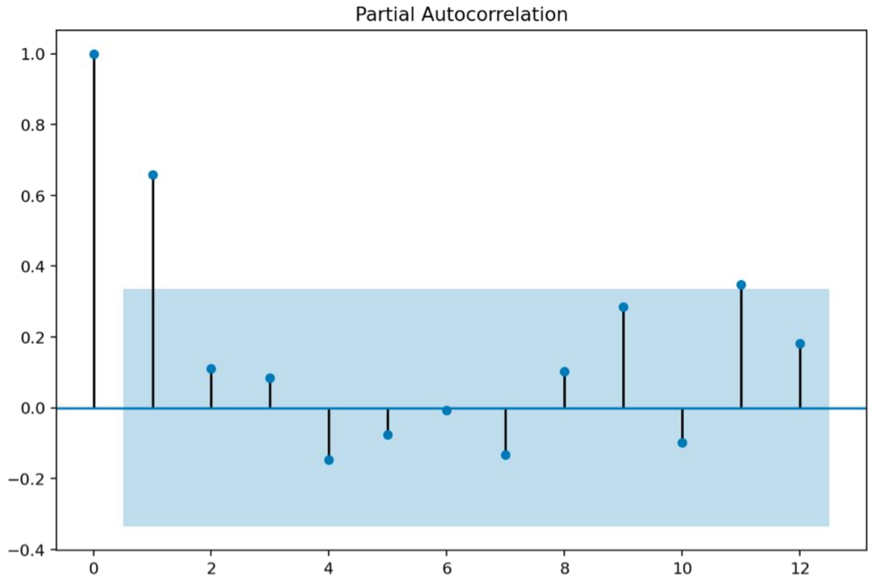


Test for Stationarity of data

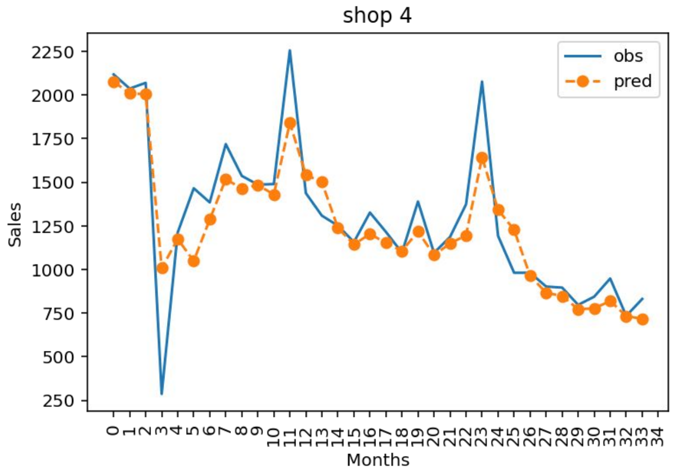


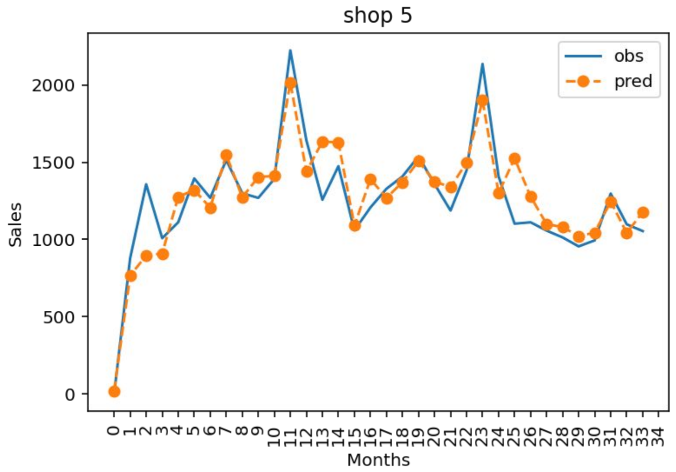
ACF and PACF Plots - p,d and q in ARIMA

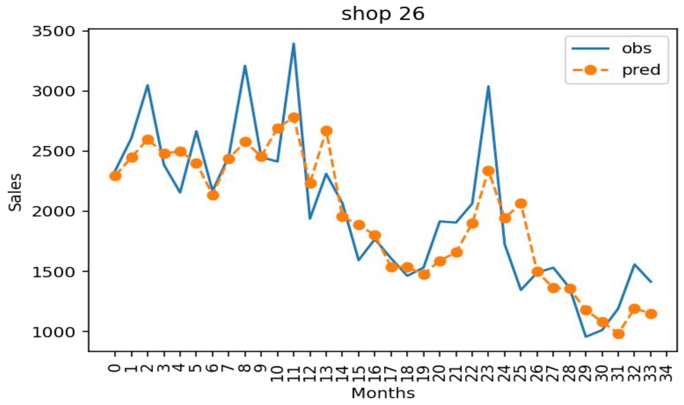


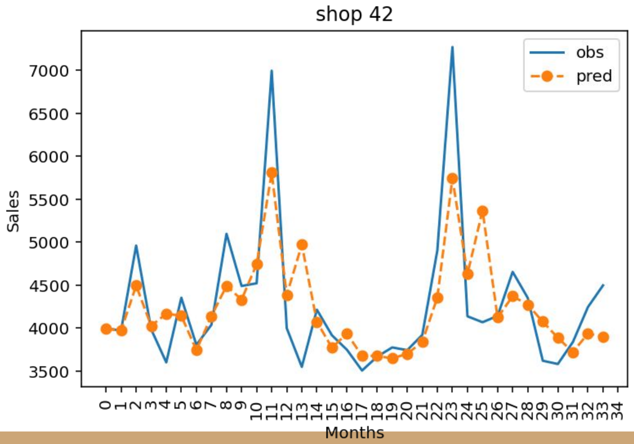


ARIMA results on the training set





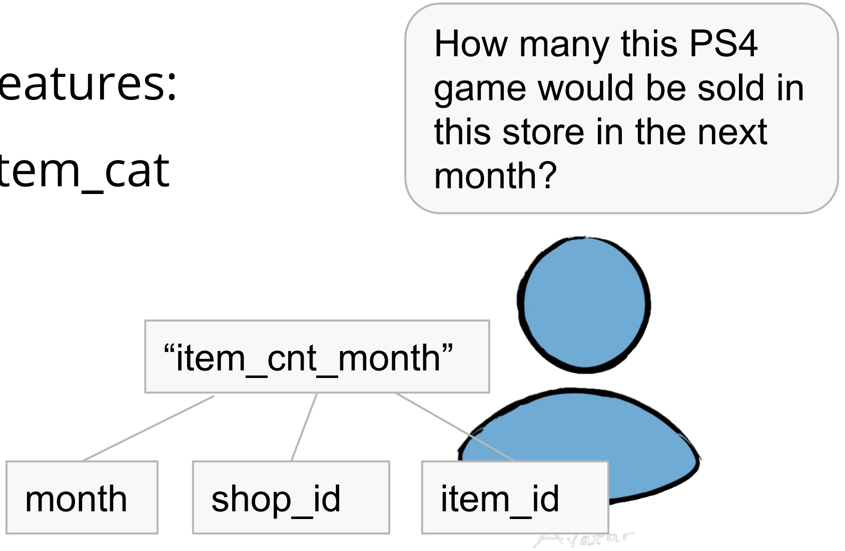




Feature Engineering 1

Aggregated item count and price related features:

