ML Project
Future Sales Prediction

Zack Chen, Dec 1st, 2020

Project Overview

- Time-series dataset consisting of 33 months' daily sales data of a Russian software firms - 1C Company.

- Goal: Predict total sales for every product and store in the next month(34th).

Project Overview - Train Dataset

- Store transactions from Jan. 2013 - Oct. 2015
  - Total 33 months
- Each transaction contains:
  - Shop name/ID
  - Item name/ID
  - Category name/ID
  - Item price
  - Item count
- 2.9 million records
- All text in Russian

	date	date_block_num	shop_id	item_id	item_price	tem_cnt_day	tem_name	item_category_id	item_category_name
0	2013-01-02	0	59	22154	999.00	1.0	ЯВЛЕНИЕ 2012 (BD)	37	Кино - Blu-Ray
1	2013-01-03	0	25	2552	899.00	1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил
2	2013-01-05	0	25	2552	899.00	-1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил
3	2013-01-06	0	25	2554	1709.05	1.0	DEEP PURPLE Who Do You Think We Are LP	58	Музыка - Винил
4	2013-01-15	0	25	2555	1099.00	1.0	DEEP PURPLE 30 Very Best Of 2CD (Фирм.)	56	Музыка - CD фирменного производства
...	...	...	...	...	...	...	...	...	...
2935844	2015-10-10	33	25	7409	299.00	1.0	V/A Nu Jazz Selection (digipack)	55	Музыка - CD локального производства
2935845	2015-10-09	33	25	7460	299.00	1.0	V/A The Golden Jazz Collection 1 2CD	55	Музыка - CD локального производства
2935846	2015-10-14	33	25	7459	349.00	1.0	V/A The Best Of The 3 Tenors	55	Музыка - CD локального производства
2935847	2015-10-22	33	25	7440	299.00	1.0	V/A Relax Collection Planet MP3 (mp3-CD) (jewel)	57	Музыка - MP3
2935848	2015-10-03	33	25	7460	299.00	1.0	V/A The Golden Jazz Collection 1 2CD	55	Музыка - CD локального производства

2935849 rows × 12 columns

Project Overview - Test Dataset

- Predict the next month's (34th) sales of every item_id of every shop_id
- Total unique pairs of shop_id and item_id:
    - 42 * 5100 = 214200
    - 214200 rows
- Only given shop_id and item_id

ID	shop_id	item_id
0	5	5037
1	5	5320
2	5	5233
3	5	5232
4	5	5268
...	...	...
214195	45	18454
214196	45	16188
214197	45	15757
214198	45	19648
214199	45	969

214200 rows × 2 columns

EDA

Items Table

- 22170 unique items
- To be joined with the main train table (sale transactions table)

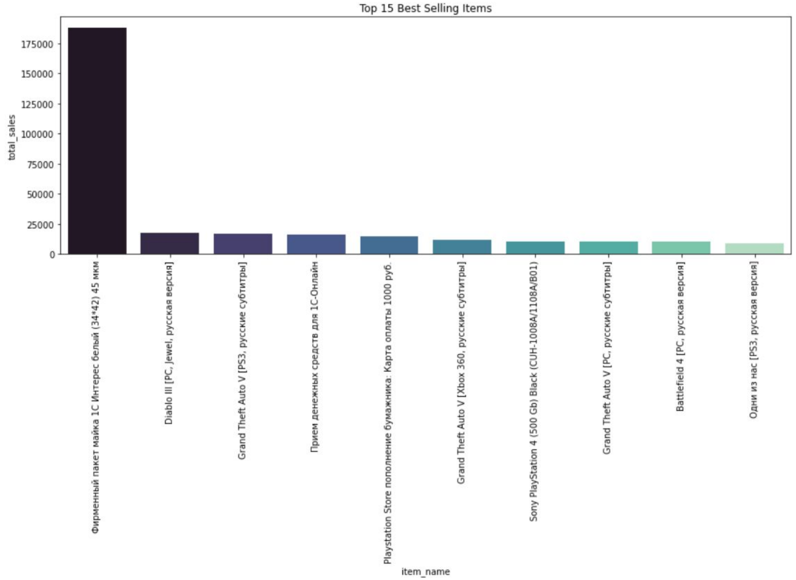
	item_name	item_id	item_category_id
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40
1	!ABBYY FineReader 12 Professional Edition Full...	1	76
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40
4	***КОРОБКА (СТЕКЛО) D	4	40
...	...	...	...
22165	Ядерный титбит 2 [PC, Цифровая версия]	22165	31
22166	Язык запросов 1С:Предприятия [Цифровая версия]	22166	54
22167	Язык запросов 1С:Предприятия 8 (+CD). Хрустале...	22167	49
22168	Яйцо для Little Inu	22168	62
22169	Яйцо дракона (Игра престолов)	22169	69

22170 rows × 3 columns

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	item_name	item_category_id
0	02.01.2013	0	59	22154	999.00	1.0	ЯВЛЕНИЕ 2012 (BD)	37
1	03.01.2013	0	25	2552	899.00	1.0	DEEP PURPLE The House Of Blue Light LP	58
2	05.01.2013	0	25	2552	899.00	-1.0	DEEP PURPLE The House Of Blue Light LP	58
3	06.01.2013	0	25	2554	1709.05	1.0	DEEP PURPLE Who Do You Think We Are LP	58
4	15.01.2013	0	25	2555	1099.00	1.0	DEEP PURPLE 30 Very Best Of 2CD (Фирм.)	56

Top 10 Best Selling Items

	item_name	item_cnt_day
20602	Фирменный пакет майка 1С Интерес белый (34*42)...	187642.0
2749	Diablo III [PC, Jewel, русская версия]	17245.0
3654	Grand Theft Auto V [PS3, русские субтитры]	16642.0
17418	Прием денежных средств для 1С-Онлайн	15830.0
5717	Playstation Store пополнение бумажника: Карта ...	14515.0
3656	Grand Theft Auto V [Xbox 360, русские субтитры]	11688.0
6543	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	10289.0
3653	Grand Theft Auto V [PC, русские субтитры]	10099.0
1814	Battlefield 4 [PC, русская версия]	10032.0
16493	Одни из нас [PS3, русская версия]	9227.0



