Predict Future Sales

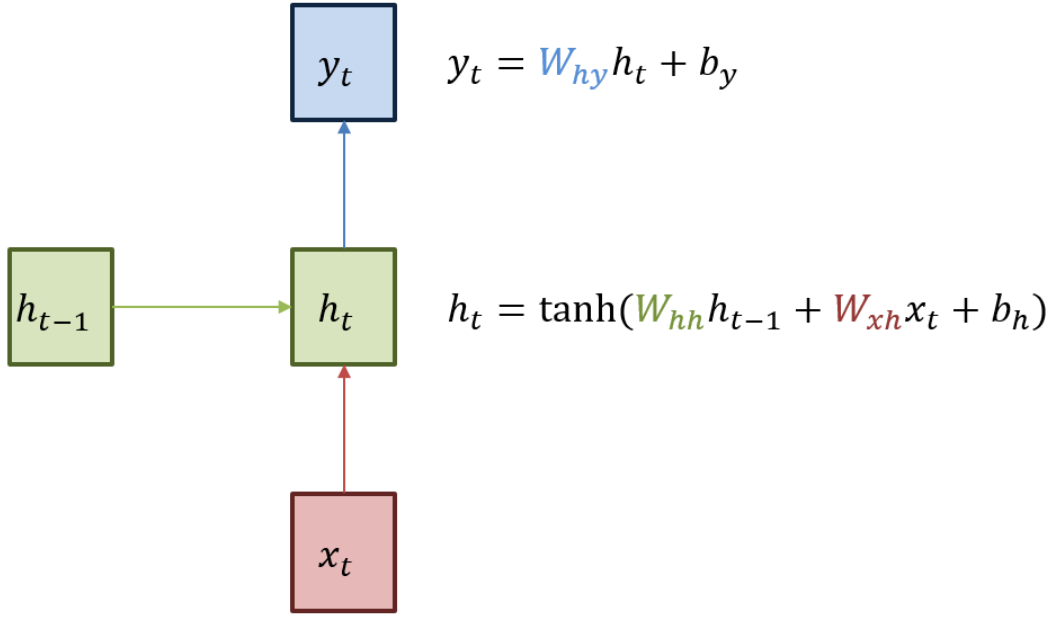
최연석, 오서영

1. LSTM

Predict Future Sales

- In this competition you will work with a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms.
  We are asking you to **predict total sales for every product and store in the next month.**

LSTM (Long Short Term Memory)
: 순차적으로 등장하는 데이터 처리에 적합한 모델



1. LSTM

1. Import Packages

```
import warnings

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.models import Sequential
from keras.layers import Dropout

from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.metrics import mean_squared_error
from numpy import sqrt
```

1. LSTM

2. Data Exploration

```
train = pd.read_csv('sales_train.csv')

print ('# shops: ', train['shop_id'].max())
print ('# items: ', train['item_id'].max())
print ('# month: ', train['date_block_num'].max())
print ('Shape of train : ', train.shape)
train.head()
```

```
# shops:  59
# items:  22169
# month:  33
Shape of train :  (2935849, 6)
```

	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

1. LSTM

```
test = pd.read_csv('test.csv')
test.head()
```

	ID	shop_id	item_id
0	0	5	5037
1	1	5	5320
2	2	5	5233
3	3	5	5232
4	4	5	5268

```
items = pd.read_csv('items.csv')
print ('# categories: ', items['item_category_id'].max())
print("Shape of items :", items.shape)
items.head()
```

```
# categories:  83
Shape of items : (22170, 3)
```

	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

1. LSTM

check!

- **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
- **date_block_num** : a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1,..., October 2015 is 33
- **date** : date in format dd/mm/yyyy
- **item_cnt_day** : number of products sold. You are predicting a monthly amount of this measure
- **item_price** : current price of an item
- **item_name** : name of item
- **shop_name** : name of shop
- **item_category_name** : name of item category