- average item count per month shop-item_cat
- average item count per month city
- average item count per month shop
- avg price per year item_cat
- avg price per year item
- ....

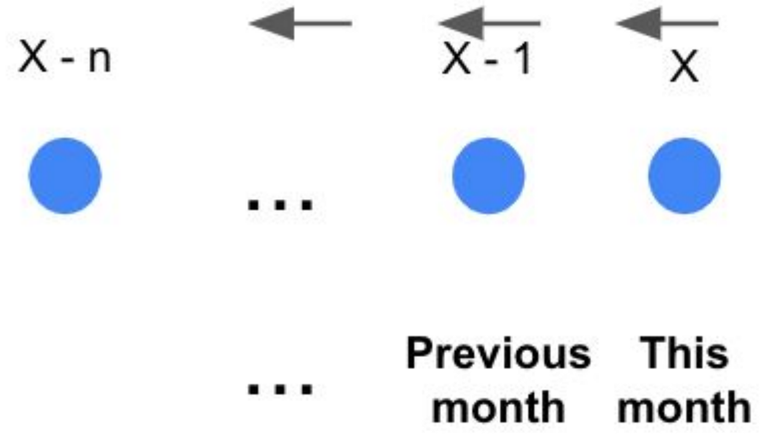

How many this PS4 game would be sold in this store in the next month?
“item_cnt_month”
month
shop_id
item_id

*These features possibly caused multicollinearity. When we drop some of those features, the model performs better.