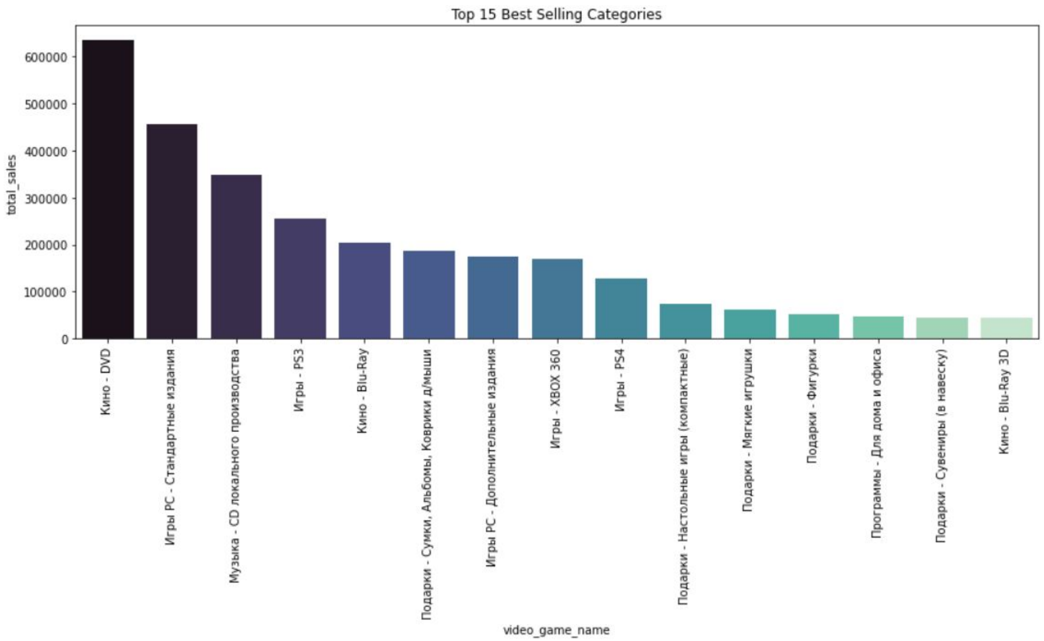
Categories Table

- 83 unique categories (rows)
- Create a new "isVideoGame" column
- To be joined with the main train table (sale transactions table)

	item_category_name	item_category_id	isVideoGame
0	PC - Гарнитуры/Наушники	0	False
1	Аксессуары - PS2	1	True
2	Аксессуары - PS3	2	True
3	Аксессуары - PS4	3	True
4	Аксессуары - PSP	4	True
5	Аксессуары - PSVita	5	True

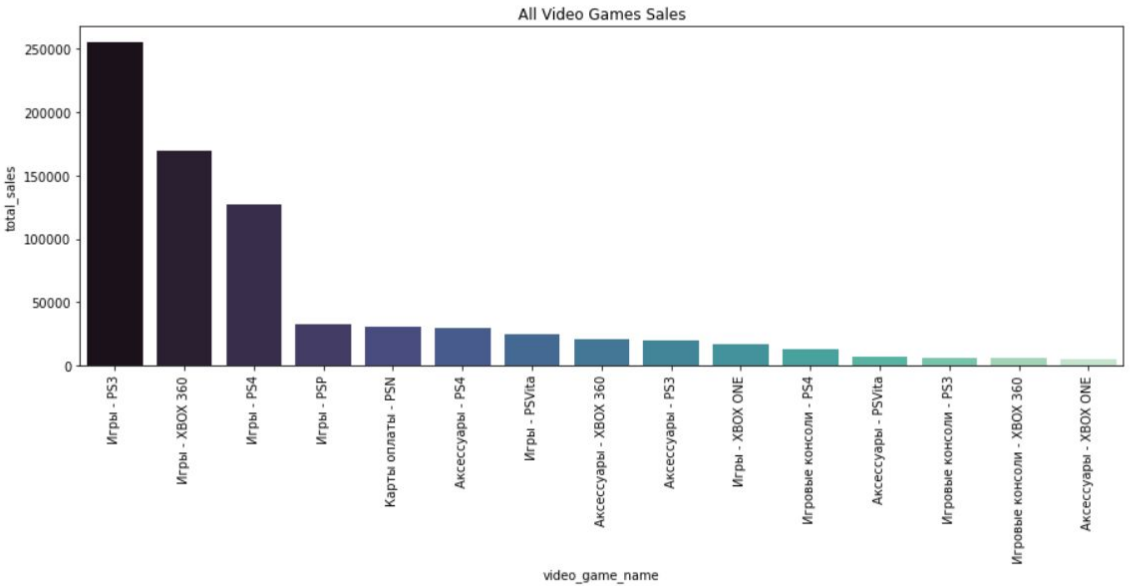
	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	item_name	item_category_id	item_category_name	isVideoGame
0	02.01.2013	0	59	22154	999.00	1.0	ЯВЛЕНИЕ 2012 (BD)	37	Кино - Blu-Ray	False
1	03.01.2013	0	25	2552	899.00	1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил	False
2	05.01.2013	0	25	2552	899.00	-1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил	False
3	06.01.2013	0	25	2554	1709.05	1.0	DEEP PURPLE Who Do You Think We Are LP	58	Музыка - Винил	False
4	15.01.2013	0	25	2555	1099.00	1.0	DEEP PURPLE 30 Very Best Of 2CD (Фирм.)	56	Музыка - CD фирменного производства	False

Categories Table



	item_category_name	total_sales
40	Кино - DVD	634171.0
30	Игры PC - Стандартные издания	456540.0
55	Музыка - CD локального производства	348591.0
19	Игры - PS3	254887.0
37	Кино - Blu-Ray	203284.0
71	Подарки - Сумки, Альбомы, Коврики д/мыши	187998.0
28	Игры PC - Дополнительные издания	174954.0
23	Игры - XBOX 360	169944.0
20	Игры - PS4	127319.0
65	Подарки - Настольные игры (компактные)	73077.0
63	Подарки - Мягкие игрушки	60856.0
72	Подарки - Фигурки	51621.0
75	Программы - Для дома и офиса	48224.0
70	Подарки - Сувениры (в навеску)	45067.0
38	Кино - Blu-Ray 3D	45032.0

Best Selling Video Games



	video_game_name	total_sales
15	Игры - PS3	254887.0
19	Игры - XBOX 360	169944.0
16	Игры - PS4	127319.0
17	Игры - PSP	33066.0
21	Карты оплаты - PSN	31244.0
2	Аксессуары - PS4	29807.0
18	Игры - PSVita	25123.0
5	Аксессуары - XBOX 360	20472.0
1	Аксессуары - PS3	19597.0
20	Игры - XBOX ONE	16886.0
9	Игровые консоли - PS4	13230.0
4	Аксессуары - PSVita	7413.0
8	Игровые консоли - PS3	6403.0
12	Игровые консоли - XBOX 360	5980.0
6	Аксессуары - XBOX ONE	5358.0

