1 Business Problem

1.1 Overview

In the era of online stores, It's become essential and mandatory to know the demand of any product in near future. It will help stores to stock the product which demand is going to be high in near future. If we are able to do so, it will directly improve revenue of company.
i.e. if we have any event in next week, definitely the sell will increase on some product depends on the event.

1.2 Objective:

Our main objective is to predict the product's sell for the next 28 days.

1.3 Performance Matrix:

RMSE (Root Mean Square Error) : RMSE is a good measure when we care more about prediction. It is a square root of the average squared error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

RMSE gives more importance to the most significant error. If there is slighlty error in demand our RMSE will increase significantly.

1.4 Why ML approach, and Why not normal Statstical Method

Statstical method can perfrom good when we have univarite dataset or one step prediction. It has been seen that for multi step prediction , ML methods works well.

In [2]:
```python
sample_sub = pd.read_csv("sample_submission.csv") #sample submission file
sample_sub.head()
```

Out[2]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20	F21	F22	F23	F24	F25
0	HOBBIES_1_001_CA_1_validation	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	HOBBIES_1_002_CA_1_validation	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	HOBBIES_1_003_CA_1_validation	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
3	HOBBIES_1_004_CA_1_validation	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
4	HOBBIES_1_005_CA_1_validation	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 29 columns

In [4]:
```python
# #load data
calendar = pd.read_csv('calendar.csv')
sell_price = pd.read_csv("sell_prices.csv")
sales_train_eval = pd.read_csv("sales_train_evaluation.csv")
```

In [6]:
```python
#downcast all the dataframe to avoid memory error
sample_sub = downcast(sample_sub)
sales_train_eval = downcast(sales_train_eval)
sell_price = downcast(sell_price)
calendar = downcast(calendar)
```

```
29it [00:00, 204.38it/s]
1947it [01:16, 25.54it/s]
4it [00:01,  3.72it/s]
14it [00:00, 1146.97it/s]
```

In [7]:

```
"""
1. Add new categories as no event  in event_name_1, event_name_2, event_type_1, event_type_2
2. fill the null values to no event
Note:  we are filling the null values before spltting to avoid memory error and
"""

calendar["event_name_1"] = calendar["event_name_1"].cat.add_categories("no_event")
calendar["event_name_1"] = calendar['event_name_1'].fillna("no_event")

calendar["event_name_2"] = calendar["event_name_2"].cat.add_categories("no_event")
calendar["event_name_2"] = calendar['event_name_2'].fillna("no_event")

calendar["event_type_2"] = calendar["event_type_2"].cat.add_categories("no_event")
calendar["event_type_2"] = calendar['event_type_2'].fillna("no_event")

calendar["event_type_1"] = calendar["event_type_1"].cat.add_categories("no_event")
calendar["event_type_1"] = calendar['event_type_1'].fillna("no_event")
```

In [8]:
```
#melt sales_train_eval data
sales_train_eval = pd.melt(sales_train_eval, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id'], var

#downacast the dataframe
sales_train_eval = downcast(sales_train_eval)

"""we have 60K datapoints in sample submission with 30k from evaluation and 30k from validation
1. submission_rows_eval = get all the id which has evaluation in the end
2. submission_rows_val = get all the id which has validation in the end
"""
submission_rows_eval = [row for row in sample_sub['id'] if 'evaluation' in row] #get all the evaluation rows
submission_rows_val = [row for row in sample_sub['id'] if 'validation' in row]#get all the validation rows


#get all evaluation from submission data
submission_eval = sample_sub[sample_sub['id'].isin(submission_rows_eval)]

#get all valdation from submission data
submission_val = sample_sub[sample_sub['id'].isin(submission_rows_val)]


"""split the data in eval and val part with by days.
submission_eval will start from d_1914 and end on d_1941
submission_val will start from d_1942 and end on d_1969

"""
submission_eval.columns = ['id', 'd_1914', 'd_1915', 'd_1916', 'd_1917', 'd_1918', 'd_1919', 'd_1920', 'd_1921', 'd_1922',
                      'd_1932', 'd_1933', 'd_1934', 'd_1935', 'd_1936', 'd_1937', 'd_1938', 'd_1939', 'd_1940', 'd_1941']

submission_val.columns = ['id', 'd_1942', 'd_1943', 'd_1944', 'd_1945', 'd_1946', 'd_1947', 'd_1948', 'd_1949', 'd_1950',
'd_1953', 'd_1954', 'd_1955', 'd_1956', 'd_1957', 'd_1958', 'd_1959',
                      'd_1960', 'd_1961', 'd_1962', 'd_1963', 'd_1964', 'd_1965', 'd_1966', 'd_1967', 'd_1968', 'd_1969']


#dropping duplicates if any
drop_duplicate_product = sales_train_eval[['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id']].drop_duplicates(
submission_val['id'] = submission_val['id'].replace("_validation" , "_evaluation")

#merge validation and evaluation data on id
submission_val = submission_val.merge(drop_duplicate_product, how = 'left', on = 'id')
submission_eval = submission_eval.merge(drop_duplicate_product, how = 'left', on = 'id')
submission_val['id'] = submission_val['id'].replace( "_evaluation", "_validation" )

submission_val = pd.melt(submission_val, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id'],
                         var_name = 'day', value_name = 'demand')

submission_eval = pd.melt(submission_eval, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id'],
```

```
                           var_name = 'day', value_name = 'demand')

#create one new columns as part which has our train, test1 and test2
sales_train_eval['part'] = 'train'
submission_eval['part'] = 'test1'
submission_val['part'] = 'test2'

data = pd.concat([sales_train_eval, submission_val, submission_eval], axis = 0)

data = downcast(data)
```

```
8it [00:04,  1.89it/s]
9it [00:22,  2.45s/it]
```