Feature Engineering 2

Time lag features:

- Delta _price_lag
  - Trace back 3 months for price change
  - Use the nearest one as Delta_price_lag
- Item_cnt_month_lag_1
- Item_cnt_month_lag_2
- ...
- Item_cnt_month_lag_6
  - Trace back 1 - 6 months for item_cnt_month as lag features



Feature Engineering 3

Date and seasonality related features:

- Month (0 - 11)
- Year (0 - 2)
- Season (0 - 3)
- Number of days in the month (28, 30, 31)
- Russian Holidays

Proposed Solutions

- XGBoost
  a. Pros:
     i. Regularization.
     ii. Parallel Processing.
     iii. Cross-validation (determine the perfect # of boosts).
     iv. Accurate results with low RMSE and useful **feature importance** plots available.
  b. Cons
     i. Not great at non-classification problems (generally, not an issue in our project).
     ii. Resource intensive.
        1. Takes a LOT of RAM for large datasets.
     iii. Advent of LightGBM leaves much to be desired for computation speed.
- CatBoost
  c. Pros:
     i. Out-of-the-box categorical feature support (less time spent pre-processing data).
     ii. Default parameters work great for the CatBoostRegressor. Thus, parameter tuning is minimized.
  d. Cons
     i. That said, model is new and lacks some tuning capabilities compared to XGBoost/LightGBM.
     ii. Only superior in datasets with lots of categorical features.

Proposed Solutions

- CNN
  a. Pros:
     i. “Auto-extraction” of important features.
     ii. Simple to set-up.
     iii. Very flexible models to accommodate any size of dataset.
     iv. With minor tuning, can achieve sizable increases in accuracy.
  b. Cons
     i. Computationally expensive to run for epoch sizes greater than a few 100 (and RAM hungry).
     ii. Even with data augmentation, can run into overfit situations when the data size is small.
- LSTM
  c. Pros:
     i. Superior to RNNs since it drastically reduces the problem of the vanishing gradient.
     ii. Can retain info about the data for a long period of time (the L & T of LSTM).’
     iii. Excellent at making predictions for time-series data.
  d. Cons
     i. Computation time can be long and the cost can be expensive (in terms of system resources).
     ii. Prone to overfitting, and applying dropouts to resolve overfitting is difficult.
     iii. Is sensitive to random weight initializations and requires small weights when training begins.

Proposed Solutions

- LightGBM
  a. Pros:
     i. Compared to XGBoost, very light on system resources due to its unique binning process.
     ii. Boasts an incredibly fast run-time in comparison to XGBoost, and can be tuned to achieve similar performance (from experience, a default parameter LightGBM model can run 200 epochs in 30 seconds).
     iii. Performs equally well on datasets of either a large or small size.
     iv. Can also be GPU accelerated as XGBoost is.
  b. Cons:
     i. Very prone to overfitting, more so than RandomForests and XGBoost since it grows leaf-wise.
     ii. Documentation is good, but user base is still small in comparison to XGBoost's.

Model Justification

- Neural Networks
- Long Short Term Memory (LSTM)
- Bayesian Ridge -> Baseline Model
- LightGBM
- XGBoost
- CatBoost

Neural networks performs the best out of all the models implemented.

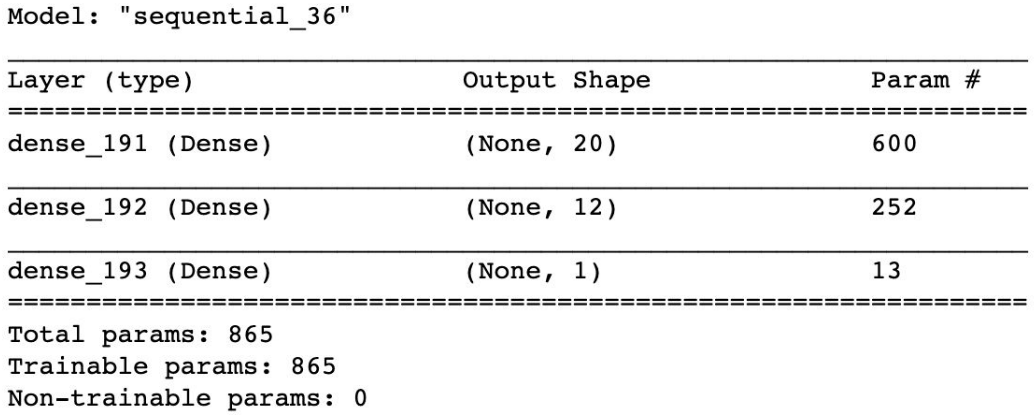
Bayesian Ridge Regression

- Linear regression model using probability distributions instead of point estimates
- Rather than find single best value of the model parameters, determine the posterior distribution for the model parameters and for model prediction, mean can be used as estimates for parameters and response value
- Priors on the coefficients can be incorporated into the model
    - Prior for the coefficients is given by a Gaussian distribution
    - Prior for regularization parameters is given by non-informative gamma distribution
- Validation RMSE: 0.5125
- Test RMSE from Kaggle: 1.23994
- Better model result might be possible if more domain knowledge