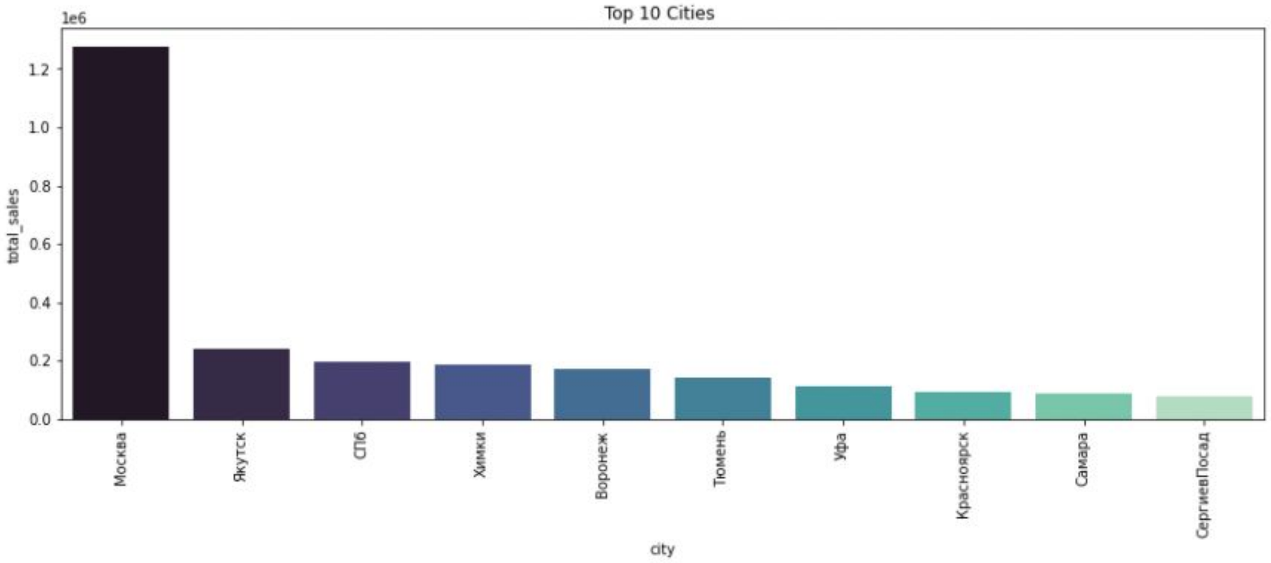
Shops Table

- 59 unique shops
- First terms in the shop_names are cities
- To be joined with the main train table (sale transactions table)

	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
5	Вологда ТРЦ "Мармелад"	5

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	item_name	item_category_id	item_category_name	isVideoGame	shop_name	city
0	02.01.2013	0	59	22154	999.00	1.0	ЯВЛЕНИЕ 2012 (BD)	37	Кино - Blu-Ray	False	Ярославль ТЦ "Альтаир"	Ярославль
1	03.01.2013	0	25	2552	899.00	1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил	False	Москва ТРК "Атриум"	Москва
2	05.01.2013	0	25	2552	899.00	-1.0	DEEP PURPLE The House Of Blue Light LP	58	Музыка - Винил	False	Москва ТРК "Атриум"	Москва
3	06.01.2013	0	25	2554	1709.05	1.0	DEEP PURPLE Who Do You Think We Are LP	58	Музыка - Винил	False	Москва ТРК "Атриум"	Москва

Top 15 Cities



	city	total_sales
13	Москва	1276376.0
29	Якутск	240857.0
19	СПб	195542.0
26	Химки	185790.0
4	Воронеж	171142.0
24	Тюмень	142095.0
25	Уфа	111401.0
11	Красноярск	91324.0
20	Самара	86833.0
21	СергиевПосад	78990.0

FEATURE ENGINEERING

Sales Table

- 2935849 sales records (2.9M)
- Consider 'date_block_num' as month,
  - Jan 2013 -> 0
  - Oct 2015 -> 33

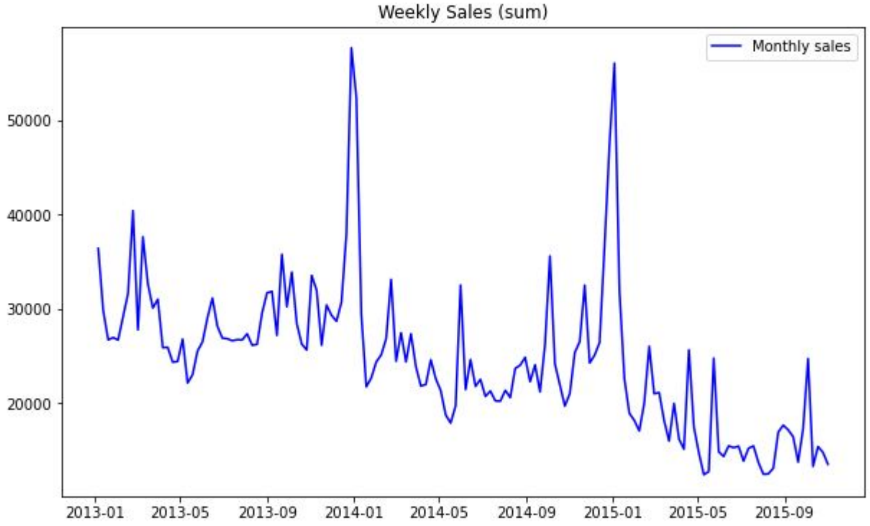
	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	Year	Month
0	2013-01-02	0	59	22154	999.00	1.0	2013	1
1	2013-01-03	0	25	2552	899.00	1.0	2013	1
2	2013-01-05	0	25	2552	899.00	-1.0	2013	1
3	2013-01-06	0	25	2554	1709.05	1.0	2013	1
4	2013-01-15	0	25	2555	1099.00	1.0	2013	1
5	2013-01-10	0	25	2564	349.00	1.0	2013	1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Year	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013	2014	2014	2014	2014	2014
Month	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5
item_cnt_day	115690	108613	121347	94109	91759	100403	100548	104772	96137	94202	96736	143246	99349	89830	92733	77906	78529