In [9]:
```
#drop 'weekday', 'wday', 'month', 'year' from calendar
calendar.drop(['weekday', 'wday', 'month', 'year'], inplace = True, axis = 1)
data = data[data['part'] != 'test2']
data
```

Out[9]:

	id	item_id	dept_id	cat_id	store_id	state_id	day	demand	part
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	train
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	train
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	train
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	train
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	train
...	...	...	...	...	...	...	...	...	...
853715	FOODS_3_823_WI_3_evaluation	FOODS_3_823	FOODS_3	FOODS	WI_3	WI	d_1941	0	test1
853716	FOODS_3_824_WI_3_evaluation	FOODS_3_824	FOODS_3	FOODS	WI_3	WI	d_1941	0	test1
853717	FOODS_3_825_WI_3_evaluation	FOODS_3_825	FOODS_3	FOODS	WI_3	WI	d_1941	0	test1
853718	FOODS_3_826_WI_3_evaluation	FOODS_3_826	FOODS_3	FOODS	WI_3	WI	d_1941	0	test1
853719	FOODS_3_827_WI_3_evaluation	FOODS_3_827	FOODS_3	FOODS	WI_3	WI	d_1941	0	test1

60034810 rows × 9 columns

In [12]:

```
#merge data with calendar
data = pd.merge(data, calendar, how = 'left',left_on = ['day'], right_on = ['d'])
data.drop(['d', 'day'], inplace = True, axis = 1)
data
```

Out[12]:

	id	item_id	dept_id	cat_id	store_id	state_id	demand	part	date	wm_yr_wk	eve
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	train	2011-01-29	11101	
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	train	2011-01-29	11101	
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	train	2011-01-29	11101	
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	train	2011-01-29	11101	
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	train	2011-01-29	11101	
...	...	...	...	...	...	...	...	...	...	...	
60034805	FOODS_3_823_WI_3_evaluation	FOODS_3_823	FOODS_3	FOODS	WI_3	WI	0	test1	2016-05-22	11617	
60034806	FOODS_3_824_WI_3_evaluation	FOODS_3_824	FOODS_3	FOODS	WI_3	WI	0	test1	2016-05-22	11617	
60034807	FOODS_3_825_WI_3_evaluation	FOODS_3_825	FOODS_3	FOODS	WI_3	WI	0	test1	2016-05-22	11617	
60034808	FOODS_3_826_WI_3_evaluation	FOODS_3_826	FOODS_3	FOODS	WI_3	WI	0	test1	2016-05-22	11617	
60034809	FOODS_3_827_WI_3_evaluation	FOODS_3_827	FOODS_3	FOODS	WI_3	WI	0	test1	2016-05-22	11617	

60034810 rows × 17 columns

In [13]:
```
#merge data with sell price
data = data.merge(sell_price, on = ['store_id', 'item_id', 'wm_yr_wk'], how = 'left')
print(f' dataset  has {data.shape[0]} rows and {data.shape[1]} columns')
data = downcast(data)
```

```
0it [00:00, ?it/s]

 dataset  has 60034810 rows and 18 columns

18it [00:02,  6.24it/s]
```

10 Feature Enginerring

In [14]:
```
"""
Label Encode of all categorical values present in dataset

1. For 'event_name_1', 'event_type_1', 'event_name_2' and  'event_type_2' will check null values and fill no_event and the
2. For state_id, store_id, cat_id, dept_id, item_id will direclty label encode


"""
from sklearn.preprocessing import LabelEncoder
def label_encoding(train, feature):
    nan_features = ['event_name_1', 'event_type_1', 'event_name_2', 'event_type_2']
    if feature in nan_features:
        data[feature].fillna("no_event", inplace=True)

    encoder = LabelEncoder()
    encoder.fit(train[feature].values.astype(str))
    train[feature] = encoder.fit_transform(data[feature].values.astype(str))

    return train[feature]
```