Neural Network Model Summary

Base model:

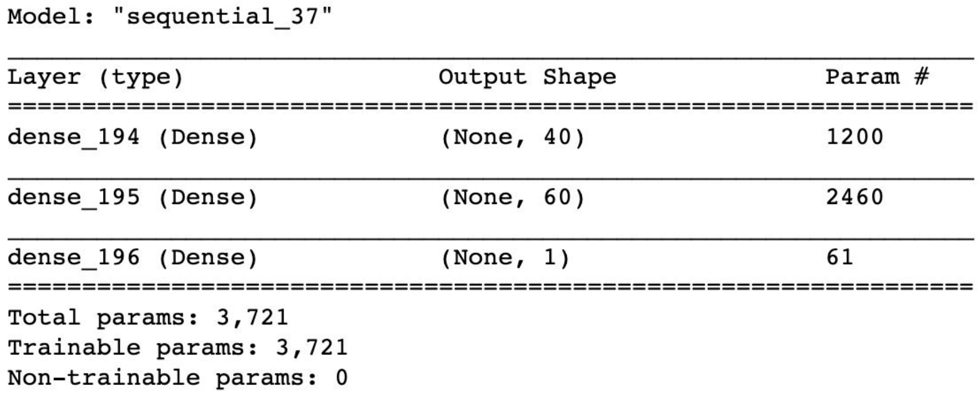
Best rmse on validation data: 1.08



```
Model: "sequential_36"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_191 (Dense)            (None, 20)                600
_________________________________________________________________
dense_192 (Dense)            (None, 12)                252
_________________________________________________________________
dense_193 (Dense)            (None, 1)                 13
=================================================================
Total params: 865
Trainable params: 865
Non-trainable params: 0
```

Wider model:

Best rmse on validation data: 1.07

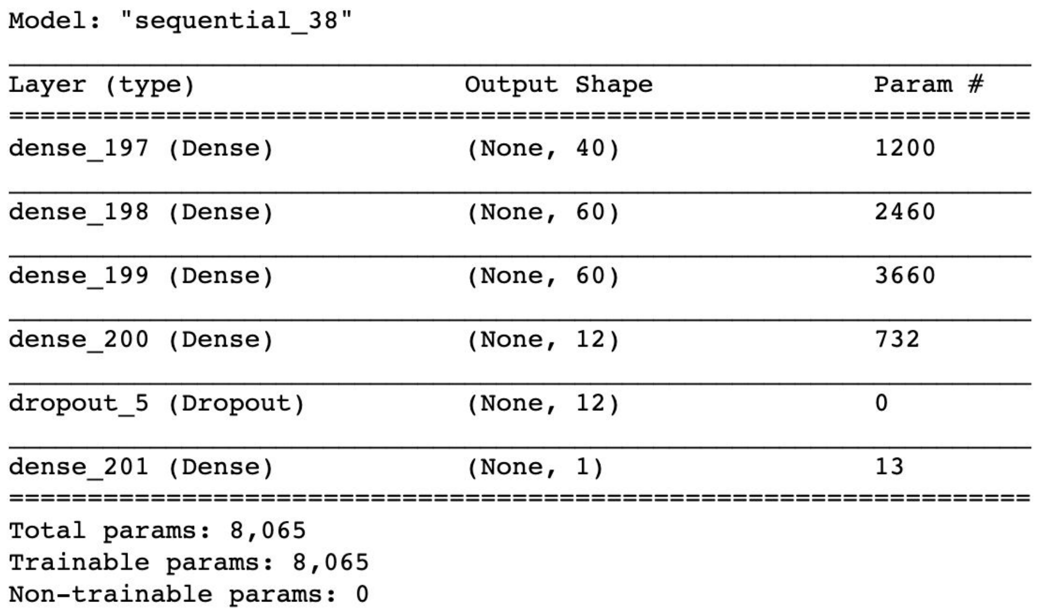


```
Model: "sequential_37"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_194 (Dense)            (None, 40)                1200
_________________________________________________________________
dense_195 (Dense)            (None, 60)                2460
_________________________________________________________________
dense_196 (Dense)            (None, 1)                 61
=================================================================
Total params: 3,721
Trainable params: 3,721
Non-trainable params: 0
```

Neural Network Model Summary

Deeper model:

Best rmse on validation data:1.16

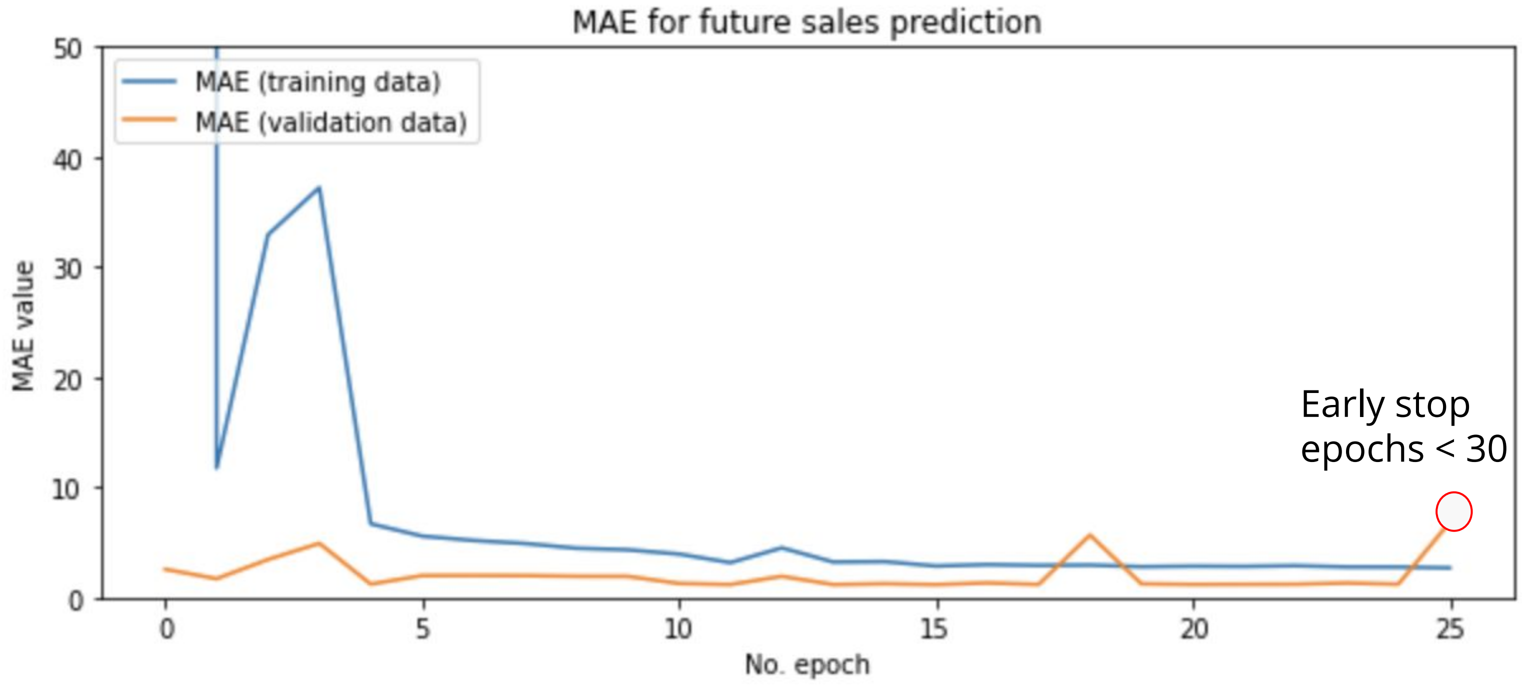


```
Model: "sequential_38"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_197 (Dense)            (None, 40)                1200
_________________________________________________________________
dense_198 (Dense)            (None, 60)                2460
_________________________________________________________________
dense_199 (Dense)            (None, 60)                3660
_________________________________________________________________
dense_200 (Dense)            (None, 12)                732
_________________________________________________________________
dropout_5 (Dropout)          (None, 12)                0
_________________________________________________________________
dense_201 (Dense)            (None, 1)                 13
=================================================================
Total params: 8,065
Trainable params: 8,065
Non-trainable params: 0
```

Neural Network

- Feature selection
    - For the new generated feature, add one a time, and test it in model.
    - Feature selection improved best validation rmse from 1.31 to 1.07
- Shuffle the samples or Not
    - Use first 31 months to train, use 32th month to valid, results better prediction.
- PCA
    - We tried PCA with total explained variance ratio = 99%
    - Feature number reduced from 18 to 16
    - We see rmse on validation data reduced to 0.99. (Didn't improve Kaggle score)
- Wider or deeper model
    - We built a base neural network model, first make it wider, then make it deeper
    - Compare the validation rmse among those models.
    - For this case, a wider network improved rmse on both validation data and Kaggle test.
    - The deeper and wider network doesn't work better. Model is under fitting.
- Choose optimal number of epochs to train
    - Run large number of epochs sometime drive to overfitting, especially in this case.
    - Use **keras.callbacks** to recall the best weights