	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

1. LSTM

```
## Predict the number of products sold

sub = pd.read_csv('sample_submission.csv')
sub.head()
```

	ID	item_cnt_month
0	0	0.5
1	1	0.5
2	2	0.5
3	3	0.5
4	4	0.5

1. LSTM

3. Make dataset - Practice with Example

```
train_clean = train.drop(labels = ['date', 'item_price'], axis = 1)

train_clean = train_clean.groupby(["item_id","shop_id","date_block_num"]).sum().reset_index()
train_clean = train_clean.rename(index = str, columns = {"item_cnt_day":"item_cnt_month"})
train_clean = train_clean[["item_id","shop_id","date_block_num","item_cnt_month"]]

print("Shape of train after cleaning :", train_clean.shape)
train_clean.head()
```

```
Shape of train after cleaning : (1609124, 4)
```

	item_id	shop_id	date_block_num	item_cnt_month
0	0	54	20	1.0
1	1	55	15	2.0
2	1	55	18	1.0
3	1	55	19	1.0
4	1	55	20	1.0

1. LSTM

```
check = train_clean[["shop_id","item_id","date_block_num","item_cnt_month"]]
check = check.loc[check['shop_id'] == 10]
check = check.loc[check['item_id'] == 5037]

check
```

	shop_id	item_id	date_block_num	item_cnt_month
400473	10	5037	20	5.0
400474	10	5037	21	2.0
400475	10	5037	22	1.0
400476	10	5037	23	1.0
400477	10	5037	24	1.0
400478	10	5037	31	1.0

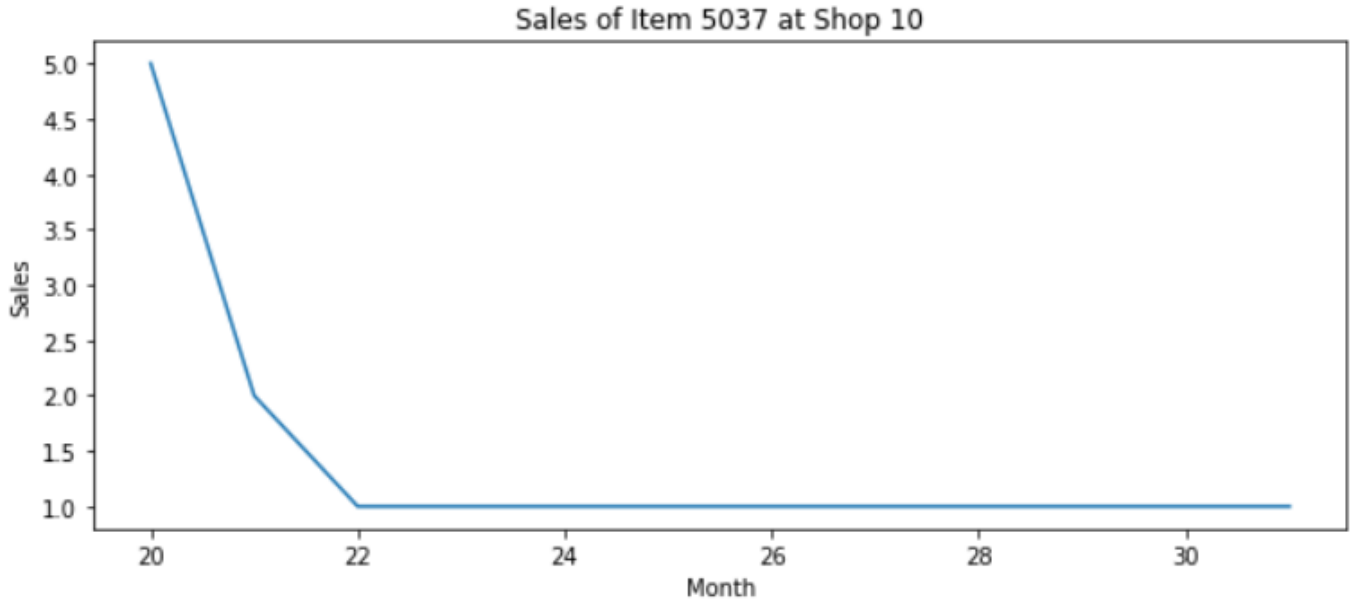
1. LSTM

check!