In [15]:
```
data['item_id']  = label_encoding(data,"item_id" )
data['dept_id']  = label_encoding(data,"dept_id" )
data['cat_id'] = label_encoding(data,"cat_id" )
data['store_id']  = label_encoding(data,"store_id" )
data['state_id']  = label_encoding(data,"state_id" )
data['event_name_1']  = label_encoding(data,"event_name_1" )
data['event_name_2']  = label_encoding(data,"event_name_2" )
data['event_type_1']  = label_encoding(data,"event_type_1" )
data['event_type_2']  = label_encoding(data,"event_type_2" )
```

In [16]:
```
#changeing in date dataframe
data['date'] = pd.to_datetime(data['date'])
data['year']=data['date'].dt.year
data['month']=data['date'].dt.month
data['day']=data['date'].dt.day
data['week']=data['date'].dt.week
```

In [17]:
```
data.to_pickle("data.pkl")  #save all the data
```

11 Load Data

In [2]:
```python
data = pd.read_pickle("data.pkl")
data.head()
```

Out[2]:

	id	item_id	dept_id	cat_id	store_id	state_id	demand	part	date	wm_yr_wk	...	event_name_
0	HOBBIES_1_001_CA_1_evaluation	1437	3	1	0	0	0	train	2011-01-29	11101	...	
1	HOBBIES_1_002_CA_1_evaluation	1438	3	1	0	0	0	train	2011-01-29	11101	...	
2	HOBBIES_1_003_CA_1_evaluation	1439	3	1	0	0	0	train	2011-01-29	11101	...	
3	HOBBIES_1_004_CA_1_evaluation	1440	3	1	0	0	0	train	2011-01-29	11101	...	
4	HOBBIES_1_005_CA_1_evaluation	1441	3	1	0	0	0	train	2011-01-29	11101	...	

5 rows × 22 columns

In [3]:
```python
sample_sub = pd.read_csv("sample_submission.csv")
```

11.0.0.1 lag feature:

3 days shift from 28 day to 30 days.
Here we will create 3 new features lag_28, lag_29 and lag_30 which will be shift by days 28, 29 and 30 respectively

In [4]:
```python
#adding rollog features
from tqdm import tqdm
#https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/134777
for i in tqdm(range(28, 31)):
    index_name = "lag_"+str(i)
    data[index_name] = data.groupby(['id'])['demand'].transform(lambda x: x.shift(i))
```

```
100%|██████████| 3/3 [01:50<00:00, 36.85s/it]
```

11.0.0.2 Rolling

taking group of n (here 7 days) and take the avg of it.

In [8]:
```
data['rolling_mean_t7'] = data.groupby(['id'])['demand'].transform(lambda x: x.shift(28).rolling(7).mean())
data['rolling_std_t7'] = data.groupby(['id'])['demand'].transform(lambda x: x.shift(28).rolling(7).std())

data = downcast(data)
```

```
28it [00:16,  1.68it/s]
```

In [9]:
```
#fill the missing values with interpolate
data['sell_price_inter'] = data['sell_price'].interpolate(method='linear', inplace=True)

#we left with 7 missing values after filling with interpolate, so fill with 0
data['sell_price'] = data['sell_price'].fillna(0)

#filling all the nan values to zero
data['rolling_mean_t7'] = data['rolling_mean_t7'].fillna(0)
data['rolling_std_t7'] = data['rolling_std_t7'].fillna(0)
data['lag_28'] = data['lag_28'].fillna(0)
data['lag_29'] = data['lag_29'].fillna(0)
data['lag_30'] = data['lag_30'].fillna(0)
data = downcast(data)
```

```
28it [00:15,  1.75it/s]
```

In [11]:
```python
#checking null values

def missing_values(df):
    missing_count = df.isnull().sum()
    percent_missing = missing_count * 100 / len(df)
    values_available = len(df) - missing_count

    missing_value_df = pd.DataFrame({
        'column_name': df.columns,
        "non_misisng_count": values_available,
        'missing_count': missing_count,

        'percent_missing': percent_missing
    })
    missing_value_df.sort_values('percent_missing', inplace=True)

    return missing_value_df


missing_values(data)
```