Neural Network Model Evaluation



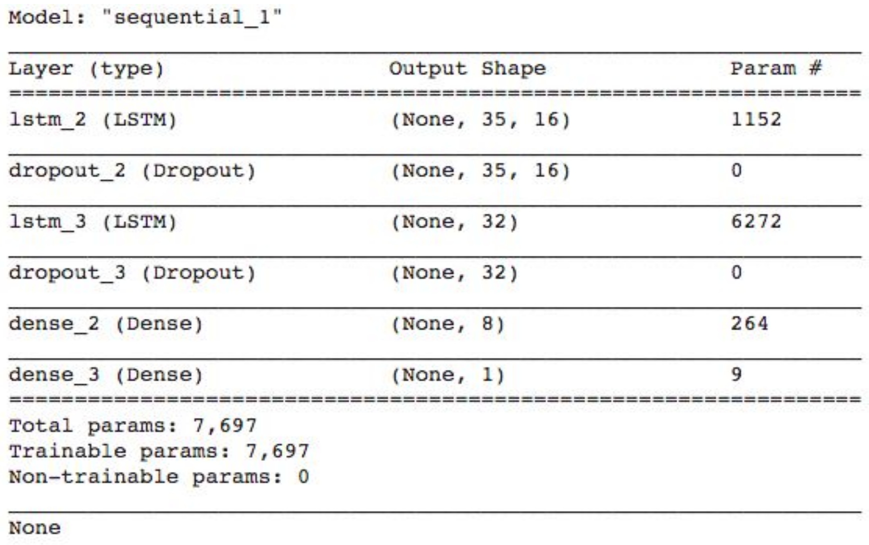
Early stop
epochs < 30

LSTM - Long Short Term Memory Network

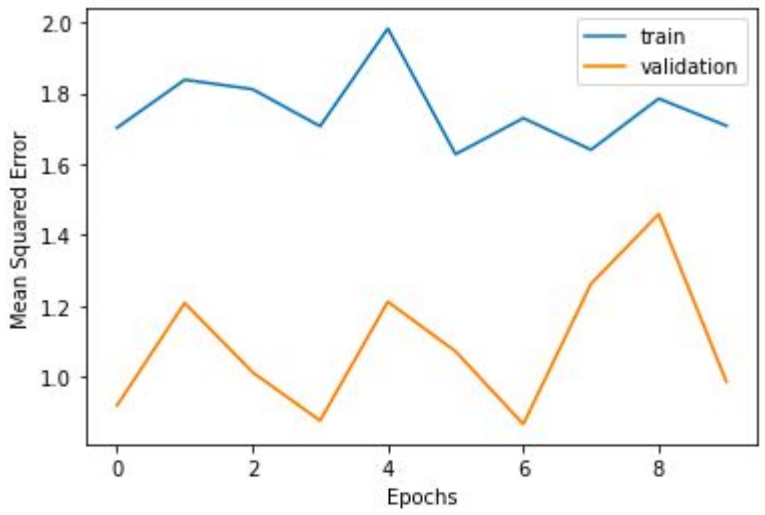
- LSTM models are among the most powerful models in ML for many time series forecasting and prediction problems.
- Ability to handle exploding/vanishing gradient problem.
- They can selectively remember patterns for a long duration of time.
    - This property gives it a huge advantage over other conventional approaches.
- Requires lot of training for the model to learn all the sequential data.
- Validation RMSE : 0.8635
- Test RMSE from Kaggle : 1.2221

LSTM Model architecture

- 2 LSTM layers , 2 Dense layers , 2 Dropout layers in between LSTM layers to avoid overfitting with 7697 total parameters.Early stopping and training for about 100 epochs.
- Imputed null values , Scaling(minmax) and adding complexity to the model didn't improve score.