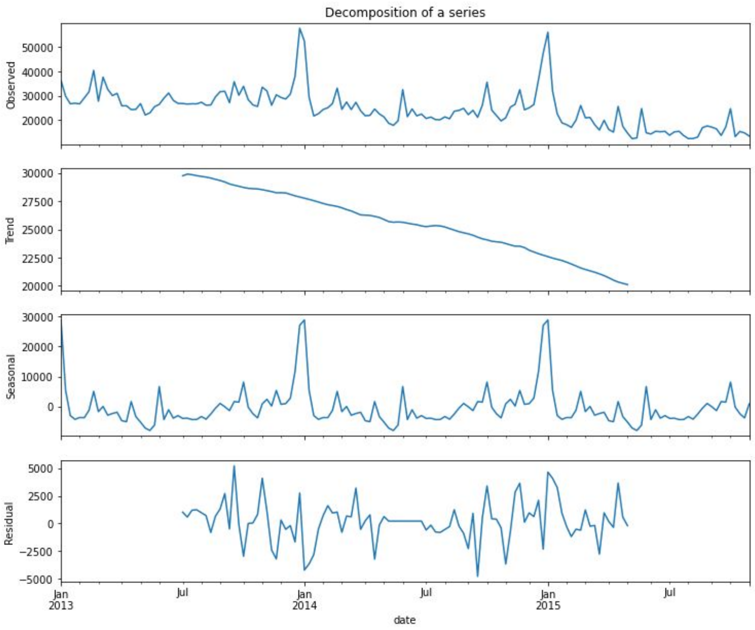
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
2014	2014	2014	2014	2014	2014	2014	2015	2015	2015	2015	2015	2015	2015	2015	2015	2015
6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
82408	78760	86614	73157	79361	86428	130786	88522	71808	69977	56274	54548	54617	55549	57029	50588	53514

Weekly Sales

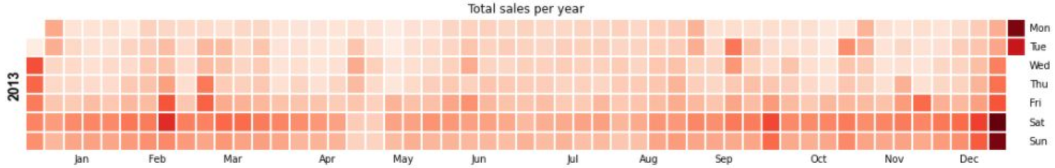
```
train[["date", "item_cnt_day"]].set_index("date").resample("W").sum()
```

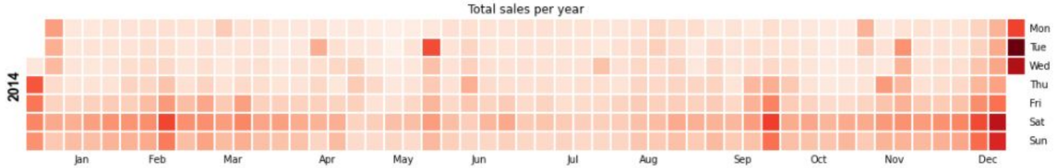


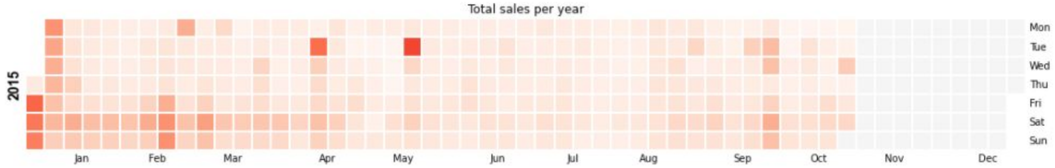
Sales Decomposition



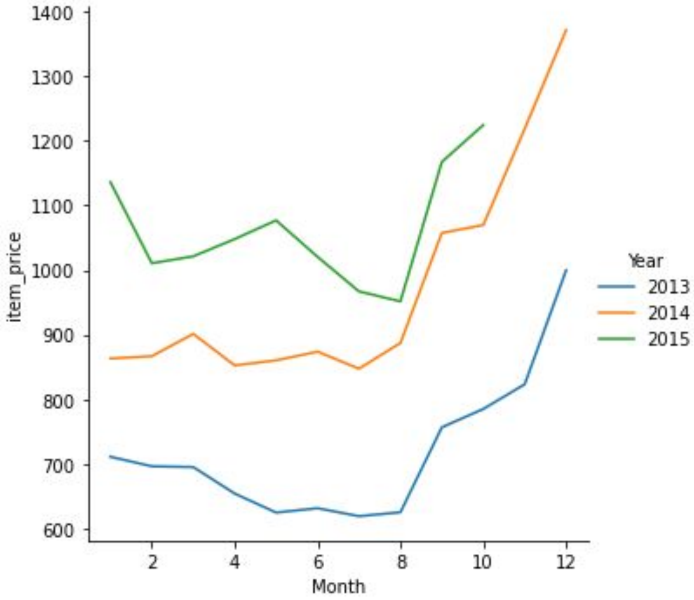
Sales Calmap





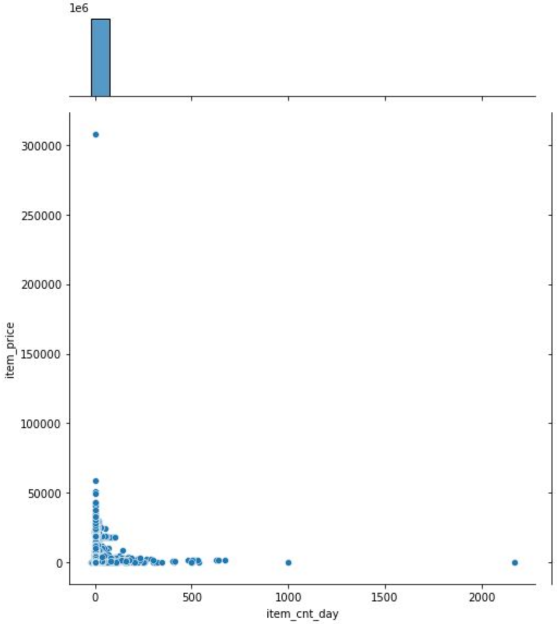


Item Price



1400
1300
1200
1100
1000
900
800
700
600
item_price
2
4
6
8
10
12
Month
Year
2013
2014
2015