Out[11]:

	column_name	non_misisng_count	missing_count	percent_missing
id	id	60034810	0	0.0
rolling_mean_t7	rolling_mean_t7	60034810	0	0.0
lag_30	lag_30	60034810	0	0.0
lag_29	lag_29	60034810	0	0.0
lag_28	lag_28	60034810	0	0.0
week	week	60034810	0	0.0
day	day	60034810	0	0.0
month	month	60034810	0	0.0
year	year	60034810	0	0.0
sell_price	sell_price	60034810	0	0.0
snap_WI	snap_WI	60034810	0	0.0
snap_TX	snap_TX	60034810	0	0.0
snap_CA	snap_CA	60034810	0	0.0
event_type_2	event_type_2	60034810	0	0.0
event_name_2	event_name_2	60034810	0	0.0

	column_name	non_misisng_count	missing_count	percent_missing
event_type_1	event_type_1	60034810	0	0.0
event_name_1	event_name_1	60034810	0	0.0
wm_yr_wk	wm_yr_wk	60034810	0	0.0
date	date	60034810	0	0.0
part	part	60034810	0	0.0
demand	demand	60034810	0	0.0
state_id	state_id	60034810	0	0.0
store_id	store_id	60034810	0	0.0
cat_id	cat_id	60034810	0	0.0
dept_id	dept_id	60034810	0	0.0
item_id	item_id	60034810	0	0.0
rolling_std_t7	rolling_std_t7	60034810	0	0.0
sell_price_inter	sell_price_inter	0	60034810	100.0

In [12]:
```
#split data time wise
x_train = data[data['date'] <= '2016-03-27']
y_train = x_train['demand']
x_val = data[(data['date'] <= '2016-04-24') & (data['date'] > '2016-03-27')]
y_val = x_val['demand']
test = data[data['date'] > '2016-04-24']
```

In [13]:
```
col = [ 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id',
       'wm_yr_wk', 'event_name_1', 'event_type_1',
       'event_name_2', 'event_type_2', 'snap_CA', 'snap_TX', 'snap_WI',
       'sell_price', 'lag_28', 'lag_29', 'lag_30', 'rolling_mean_t7',
       'rolling_std_t7', 'year', 'month', 'day', 'week']
```

12 Model -1 -Light GBM

In [ ]:

```python
import lightgbm as lgb
from sklearn.metrics import mean_squared_error
params = {
    'boosting_type': 'gbdt',
    'metric': 'rmse',
    'objective': 'regression',
    'n_jobs': -1,
    'seed': 236,
    'learning_rate': 0.1,
    'bagging_fraction': 0.75,
    'bagging_freq': 10,
    'colsample_bytree': 0.75}

train_set = lgb.Dataset(x_train[col], y_train)
val_set = lgb.Dataset(x_val[col], y_val)
```

In [24]:

```python
model = lgb.train(params, train_set, num_boost_round = 2500, early_stopping_rounds =200, valid_sets = [train_set, val_set]
val_pred = model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f' rmse score is {val_score}')
y_pred = model.predict(test[col])
test['demand'] = y_pred
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 5.318051 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1700
[LightGBM] [Info] Number of data points in the train set: 57473650, number of used features: 23
[LightGBM] [Info] Start training from score 1.122458
Training until validation scores don't improve for 200 rounds
[100]	training's rmse: 2.46608	valid_1's rmse: 2.28935
[200]	training's rmse: 2.41984	valid_1's rmse: 2.31824
Early stopping, best iteration is:
[60]	training's rmse: 2.49988	valid_1's rmse: 2.2813
 rmse score is 5.204316944117257
```

In [25]:
```python
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
(60980, 29)
```

Out[25]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.866731	0.691625	0.549294	0.516352	0.476673	0.613061	0.924286	0.889703	1.102997	...	0.847543	1.250974
1	HOBBIES_1_002_CA_1_evaluation	0.474192	0.474192	0.462583	0.417932	0.384453	0.357331	0.357331	0.320699	0.277132	...	0.270198	0.332308
2	HOBBIES_1_003_CA_1_evaluation	0.500992	0.470716	0.482325	0.512601	0.473532	0.484464	0.533440	0.539595	0.499525	...	0.762564	0.768874
3	HOBBIES_1_004_CA_1_evaluation	2.387544	2.040123	1.797237	1.649948	2.009645	1.286213	1.407418	1.651772	1.789729	...	1.533549	1.227199
4	HOBBIES_1_005_CA_1_evaluation	1.093541	0.887497	1.168761	1.601959	1.460028	1.264221	1.583555	1.506963	1.737813	...	1.234848	1.102608

5 rows × 29 columns

In [26]:
```python
final.to_csv('submission.csv', index = False)
```

In [27]:
```python
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "lightgbm-1"
```

```
100%|██████████████████████████████| 20.4M/20.4M [00:01<00:00, 12.0MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

submission.csv 0.86572 5.44561

15 minutes ago by Namratesh Shrivastav

lightgbm

13 light GbM hyper tunning

In [11]:
```python
from sklearn.model_selection import GridSearchCV

from lightgbm import LGBMRegressor
hyperparameters = {
'boosting_type': ['gbdt'],
'metric': ['rmse'],
'objective': ['regression'],
'n_jobs': [-1],
'seed':[ 236],
'learning_rate': [0.1, 0.2, 0.3],
'bagging_fraction': [0.75],
'bagging_freq': [10, 5],
'colsample_bytree':[ 0.75]}

clf = LGBMRegressor()
gsearch = GridSearchCV(estimator=clf, param_grid=hyperparameters)
gsearch.fit(x_train[col], y_train)
print(gsearch.best_params_, gsearch.best_score_)
```