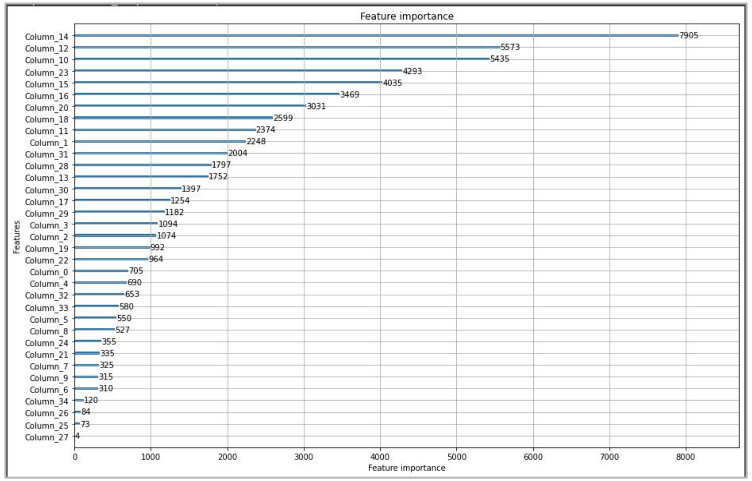


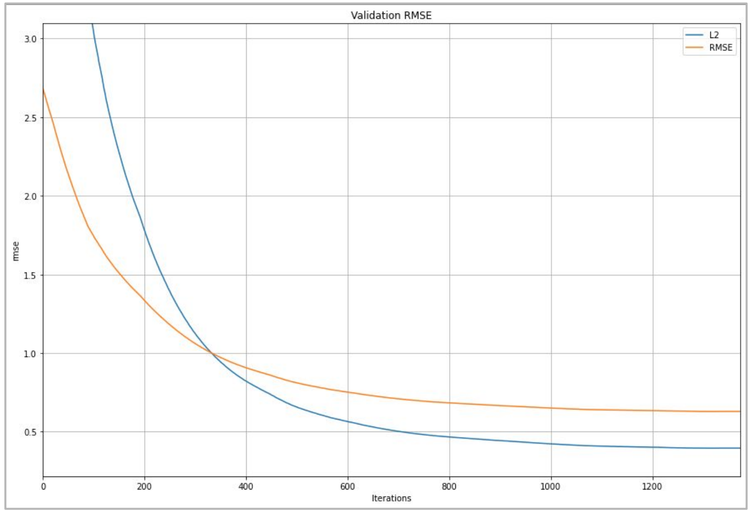
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 35, 16)            1152      
_________________________________________________________________
dropout_2 (Dropout)          (None, 35, 16)            0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 32)                6272      
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 264       
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9         
=================================================================
Total params: 7,697
Trainable params: 7,697
Non-trainable params: 0
_________________________________________________________________
None
```



LightGBM

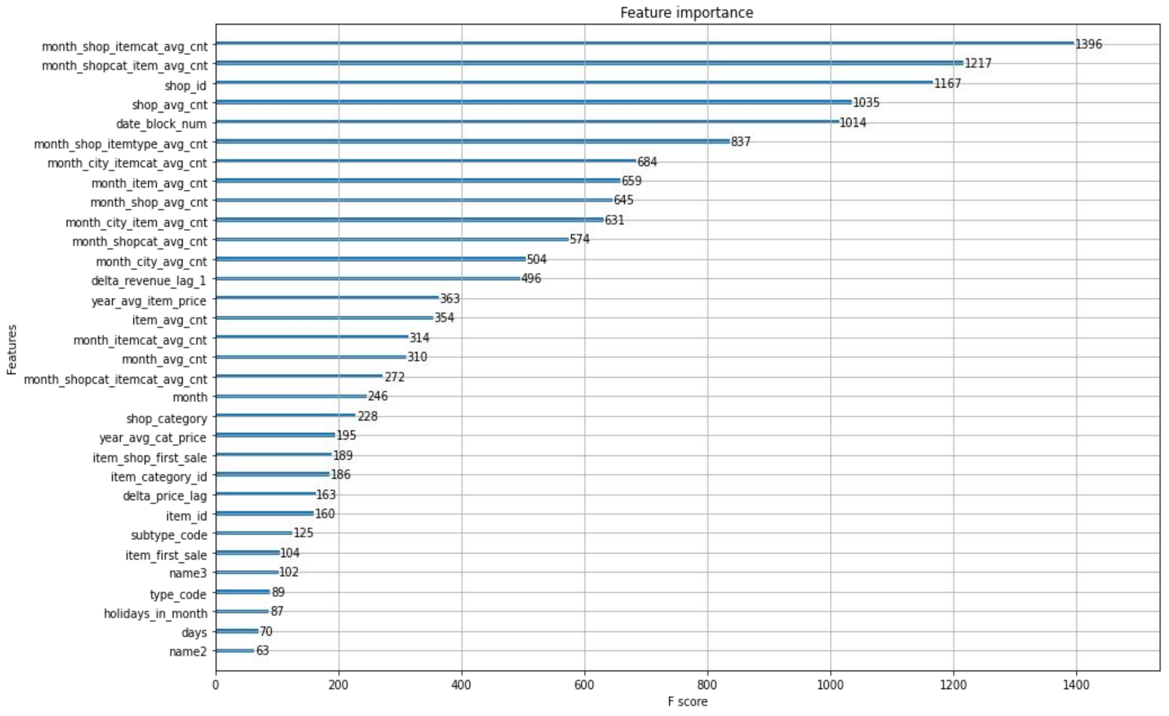
- LightGBM extends the gradient boosting algorithm by adding a type of automatic feature selection as well as focusing on boosting examples with larger gradients.
- This can result in a dramatic speedup of training and improved predictive performance.
- LightGBM performed pretty good on this dataset but its not the best and this can be rectified with further pruning or a different type of feature engineering of our lag features.





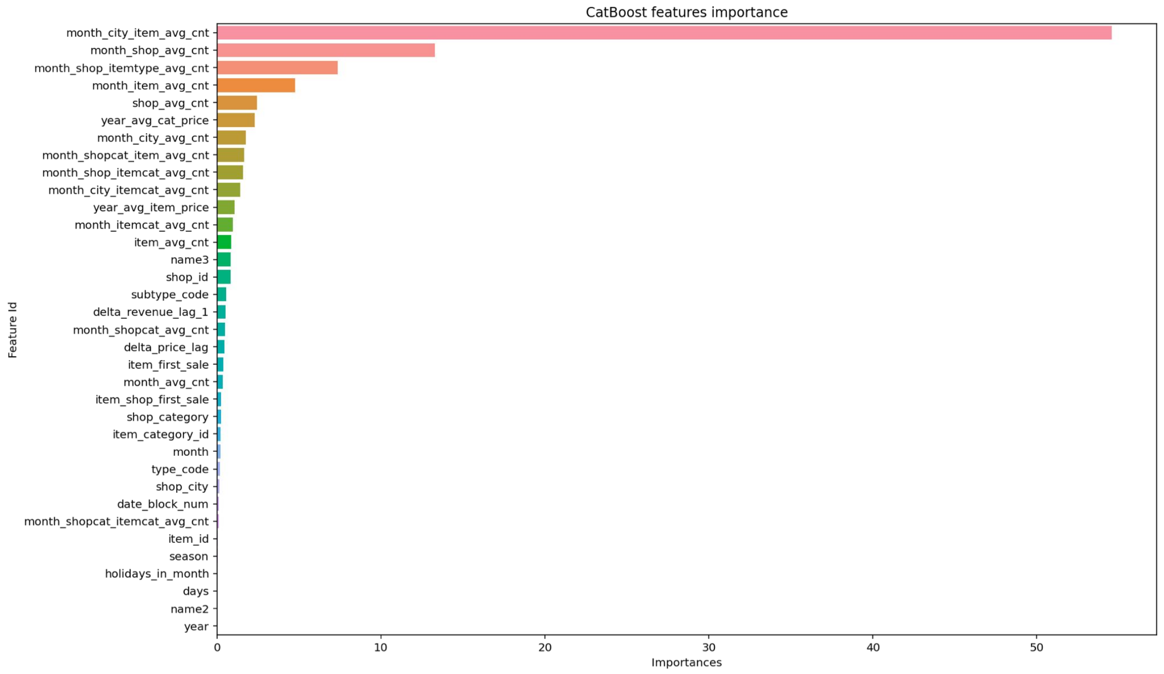
XGBoost

- XGBoost is designed for classification and regression on tabular datasets, although it can be used for time series forecasting.
- Provides a highly efficient implementation of the stochastic gradient boosting algorithm and access to a suite of model hyperparameters designed to provide control over the model training process.



CatBoost Regressor

- Boosting algorithm developed by Yandex.