The Label - ‘item_cnt_day’




```
count    2.935849e+06
mean     1.242641e+00
std      2.618834e+00
min     -2.200000e+01
25%      1.000000e+00
50%      1.000000e+00
75%      1.000000e+00
max      2.169000e+03
Name: item_cnt_day, dtype: float64
```

FEATURE ENGINEERING

Test Data Set

test contains 363 new item_id that train data doesn't have

```
test.item_id.nunique()-len(set(test.item_id).intersection(set(train.item_id)))
```

```
363
```

test does not contain any new stores

```
test.shop_id.nunique()-len(set(test.shop_id).intersection(set(train.shop_id)))
```

```
0
```

Test table can be seen as the 34th month

```
test['date_block_num']=34
```

ID	shop_id	item_id	date_block_num
0	5	5037	34
1	5	5320	34
2	5	5233	34
3	5	5232	34
4	5	5268	34
...	...	...	...
214195	45	18454	34
214196	45	16188	34
214197	45	15757	34
214198	45	19648	34
214199	45	969	34

214200 rows × 3 columns

Matrix Table - The New Training Set

- Created a new table that contains all possible combinations of **['date_block_num','shop_id','item_id']** to cover missing records.
- Group by ['date_block_num','shop_id','item_id'], get **sum of 'item_cnt_day'** since we only care about item sales per month.
- Left join with original **Sales** table.
- Concatenate with **Test** table
- Fillna(0)

	date_block_num	shop_id	item_id	item_cnt_block	city_code	item_category_id	type_code	subtype_code
0	0	2	19	0.00	0	40	11	4
1	0	2	27	1.00	0	19	5	10
2	0	2	28	0.00	0	30	8	55
3	0	2	29	0.00	0	23	5	16
4	0	2	32	0.00	0	40	11	4
...	...	...	...	...	...	...	...	...
11127999	34	45	18454	0.00	20	55	13	2
11128000	34	45	16188	0.00	20	64	14	42
11128001	34	45	15757	0.00	20	55	13	2
11128002	34	45	19648	0.00	20	40	11	4
11128003	34	45	969	0.00	20	37	11	1

11128004 rows × 8 columns

Original train table: 2935849 rows × 8 columns

Time Series Dataset - Lag Features

- **Lag**: The Lag feature is simply using a previous target value as a feature to predict the current one.

Months to shift Target feature

```
def lag_func(df, i, col):

    temp = df[['date_block_num','shop_id','item_id',col]].copy()
    temp['date_block_num'] += i
    temp.columns = ['date_block_num','shop_id','item_id', col+'_lag_'+str(i)]
    df = pd.merge(df, temp, on = ['date_block_num','shop_id','item_id'], how = 'left')

    return df
```

Lag Function

- Why not .shift(1) ?

```
lag_func(matrix, 1, 'item_cnt_block')
```

	date_block_num	shop_id	item_id	item_cnt_block	city_code	item_category_id	type_code	subtype_code	item_cnt_lag_1
4836058	13	2	32	0.00	0	40	11	4	1.00
5164225	14	2	32	1.00	0	40	11	4	0.00
5507039	15	2	32	0.00	0	40	11	4	1.00
5838965	16	2	32	0.00	0	40	11	4	0.00
6166530	17	2	32	0.00	0	40	11	4	0.00
6493901	18	2	32	1.00	0	40	11	4	0.00
6824456	19	2	32	0.00	0	40	11	4	1.00
7148969	20	2	32	2.00	0	40	11	4	0.00
7452513	21	2	32	2.00	0	40	11	4	2.00
7781882	22	2	32	0.00	0	40	11	4	2.00

Lag Features - Test Set

- How to avoid Data Leakage?

	date_block_num	shop_id	item_id	item_cnt_block	city_code	item_category_id	type_code	subtype_code	item_cnt_block_lag_1	item_cnt_block_lag_
0	0	2	19	0.00	0	40	11	4	nan	na
1	0	2	27	1.00	0	19	5	10	nan	na
2	0	2	28	0.00	0	30	8	55	nan	na
3	0	2	29	0.00	0	23	5	16	nan	na
4	0	2	32	0.00	0	40	11	4	nan	na
...	...	...	...	...	...	...	...	...	...	
11127999	34	45	18454	0.00	20	55	13	2	1.00	0.(
11128000	34	45	16188	0.00	20	64	14	42	0.00	0.(
11128001	34	45	15757	0.00	20	55	13	2	0.00	0.(
11128002	34	45	19648	0.00	20	40	11	4	0.00	0.(
11128003	34	45	969	0.00	20	37	11	1	0.00	0.(

Lag Features - More Features

Month

Label, sales count

```
matrix['mean_block_item_cnt']=matrix.groupby(['date_block_num', 'item_id'])['item_cnt_block'].transform('mean')
for i in [1,2,3,6,12]:
    matrix = lag_func(matrix, i, 'mean_block_item_cnt')
matrix.drop(['mean_block_item_cnt'], axis=1, inplace=True)
```

Lag Features - More Features

```
matrix['mean_city_cnt']=matrix.groupby(['date_block_num', 'city_code'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_city_cnt')
matrix.drop(['mean_city_cnt'], axis=1, inplace=True)

matrix['mean_cat_cnt']=matrix.groupby(['date_block_num', 'item_category_id'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_cat_cnt')
matrix.drop(['mean_cat_cnt'], axis=1, inplace=True)

matrix['mean_type_cnt']=matrix.groupby(['date_block_num', 'type_code'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_type_cnt')
matrix.drop(['mean_type_cnt'], axis=1, inplace=True)

matrix['mean_subtype_cnt']=matrix.groupby(['date_block_num', 'subtype_code'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_subtype_cnt')
matrix.drop(['mean_subtype_cnt'], axis=1, inplace=True)

matrix['mean_shop_cat_cnt']=matrix.groupby(['date_block_num', 'shop_id','item_category_id'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_shop_cat_cnt')
matrix.drop(['mean_shop_cat_cnt'], axis=1, inplace=True)

matrix['mean_shop_type_cnt']=matrix.groupby(['date_block_num', 'shop_id','type_code'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_shop_type_cnt')
matrix.drop(['mean_shop_type_cnt'], axis=1, inplace=True)

matrix['mean_shop_subtype_cnt']=matrix.groupby(['date_block_num', 'shop_id','subtype_code'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_shop_subtype_cnt')
matrix.drop(['mean_shop_subtype_cnt'], axis=1, inplace=True)

matrix['mean_city_item_cnt']=matrix.groupby(['date_block_num', 'city_code','item_id'])['item_cnt_block'].transform('mean')
matrix = lag_func(matrix, 1, 'mean_city_item_cnt')
matrix.drop(['mean_city_item_cnt'], axis=1, inplace=True)
```