```
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
```

```
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=10, subsample_freq=0 will be ignored. Current value: bagging_freq=10
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.75, subsample=1.0 will be ignored. Current value: bagging_fraction=0.75
{'bagging_fraction': 0.75, 'bagging_freq': 5, 'boosting_type': 'gbdt', 'colsample_bytree': 0.75, 'learning_rate': 0.2,
'metric': 'rmse', 'n_jobs': -1, 'objective': 'regression', 'seed': 236} 0.5598691345021387
```

In [11]:
```
from lightgbm import LGBMRegressor
from sklearn.metrics import mean_squared_error
hyperparameters = {
'boosting_type': ['gbdt'],
'metric': ['rmse'],
'objective': ['regression'],
'n_jobs': [-1],
'seed':[ 236],
'learning_rate': [0.2],
'bagging_fraction': [0.75],
'bagging_freq': [ 5],
'colsample_bytree':[ 0.75]}

import lightgbm as lgb

# # clf = LGBMRegressor(hyperparameters)
# model = clf.fit(x_train[col], y_train)
train_set = lgb.Dataset(x_train[col], y_train)
val_set = lgb.Dataset(x_val[col], y_val)

model = lgb.train(hyperparameters, train_set, num_boost_round = 2500, early_stopping_rounds =200, valid_sets = [train_set,
val_pred = model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f' rmse score is {val_score}')
y_pred = model.predict(test[col])
test['demand'] = y_pred
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 4.385325 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1700
[LightGBM] [Info] Number of data points in the train set: 57473650, number of used features: 23
[LightGBM] [Info] Start training from score 1.122458
Training until validation scores don't improve for 200 rounds
[100]	training's rmse: 2.42718	valid_1's rmse: 2.30946
[200]	training's rmse: 2.38217	valid_1's rmse: 2.35073
Early stopping, best iteration is:
[44]	training's rmse: 2.48082	valid_1's rmse: 2.27593
 rmse score is 5.179855518187406
```

In [12]:

```
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
(60980, 29)
```

Out[12]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.892216	0.648964	0.541736	0.499136	0.446502	0.599287	0.963917	0.932590	1.136494	...	0.878585	1.273115
1	HOBBIES_1_002_CA_1_evaluation	0.533759	0.533759	0.521366	0.455940	0.432987	0.406253	0.419053	0.366243	0.317825	...	0.304757	0.370495
2	HOBBIES_1_003_CA_1_evaluation	0.542841	0.477415	0.489808	0.555234	0.495246	0.544175	0.580427	0.585628	0.542739	...	0.810626	0.830537
3	HOBBIES_1_004_CA_1_evaluation	2.398092	2.114076	1.832743	1.705512	2.039744	1.381708	1.458651	1.756863	1.881938	...	1.560355	1.256201
4	HOBBIES_1_005_CA_1_evaluation	1.167763	0.928069	1.171550	1.627729	1.448893	1.296523	1.596973	1.516384	1.747969	...	1.269351	1.095965

5 rows × 29 columns

In [13]:

```
final.to_csv('submission.csv', index = False)
```

In [14]:

```
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "lightgbm with best parameter"
```

```
100%|████████████████████████████████| 20.4M/20.4M [00:02<00:00, 9.28MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

submission.csv 0.86946 5.44561

13 hours ago by Namratesh Shrivastav

lightgbm with best parameter

Submission

14 Model -2 XG Boost

In [25]:
```
import xgboost as xgb
xgb_model = xgb.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, learning_rate=0.01, seed = 123)
xgb_model.fit(x_train[col], y_train, verbose = True, eval_metric=mean_squared_error )
```

```
[07:44:15] WARNING: ../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor of reg:squarederror.
```

Out[25]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.01, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=10, n_jobs=8, num_parallel_tree=1,
             objective='reg:linear', random_state=123, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, seed=123, subsample=1,
             tree_method='approx', validate_parameters=1, verbosity=None)
```

In [26]:
```
val_pred = xgb_model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f'Our val rmse score is {val_score}')
```

```
Our val rmse score is 13.258448600769043
```

In [30]:
```
y_pred = xgb_model.predict(test[col])
test['demand'] = y_pred
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
(60980, 29)
```

Out[30]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	...	0.502423	0.502423
1	HOBBIES_1_002_CA_1_evaluation	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	...	0.502423	0.502423
2	HOBBIES_1_003_CA_1_evaluation	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	...	0.502423	0.502423
3	HOBBIES_1_004_CA_1_evaluation	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	...	0.502423	0.502423
4	HOBBIES_1_005_CA_1_evaluation	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	0.502423	...	0.502423	0.502423

5 rows × 29 columns

In [31]:
```
final.to_csv('submission.csv', index = False)
```

In [32]:
```
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "xgboost_with_no_extra_features-2"
```