- Improved model performance by highlighting categorical features
- Validation RMSE: 0.70
- Test RMSE from Kaggle: 1.25330



Result Analysis

- Our LSTM and neural network models worked the best.
  - Possible correlation between the structure of the dataset and engineered features and the nature of the models?
- Most top implementations used XGBoost, but not always guaranteed to be the best solution.
- Possible to improve leaderboard standing with model tuning and elimination of poorly correlated features.
  - There may exist additional features that can improve our results.

Result Analysis

Model Name	Kaggle RMSE	Placement (total teams 11271)
XGBoost	1.24630	8726
LightGBM	1.24900	8737
Naive Bayesian	1.23994	8695
CatBoost	1.2533	8866
LSTM	1.22213	7731
Neural Network	1.20327	7523

Challenges

- Computational resources
- Language barrier
- Domain Knowledge
- Feature engineering techniques
- Multitude of models to choose from
    - Pick the right tool for the job!
        - ARIMA as a popular tool for time series analysis (used as a utility tool)
        - Variety of different models to evaluate their final validation results
            - We broached neural networks, ensemble (boosting algorithms) since they typically perform better in general
            - Used bayesian as a baseline due to its similarity to linear regression

What we learned!

- Using Jupyter vs. Google Colab.
    - Choosing the right one is important for your use-case.
- How to compete in a Kaggle competition.
- Got more experience with common Data Science libraries
    - Pandas
    - Numpy
    - Sklearn
- PCA is not always useful.

Thank you!