Price Trend

- Calculate each item’s mean price
- Calculate each item’s mean price of every month, lag for 6 months
- Calculate price trend

mean_price_delta_lag_1	mean_price_delta_lag_2	mean_price_delta_lag_3	mean_price_delta_lag_4	mean_price_delta_lag_5	mean_price_delta_lag_6
0.01	0.01	0.01	0.01	-0.21	0.01
nan	-0.03	-0.03	-0.15	-0.03	-0.03
nan	0.02	0.02	0.02	0.02	0.02
-0.10	-0.10	nan	nan	nan	nan
-0.10	-0.10	nan	nan	nan	nan

Item First Sale

- For each item in each shop, find the month of first sale and months the item has been for sale

```
matrix['shop_item_first_sale']=matrix.groupby(['shop_id','item_id'])['date_block_num'].transform('min')
matrix['shop_item_since_first_sale']=matrix['date_block_num']-matrix['shop_item_first_sale']
```

- For each item, find the month of first global sale and months since then

```
matrix['item_first_sale']=matrix.groupby(['item_id'])['date_block_num'].transform('min')
matrix['item_since_first_sale']=matrix['date_block_num']-matrix['item_first_sale']
```

Final Preparation

- Fill NA values with 0
- Run memory reduction

```
 #   Column                      Dtype
---  ------                      -----
 0   date_block_num              int64
 1   shop_id                     int64
 2   item_id                     int64
 3   item_cnt_block              float64
 4   city_code                   int64
 5   item_category_id            int64
 6   type_code                   int64
 7   subtype_code                int64
 8   item_cnt_block_lag_1        float64
 9   item_cnt_block_lag_2        float64
 10  item_cnt_block_lag_3        float64
 11  mean_block_cnt_lag_1        float64
 12  mean_block_item_cnt_lag_1   float64
 13  mean_block_shop_cnt_lag_1   float64
 14  mean_city_cnt_lag_1         float64
 15  mean_cat_cnt_lag_1          float64
 16  mean_type_cnt_lag_1         float64
 17  mean_subtype_cnt_lag_1      float64
 18  mean_shop_cat_cnt_lag_1     float64
 19  mean_shop_type_cnt_lag_1    float64
 20  mean_shop_subtype_cnt_lag_1 float64
 21  mean_city_item_cnt_lag_1    float64
 22  recent_price_delta          float64
 23  shop_revenue_delta_lag_1    float64
 24  shop_item_since_first_sale  int64
 25  item_since_first_sale       int64
 26  month                       int64
dtypes: float64(17), int64(10)
memory usage: 2.2 GB
```

```
 #   Column                      Dtype
---  ------                      -----
 0   date_block_num              int8
 1   shop_id                     int8
 2   item_id                     int16
 3   item_cnt_block              float16
 4   city_code                   int8
 5   item_category_id            int8
 6   type_code                   int8
 7   subtype_code                int8
 8   item_cnt_block_lag_1        float16
 9   item_cnt_block_lag_2        float16
 10  item_cnt_block_lag_3        float16
 11  mean_block_cnt_lag_1        float16
 12  mean_block_item_cnt_lag_1   float16
 13  mean_block_shop_cnt_lag_1   float16
 14  mean_city_cnt_lag_1         float16
 15  mean_cat_cnt_lag_1          float16
 16  mean_type_cnt_lag_1         float16
 17  mean_subtype_cnt_lag_1      float16
 18  mean_shop_cat_cnt_lag_1     float16
 19  mean_shop_type_cnt_lag_1    float16
 20  mean_shop_subtype_cnt_lag_1 float16
 21  mean_city_item_cnt_lag_1    float16
 22  recent_price_delta          float16
 23  shop_revenue_delta_lag_1    float16
 24  shop_item_since_first_sale  int8
 25  item_since_first_sale       int8
 26  month                       int8
dtypes: float16(17), int16(1), int8(9)
memory usage: 477.6 MB
```