- 10_shop sold five 5037_items on 20_date.

```
# Interpreting an example

plt.figure(figsize=(10,4))
plt.title('Sales of Item 5037 at Shop 10')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.plot(check["date_block_num"], check["item_cnt_month"]);
```



1. LSTM

```
num_month = train['date_block_num'].max()
month_list=[i for i in range(num_month+1)]

shop = []
for i in range(num_month+1):
    shop.append(10)

item = []

for i in range(num_month+1):
    item.append(5037)

ans = pd.DataFrame({'shop_id':shop, 'item_id':item,'date_block_num':month_list})
ans
```

	shop_id	item_id	date_block_num
0	10	5037	0
1	10	5037	1
2	10	5037	2
3	10	5037	3
4	10	5037	4
5	10	5037	5

1. LSTM

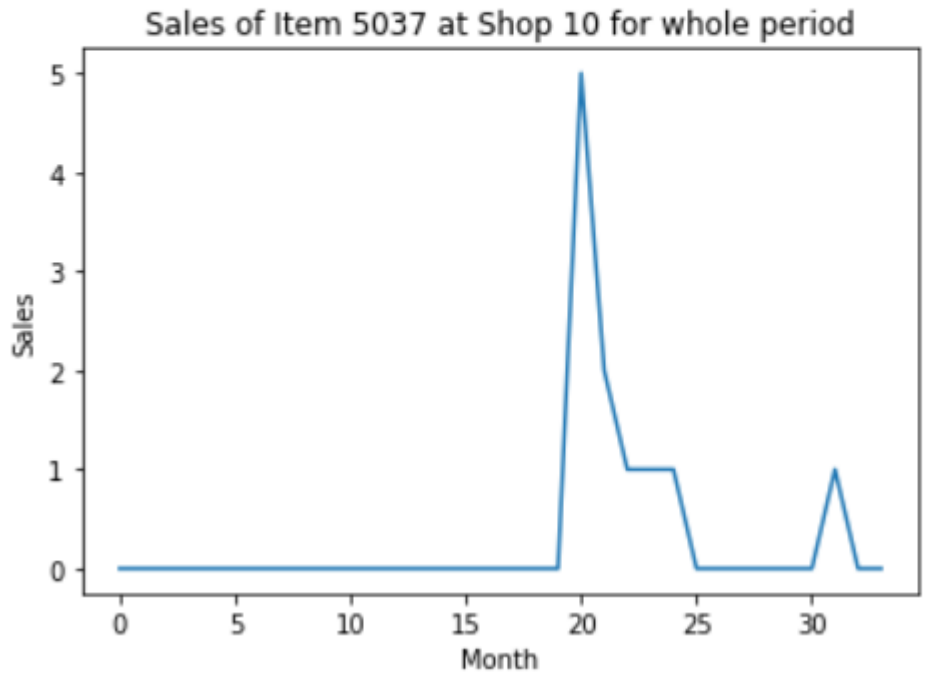
```
ex_sales = pd.merge(check, ans, how = 'right', on = ['shop_id','item_id','date_block_num'])
ex_sales = ex_sales.sort_values(by = ['date_block_num'])
ex_sales.fillna(0.00, inplace=True)
ex_sales
```

	shop_id	item_id	date_block_num	item_cnt_month
6	10	5037	0	0.0
7	10	5037	1	0.0
8	10	5037	2	0.0
9	10	5037	3	0.0
10	10	5037	4	0.0
11	10	5037	5	0.0
12	10	5037	6	0.0
13	10	5037	7	0.0

1. LSTM

```
# Interpreting an example

plt.title('Sales of Item 5037 at Shop 10 for whole period')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.plot(ex_sales["date_block_num"], ex_sales["item_cnt_month"]);
```



1. LSTM

```
for i in range(1,6):
    ex_sales["# " + str(i)] = ex_sales.item_cnt_month.shift(i)
ex_sales.fillna(0.0, inplace=True)
ex_sales
```

20	10	5037	14	0.0	0.0	0.0	0.0	0.0	0.0
21	10	5037	15	0.0	0.0	0.0	0.0	0.0	0.0
22	10	5037	16	0.0	0.0	0.0	0.0	0.0	0.0
23	10	5037	17	0.0	0.0	0.0	0.0	0.0	0.0
24	10	5037	18	0.0	0.0	0.0	0.0	0.0	0.0
25	10	5037	19	0.0	0.0	0.0	0.0	0.0	0.0
0	10	5037	20	5.0	0.0	0.0	0.0	0.0	0.0
1	10	5037	21	2.0	5.0	0.0	0.0	0.0	0.0
2	10	5037	22	1.0	2.0	5.0	0.0	0.0	0.0
3	10	5037	23	1.0	1.0	2.0	5.0	0.0	0.0
4	10	5037	24	1.0	1.0	1.0	2.0	5.0	0.0
26	10	5037	25	0.0	1.0	1.0	1.0	2.0	5.0