```
100%|██████████| 20.2M/20.2M [00:02<00:00, 9.32MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

Submission and Description	Private Score	Public Score	Use for Final Score
submission.csv 8 minutes ago by Namratesh Shrivastav xgboost_with_no_extra_features-2	3.36050	5.44561	☐

14.1 Xg boost with interpolate sell price

In [18]:

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
xgb_model = xgb.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, learning_rate=0.01, seed = 123)
xgb_model.fit(x_train[col], y_train, verbose = True, eval_metric=mean_squared_error )

val_pred = xgb_model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f'Our val rmse score is {val_score}')


y_pred = xgb_model.predict(test[col])
test['demand'] = y_pred
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
[18:02:40] WARNING: ../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor of reg:squarederror.
Our val rmse score is 13.223123550415039
(60980, 29)
```

Out[18]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	...	0.501493	0.501493
1	HOBBIES_1_002_CA_1_evaluation	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	...	0.501493	0.501493
2	HOBBIES_1_003_CA_1_evaluation	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	...	0.501493	0.501493
3	HOBBIES_1_004_CA_1_evaluation	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	...	0.501493	0.501493
4	HOBBIES_1_005_CA_1_evaluation	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	0.501493	...	0.501493	0.501493

5 rows × 29 columns

In [19]:
```
final.to_csv('submission.csv', index = False)
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "xgboost_with_sell_price interpolate method"
```

```
100%|██████████| 20.6M/20.6M [00:02<00:00, 10.1MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

Submission and Description	Private Score	Public Score	Use for
submission.csv 15 minutes ago by Namratesh Shrivastav xgboost_with_sell_price interpolate method	3.39226	5.44561	☐

15 Model 3 - Adaboost

In [14]:
```
# col = [ 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id',
#        'wm_yr_wk', 'event_name_1', 'event_type_1',
#        'event_name_2', 'event_type_2', 'snap_CA', 'snap_TX', 'snap_WI',
#        'sell_price', 'year', 'month', 'day', 'week']
```

In [14]:
```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error
regr = AdaBoostRegressor(random_state=0, n_estimators=100)
regr.fit(x_train[col], y_train)
val_pred = regr.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f'rmse score is {val_score}')
```

```
rmse score is 8.115268548276887
```

In [15]:
```python
y_pred = regr.predict(test[col])
test['demand'] = y_pred
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
(60980, 29)
```

Out[15]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.711637	0.635612	0.635612	0.625639	0.625639	0.635612	1.050950	1.212913	1.060924	...	0.653187	0.742537
1	HOBBIES_1_002_CA_1_evaluation	0.625639	0.625639	0.625639	0.625639	0.625639	0.625639	0.625639	0.625639	0.625639	...	0.625639	0.625639
2	HOBBIES_1_003_CA_1_evaluation	0.645586	0.645586	0.645586	0.645586	0.645586	0.645586	0.645586	0.645586	0.645586	...	0.680736	0.663161
3	HOBBIES_1_004_CA_1_evaluation	1.271363	0.742537	0.742537	0.742537	0.742537	0.742537	1.060924	1.638225	1.476262	...	0.742537	0.742537
4	HOBBIES_1_005_CA_1_evaluation	1.078499	1.078499	1.078499	1.476262	1.476262	1.476262	1.885462	1.885462	1.885462	...	1.476262	1.476262

5 rows × 29 columns

In [16]:
```python
final.to_csv('submission.csv', index = False)
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "adaboost"
```

```
100%|██████████| 20.2M/20.2M [00:01<00:00, 10.8MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

submission.csv	2.88930	5.44561
16 hours ago by Namratesh Shrivastav		
adaboost		

16 Model 4 - Stacking Classifier

In [25]:

```
from mlxtend.regressor import StackingRegressor
from lightgbm import LGBMRegressor
import xgboost as xgb
from sklearn.svm import SVR
```

In [33]:
```
X_train1 = x_train[col]
lightgbm = LGBMRegressor()
xgboost = xgb.XGBRegressor(objective ='reg:linear',
                  seed = 123)
adaboost = AdaBoostRegressor(random_state=0, n_estimators=100)


stregr = StackingRegressor(regressors=[xgboost, adaboost],  verbose=1,
                           meta_regressor=lightgbm)