Modelling

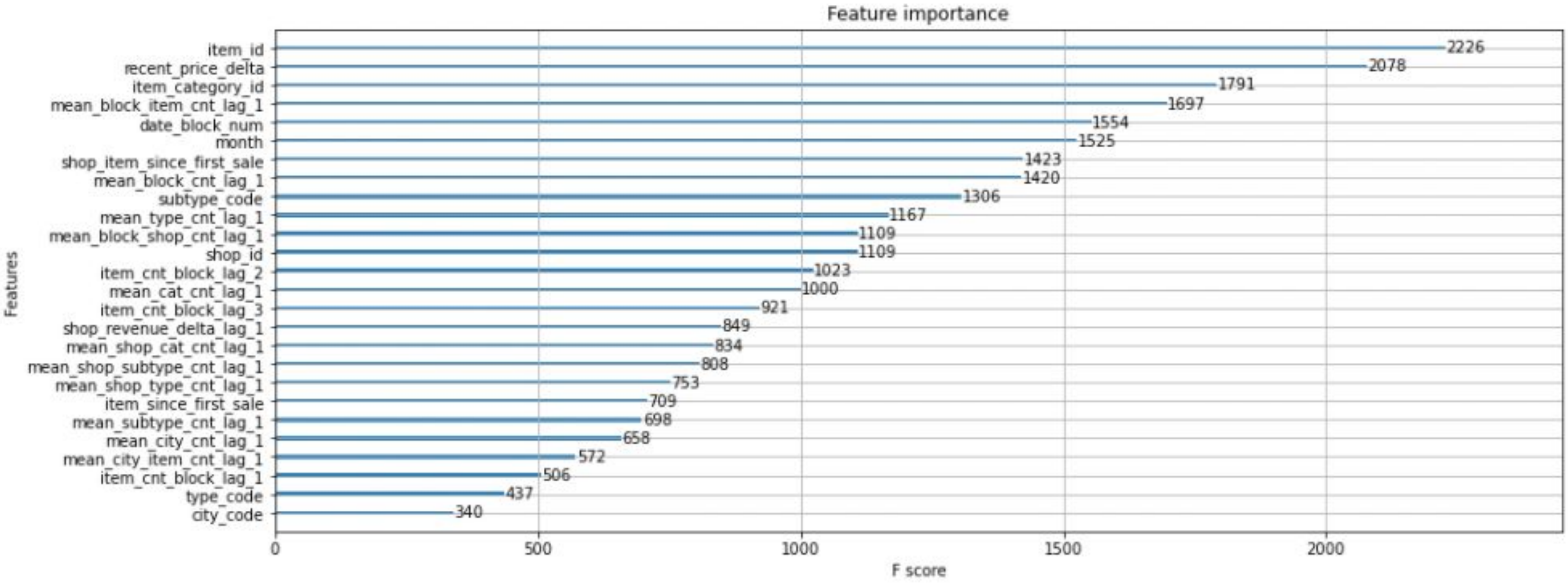
XGBoost

```python
xgb_model = XGBRegressor(
    max_depth=10,
    n_estimators=500,
    min_child_weight=100,
    colsample_bytree=0.8,
    subsample=0.8,
    eta=0.1,
    seed=42,
    n_jobs=-1)

xgb_model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=1,
    early_stopping_rounds = 5)
```

```
Will train until validation_1-rmse hasn't improved in 5 rounds.
[1]	validation_0-rmse:1.12750	validation_1-rmse:1.07936
[2]	validation_0-rmse:1.08439	validation_1-rmse:1.04846
[3]	validation_0-rmse:1.04968	validation_1-rmse:1.02231
[4]	validation_0-rmse:1.01931	validation_1-rmse:0.99962
[5]	validation_0-rmse:0.99456	validation_1-rmse:0.98253
[6]	validation_0-rmse:0.97099	validation_1-rmse:0.96729
[7]	validation_0-rmse:0.95350	validation_1-rmse:0.95535
[8]	validation_0-rmse:0.93685	validation_1-rmse:0.94603
[9]	validation_0-rmse:0.92341	validation_1-rmse:0.93819
[10]	validation_0-rmse:0.91192	validation_1-rmse:0.93083
[11]	validation_0-rmse:0.90179	validation_1-rmse:0.92446
[12]	validation_0-rmse:0.89300	validation_1-rmse:0.91965
[13]	validation_0-rmse:0.88549	validation_1-rmse:0.91690
[14]	validation_0-rmse:0.87869	validation_1-rmse:0.91424
[15]	validation_0-rmse:0.87294	validation_1-rmse:0.91187
[16]	validation_0-rmse:0.86806	validation_1-rmse:0.91049
[17]	validation_0-rmse:0.86313	validation_1-rmse:0.90911
[18]	validation_0-rmse:0.85933	validation_1-rmse:0.90738
[19]	validation_0-rmse:0.85581	validation_1-rmse:0.90643
[20]	validation_0-rmse:0.85250	validation_1-rmse:0.90485
[21]	validation_0-rmse:0.85021	validation_1-rmse:0.90381
[22]	validation_0-rmse:0.84775	validation_1-rmse:0.90299
[23]	validation_0-rmse:0.84545	validation_1-rmse:0.90260
[24]	validation_0-rmse:0.84355	validation_1-rmse:0.90221
[25]	validation_0-rmse:0.84184	validation_1-rmse:0.90191
[26]	validation_0-rmse:0.83952	validation_1-rmse:0.90161
[27]	validation_0-rmse:0.83772	validation_1-rmse:0.90309
[28]	validation_0-rmse:0.83627	validation_1-rmse:0.90372
[29]	validation_0-rmse:0.83481	validation_1-rmse:0.90342
[30]	validation_0-rmse:0.83352	validation_1-rmse:0.90281
[31]	validation_0-rmse:0.83219	validation_1-rmse:0.90242
Stopping. Best iteration:
[26]	validation_0-rmse:0.83952	validation_1-rmse:0.90161
```

Feature Importance



Light GBM

```python
lgbm_model = LGBMRegressor(
    boosting_type="gbdt",
    objective='regression',
    metric='rmse',
    random_state=42,
    n_estimators=50,
    num_leaves=32,
    max_depth=8,
    feature_fraction=0.8,
    bagging_fraction=0.8,
    bagging_freq=15,
    learning_rate=0.1,
    n_jobs=-1
)


xgb_model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=1,
    early_stopping_rounds = 5)
```

```
Will train until validation_1-rmse hasn't improved in 5 rounds.
[1]	validation_0-rmse:1.12636	validation_1-rmse:1.07894
[2]	validation_0-rmse:1.08283	validation_1-rmse:1.04771
[3]	validation_0-rmse:1.04765	validation_1-rmse:1.02387
[4]	validation_0-rmse:1.01663	validation_1-rmse:1.00186
[5]	validation_0-rmse:0.99124	validation_1-rmse:0.98260
[6]	validation_0-rmse:0.96715	validation_1-rmse:0.96695
[7]	validation_0-rmse:0.94930	validation_1-rmse:0.95504
[8]	validation_0-rmse:0.93212	validation_1-rmse:0.94511
[9]	validation_0-rmse:0.91809	validation_1-rmse:0.93740
[10]	validation_0-rmse:0.90606	validation_1-rmse:0.93026
[11]	validation_0-rmse:0.89563	validation_1-rmse:0.92313
[12]	validation_0-rmse:0.88619	validation_1-rmse:0.91879
[13]	validation_0-rmse:0.87813	validation_1-rmse:0.91844
[14]	validation_0-rmse:0.87098	validation_1-rmse:0.91569
[15]	validation_0-rmse:0.86453	validation_1-rmse:0.91325
[16]	validation_0-rmse:0.85916	validation_1-rmse:0.91209
[17]	validation_0-rmse:0.85421	validation_1-rmse:0.91041
[18]	validation_0-rmse:0.85036	validation_1-rmse:0.90918
[19]	validation_0-rmse:0.84650	validation_1-rmse:0.90837
[20]	validation_0-rmse:0.84272	validation_1-rmse:0.90758
[21]	validation_0-rmse:0.83975	validation_1-rmse:0.90701
[22]	validation_0-rmse:0.83706	validation_1-rmse:0.90655
[23]	validation_0-rmse:0.83439	validation_1-rmse:0.90671
[24]	validation_0-rmse:0.83217	validation_1-rmse:0.90623
[25]	validation_0-rmse:0.83023	validation_1-rmse:0.90594
[26]	validation_0-rmse:0.82762	validation_1-rmse:0.90555
[27]	validation_0-rmse:0.82562	validation_1-rmse:0.90795
[28]	validation_0-rmse:0.82385	validation_1-rmse:0.90873
[29]	validation_0-rmse:0.82184	validation_1-rmse:0.90829
[30]	validation_0-rmse:0.82032	validation_1-rmse:0.90792
[31]	validation_0-rmse:0.81887	validation_1-rmse:0.90800
Stopping. Best iteration:
[26]	validation_0-rmse:0.82762	validation_1-rmse:0.90555
```