0	10	5037	20	5.0
1	10	5037	21	2.0
2	10	5037	22	1.0
3	10	5037	23	1.0
4	10	5037	24	1.0

1. LSTM

```
df = ex_sales[['shop_id','item_id','date_block_num','# 1','# 2','# 3','# 4','# 5', 'item_cnt_month']]
df = df.drop(labels = ['index'], axis = 1)                                    .reset_index()
df
```

	shop_id	item_id	date_block_num	# 1	# 2	# 3	# 4	# 5	item_cnt_month
0	10	5037	0	0.0	0.0	0.0	0.0	0.0	0.0
1	10	5037	1	0.0	0.0	0.0	0.0	0.0	0.0
2	10	5037	2	0.0	0.0	0.0	0.0	0.0	0.0
20	10	5037	20	0.0	0.0	0.0	0.0	0.0	5.0
21	10	5037	21	5.0	0.0	0.0	0.0	0.0	2.0
22	10	5037	22	2.0	5.0	0.0	0.0	0.0	1.0
23	10	5037	23	1.0	2.0	5.0	0.0	0.0	1.0
24	10	5037	24	1.0	1.0	2.0	5.0	0.0	1.0

1. LSTM

4. Make Dataset for training

```
dataset = pd.read_csv('sales_train.csv')
testset = pd.read_csv('test.csv')
dataset.head()
```

	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

1. LSTM

```
# make our data in desired form
# we need total count value of an item over the whole month for a shop

dataset['date'] = pd.to_datetime(dataset['date'],format = '%d.%m.%Y')
dataset = dataset.pivot_table(index = ['shop_id','item_id'], values = ['item_cnt_day'],
                              columns = ['date_block_num'], fill_value = 0, aggfunc='sum')

dataset.reset_index(inplace = True)  # easy to manipulate
dataset.head()
```

	shop_id	item_id	item_cnt_day																		
date_block_num			0	1	2	3	4	5	6	7	...	24	25	26	27	28	29	30	31	32	33
0	0	30	0	31	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	31	0	11	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	32	6	10	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	33	3	3	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	35	1	14	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 36 columns

```
# we want to keep the data of items we have
dataset = pd.merge(testset,dataset,on = ['item_id','shop_id'],how = 'left')
dataset.drop(['shop_id','item_id','ID'],inplace = True, axis = 1)
```

1. LSTM

```
dataset.fillna(0,inplace = True)
dataset.head()
```

	(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, 6)
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 34 columns

```
x_train = np.expand_dims(dataset.values[:,:-1], axis = 2)  # execpt the last one

y_train = dataset.values[:,-1:] # last column is label

x_test = np.expand_dims(dataset.values[:,1:],axis = 2)  # all the columns execpt the first one

# lets have a look on the shape
print("Shape of x_train :", x_train.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of x_test :", x_test.shape)
```

```
Shape of x_train : (214200, 33, 1)
Shape of y_train : (214200, 1)
Shape of x_test : (214200, 33, 1)
```

1. LSTM

5. LSTM

```
model = Sequential()
model.add(LSTM(64, input_shape=(33,1)))
model.add(Dropout(0.4))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 64)                16896
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

1. LSTM

```
%%time
hist10 = model.fit(x_train,y_train,batch_size = 4096, epochs = 10, verbose=1, shuffle = False)
```

```
Epoch 1/10
214200/214200 [==============================] - 56s 260us/step - loss: 30.1989 - mean_squared_error: 30.1989
Epoch 2/10
214200/214200 [==============================] - 56s 260us/step - loss: 30.0581 - mean_squared_error: 30.0581
Epoch 3/10
214200/214200 [==============================] - 56s 261us/step - loss: 29.9824 - mean_squared_error: 29.9824
```

```
%%time
hist50 = model.fit(x_train,y_train,batch_size = 4096, epochs = 50, verbose=1, shuffle = False)
```