stregr.fit(X_train1, y_train)
```

```
Fitting 2 regressors...
Fitting regressor1: xgbregressor (1/2)
[15:45:49] WARNING: ../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor of reg:squarederror.
Fitting regressor2: adaboostregressor (2/2)
```

Out[33]:
```
StackingRegressor(meta_regressor=LGBMRegressor(),
                  regressors=[XGBRegressor(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=None,
                                           gpu_id=None, importance_type='gain',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           monotone_constraints=None,
                                           n_estimators=100, n_jobs=None,
                                           num_parallel_tree=None,
                                           objective='reg:linear',
                                           random_state=None, reg_alpha=None,
                                           reg_lambda=None,
                                           scale_pos_weight=None, seed=123,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=None),
                              AdaBoostRegressor(n_estimators=100,
                                                random_state=0)],
                  verbose=1)
```

In [34]:
```python
val_pred = stregr.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f'rmse score is {val_score}')

y_pred = stregr.predict(test[col])
test['demand'] = y_pred
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
rmse score is 5.836832340786545
(60980, 29)
```

Out[34]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.997281	0.857308	0.701866	0.644614	0.545858	0.672189	0.976698	0.758348	0.987874	...	1.133470	1.453186
1	HOBBIES_1_002_CA_1_evaluation	0.557668	0.609799	0.543312	0.446093	0.409749	0.368794	0.266792	0.261198	0.255603	...	0.407271	0.403202
2	HOBBIES_1_003_CA_1_evaluation	0.604038	0.568409	0.573471	0.569873	0.463716	0.573890	0.505423	0.457756	0.458326	...	0.889656	0.842122
3	HOBBIES_1_004_CA_1_evaluation	2.347068	2.176047	1.965574	1.716206	2.076605	1.287876	1.321107	1.698113	1.705914	...	1.716206	1.372748
4	HOBBIES_1_005_CA_1_evaluation	1.247417	0.994520	1.344363	1.680866	1.341567	1.297814	1.636019	1.480663	1.695019	...	1.377441	1.230505

5 rows × 29 columns

In [35]:
```python
final.to_csv('submission.csv', index = False)
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "stacking classifier"
```

```
100%|██████████████████████████████| 20.4M/20.4M [00:02<00:00, 9.04MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```

submission.csv 13 hours ago by Namratesh Shrivastav stacking classifier	1.14915	5.44561

17 Lightgbm with no extra parameter

In [39]:
```python
col = [ 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id',
       'wm_yr_wk', 'event_name_1', 'event_type_1',
       'event_name_2', 'event_type_2', 'snap_CA', 'snap_TX', 'snap_WI',
       'sell_price', 'year', 'month', 'day', 'week']

import lightgbm as lgb
from sklearn.metrics import mean_squared_error
params = {
    'boosting_type': 'gbdt',
    'metric': 'rmse',
    'objective': 'regression',
    'n_jobs': -1,
    'learning_rate': 0.1,
    'bagging_fraction': 0.75,
    'bagging_freq': 10,
    'colsample_bytree': 0.75}

train_set = lgb.Dataset(x_train[col], y_train)
val_set = lgb.Dataset(x_val[col], y_val)

model = lgb.train(params, train_set, num_boost_round = 3000, early_stopping_rounds =300, valid_sets = [train_set, val_set]
val_pred = model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f' rmse score is {val_score}')
y_pred = model.predict(test[col])
test['demand'] = y_pred
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 2.150310 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 936
[LightGBM] [Info] Number of data points in the train set: 57473650, number of used features: 18
[LightGBM] [Info] Start training from score 1.122458
Training until validation scores don't improve for 300 rounds
[200]	training's rmse: 3.05827	valid_1's rmse: 2.91413
[400]	training's rmse: 2.93868	valid_1's rmse: 2.82293
[600]	training's rmse: 2.875	valid_1's rmse: 2.77138
[800]	training's rmse: 2.82498	valid_1's rmse: 2.72208
[1000]	training's rmse: 2.79557	valid_1's rmse: 2.70127
[1200]	training's rmse: 2.76194	valid_1's rmse: 2.6793
[1400]	training's rmse: 2.73493	valid_1's rmse: 2.6531
[1600]	training's rmse: 2.71825	valid_1's rmse: 2.63793
[1800]	training's rmse: 2.69933	valid_1's rmse: 2.62386
[2000]	training's rmse: 2.68343	valid_1's rmse: 2.6124
[2200]	training's rmse: 2.66907	valid_1's rmse: 2.6031
```

```
[2400]	training's rmse: 2.6583	valid_1's rmse: 2.59324
[2600]	training's rmse: 2.64633	valid_1's rmse: 2.59335
[2800]	training's rmse: 2.6358	valid_1's rmse: 2.58318
[3000]	training's rmse: 2.62517	valid_1's rmse: 2.58438
Did not meet early stopping. Best iteration is:
[3000]	training's rmse: 2.62517	valid_1's rmse: 2.58438
 rmse score is 6.679020651590686
```