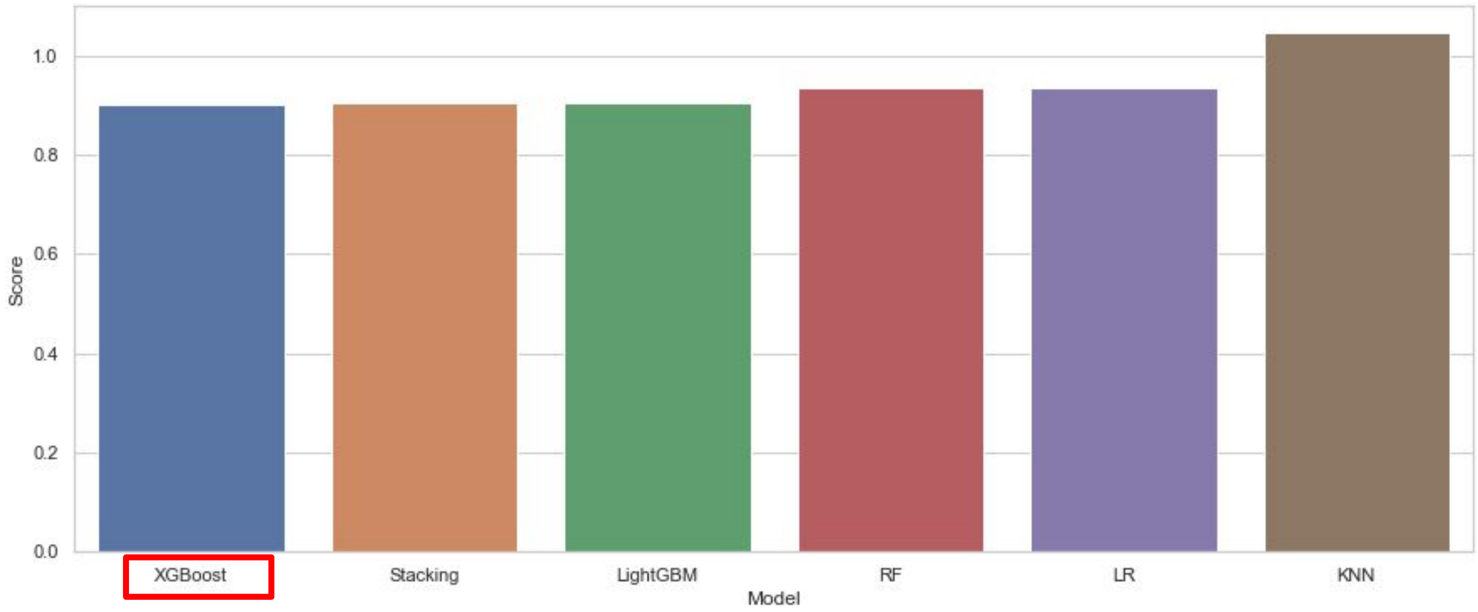
Light GBM

```python
lgbm_model = LGBMRegressor(
    boosting_type="gbdt",
    objective='regression',
    metric='rmse',
    random_state=42,
    n_estimators=50,
    num_leaves=32,
    max_depth=8,
    feature_fraction=0.8,
    bagging_fraction=0.8,
    bagging_freq=15,
    learning_rate=0.1,
    n_jobs=-1
)


xgb_model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=1,
    early_stopping_rounds = 5)
```

```
Will train until validation_1-rmse hasn't improved in 5 rounds.
[1]	validation_0-rmse:1.12636	validation_1-rmse:1.07894
[2]	validation_0-rmse:1.08283	validation_1-rmse:1.04771
[3]	validation_0-rmse:1.04765	validation_1-rmse:1.02387
[4]	validation_0-rmse:1.01663	validation_1-rmse:1.00186
[5]	validation_0-rmse:0.99124	validation_1-rmse:0.98260
[6]	validation_0-rmse:0.96715	validation_1-rmse:0.96695
[7]	validation_0-rmse:0.94930	validation_1-rmse:0.95504
[8]	validation_0-rmse:0.93212	validation_1-rmse:0.94511
[9]	validation_0-rmse:0.91809	validation_1-rmse:0.93740
[10]	validation_0-rmse:0.90606	validation_1-rmse:0.93026
[11]	validation_0-rmse:0.89563	validation_1-rmse:0.92313
[12]	validation_0-rmse:0.88619	validation_1-rmse:0.91879
[13]	validation_0-rmse:0.87813	validation_1-rmse:0.91844
[14]	validation_0-rmse:0.87098	validation_1-rmse:0.91569
[15]	validation_0-rmse:0.86453	validation_1-rmse:0.91325
[16]	validation_0-rmse:0.85916	validation_1-rmse:0.91209
[17]	validation_0-rmse:0.85421	validation_1-rmse:0.91041
[18]	validation_0-rmse:0.85036	validation_1-rmse:0.90918
[19]	validation_0-rmse:0.84650	validation_1-rmse:0.90837
[20]	validation_0-rmse:0.84272	validation_1-rmse:0.90758
[21]	validation_0-rmse:0.83975	validation_1-rmse:0.90701
[22]	validation_0-rmse:0.83706	validation_1-rmse:0.90655
[23]	validation_0-rmse:0.83439	validation_1-rmse:0.90671
[24]	validation_0-rmse:0.83217	validation_1-rmse:0.90623
[25]	validation_0-rmse:0.83023	validation_1-rmse:0.90594
[26]	validation_0-rmse:0.82762	validation_1-rmse:0.90555
[27]	validation_0-rmse:0.82562	validation_1-rmse:0.90795
[28]	validation_0-rmse:0.82385	validation_1-rmse:0.90873
[29]	validation_0-rmse:0.82184	validation_1-rmse:0.90829
[30]	validation_0-rmse:0.82032	validation_1-rmse:0.90792
[31]	validation_0-rmse:0.81887	validation_1-rmse:0.90800
Stopping. Best iteration:
[26]	validation_0-rmse:0.82762	validation_1-rmse:0.90555
```

Final Result



Next Steps:

- LSTM
- Googletrans, google translator API

Thank you.