```
Epoch 1/50
214200/214200 [==============================] - 55s 256us/step - loss: 29.5999 - mean_squared_error: 29.5999
Epoch 2/50
214200/214200 [==============================] - 57s 264us/step - loss: 29.4247 - mean_squared_error: 29.4247
Epoch 3/50
214200/214200 [==============================] - 57s 264us/step - loss: 29.4045 - mean_squared_error: 29.4045
Epoch 4/50
```

1. LSTM - Submit

10 iterations
RMSE error :
1.02245

60 iterations
RMSE error :
1.02711

4006등

2. XGBregressor

XGB Regressor (Extreme Gradient Boosting)
: 트리 기반의 앙상블 학습법

-> Kaggle Competition에서 주로 상위를 차지
-> 빠르고 성능이 좋아 자주 사용됨

2. XGBregressor

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [3]:
```
train = pd.read_csv('sales_train.csv')
test = pd.read_csv('test.csv')
categories = pd.read_csv('item_categories.csv')
item = pd.read_csv('items.csv')
shop = pd.read_csv('shops.csv')
```

2. XGBregressor

```
In [4]: train.head()
```

Out[4]:

	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

```
In [5]: train['date'] = pd.to_datetime(train.date,format="%d.%m.%Y")

In [6]: train.head()
```

Out[6]:

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

2. XGBregressor

In [14]:
```
con_data = pd.concat([train,test], ignore_index=True)
con_data.sample(5)
```

Out[14]:

	date_block_num	shop_id	item_id	item_price
2075722	21.0	57	17281	299.0
143442	1.0	18	2281	499.0
1775991	17.0	35	6500	349.5
1159667	11.0	28	4910	599.0
2771089	30.0	27	11497	799.0

In [9]:
```
train_id = train.ID
test_id = test.ID
y_sales = train.item_cnt_day
```

0	5	5037
1	5	5320
2	5	5233
3	5	5232
4	5	5268

2. XGBregressor

In [15]:
```
con_data.columns
```

Out[15]:
```
Index(['date_block_num', 'shop_id', 'item_id', 'item_price'], dtype='object')
```

In [16]:
```
con_data["item_price"] = con_data["item_price"].fillna((con_data["item_price"].mode()[0] ))
con_data["date_block_num"] = con_data["date_block_num"].fillna((con_data["date_block_num"].mode()[0] ))
con_data.isna()
con_data.isnull().sum()
```

Out[16]:
```
date_block_num    0