In [40]:

```python
val_pred = model.predict(x_val[col])
val_score = mean_squared_error(val_pred, y_val)
print(f'rmse score is {val_score}')

y_pred = model.predict(test[col])
test['demand'] = y_pred
predictions = test[['id', 'date', 'demand']]
predictions = predictions.reset_index().pivot_table( index = 'id', columns = 'date', values = 'demand')
columns = ['F' + str(i + 1) for i in range(28)]
predictions.columns = columns
evaluation_rows = [row for row in sample_sub['id'] if 'validation' in row]
evaluation = sample_sub[sample_sub['id'].isin(evaluation_rows)]

validation = sample_sub[['id']].merge(predictions, on = 'id')
final = pd.concat([validation, evaluation])
print(final.shape)
final.head()
```

```
rmse score is 6.679020651590686
(60980, 29)
```

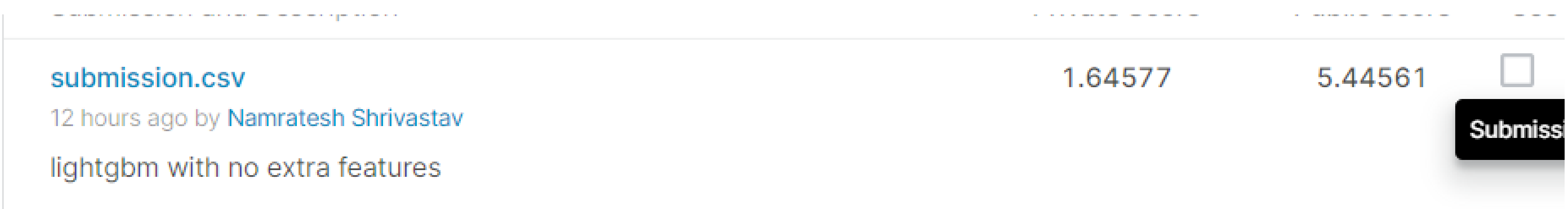
Out[40]:

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20
0	HOBBIES_1_001_CA_1_evaluation	0.757751	0.812425	0.808088	0.800299	0.713540	0.771310	0.819166	0.749804	0.907100	...	0.931898	0.706437
1	HOBBIES_1_002_CA_1_evaluation	0.909529	0.964203	0.959866	0.952077	0.865318	0.939841	0.961266	0.891084	1.048380	...	1.073999	0.848317
2	HOBBIES_1_003_CA_1_evaluation	0.839592	0.904347	0.900010	0.892220	0.805462	0.868758	0.869431	0.826649	0.961283	...	0.980701	0.777681
3	HOBBIES_1_004_CA_1_evaluation	1.794925	1.849599	1.845262	1.837472	1.750713	1.808483	1.848683	1.778500	1.935796	...	1.961415	1.735734
4	HOBBIES_1_005_CA_1_evaluation	1.105758	1.153832	1.149495	1.141706	1.054947	1.118243	1.158250	1.090669	1.250102	...	1.269519	1.041701

5 rows × 29 columns

In [42]:
```
final.to_csv('submission.csv', index = False)
!kaggle competitions submit -c m5-forecasting-accuracy -f submission.csv -m "lightgbm with no extra features"
```

```
100%|██████████████████████████████████████| 20.3M/20.3M [00:11<00:00, 1.86MB/s]
Successfully submitted to M5 Forecasting - Accuracy
```



In [11]:
```
from tabulate import tabulate
l = [["lightgbm ", 0.86946,  5.44561],
     ["lightgbm with best parameter", 0.86572,  5.44561],
     ["xgboost", 2.80085,5.44561],
     ["xgboost with no extra features", 2.80085, 5.44561],
     ["xgboost_with_sell_price interpolate method", 3.39226, 5.44561],
      ["xgboost_with_sell_price fill 0 ", 3.36050, 5.44561],
       ["adaboost ", 2.88930, 5.44561],
       ["stacking classifier " ,1.14915 , 5.44561],
       ["lightgbm with no extra features ", 1.64577, 5.44561],


    ]
table = tabulate(l, headers=["Algorithm", "Private Score", "Public Score"], tablefmt='rst')
print(table)
```

executed in 7ms, finished 13:16:51 2021-04-06

```
==========================================  ===============  ==============
Algorithm                                     Private Score    Public Score
==========================================  ===============  ==============
lightgbm                                            0.86946         5.44561
lightgbm with best parameter                        0.86572         5.44561
xgboost                                             2.80085         5.44561
xgboost with no extra features                      2.80085         5.44561
xgboost_with_sell_price interpolate method          3.39226         5.44561
xgboost_with_sell_price fill 0                      3.3605          5.44561
adaboost                                            2.8893          5.44561
stacking classifier                                 1.14915         5.44561
lightgbm with no extra features                     1.64577         5.44561
==========================================  ===============  ==============
```

17.0.1 Observation

LightGBM works well in every cases

```
In [ ]:
```