shop_id           0
item_id           0
item_price        0
dtype: int64
```

In [17]:
```
x_train = con_data[:len(train)]
x_test = con_data[len(train):]
train_x, test_x, train_y, test_y = train_test_split(x_train, y_sales,test_size = 0.2, random_state = 0)
```

In [18]:
```
from  sklearn.preprocessing  import StandardScaler
slc= StandardScaler()
train_x = slc.fit_transform(train_x)
x_test = slc.transform(x_test)
test_x = slc.transform(test_x)
```

In [19]:
```
num_folds = 10
seed = 0
scoring = 'neg_mean_squared_error'
kfold = KFold(n_splits=num_folds, random_state=seed)
```

2. XGBregressor

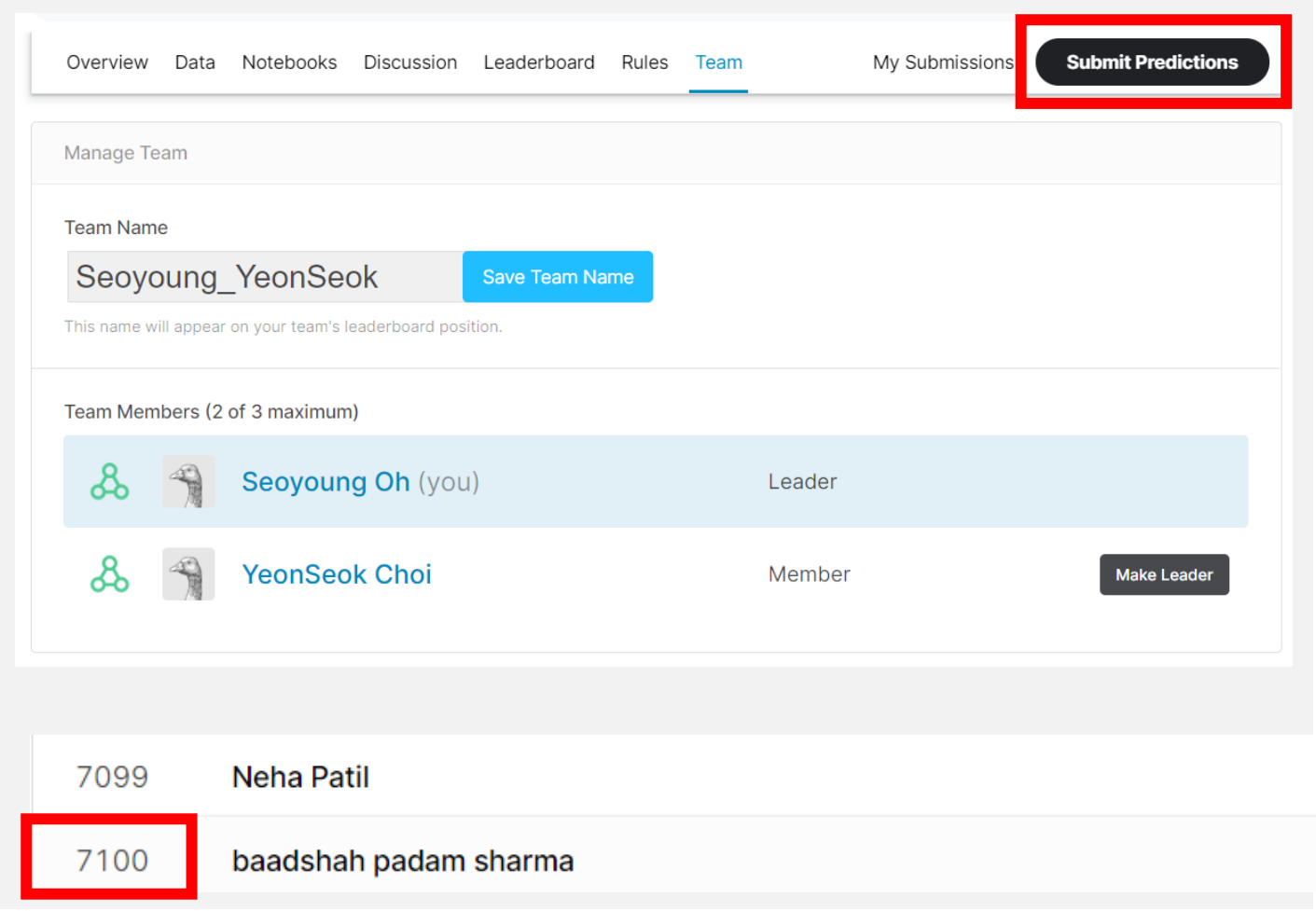
```
In [20]: model = XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.3, learning_rate = 0.1, ##
                              max_depth = 10, alpha = 10, n_estimators = 70)

In [21]: score_= cross_val_score(model, train_x, train_y, cv=kfold, scoring=scoring)

In [22]: model.fit(train_x, train_y, verbose=1)

Out[22]: XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=0.3, gamma=0, gpu_id=-1,
                      importance_type='gain', interaction_constraints='',
                      learning_rate=0.1, max_delta_step=0, max_depth=10,
                      min_child_weight=1, missing=nan, monotone_constraints='()',
                      n_estimators=70, n_jobs=0, num_parallel_tree=1,
                      objective='reg:squarederror', random_state=0, reg_alpha=10,
                      reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
                      validate_parameters=1, verbosity=None)
```

2. XGBregressor - Submit



RMSE error
: 1.43635

-> 7100 등

Conclusion

Feature Engineering !

Reference

[1] Simple Predict with Xgboost,
https://www.kaggle.com/doukanberkberber/simple-predict-with-xgboost

[2] Sales Forecast LSTM - 67% (Beginner-Friendly),
https://www.kaggle.com/carmnejsu/sales-forecast-lstm-67-beginner-friendly

[3] Simple and Easy Aprroach using LSTM,
https://www.kaggle.com/karanjakhar/simple-and-easy-aprroach-using-lstm