

University of Maryland, College Park  
Dept of Mechanical Engineering

**Predicting Future sales, Kaggle Competition**

**Nishanth Vanipenta**

**UID: 116409605**



**Submitted to Dr. Nikhil Chopra**

## Table of Contents

<b>Abstract:</b> .....	<b>3</b>
<b>Introduction:</b> .....	<b>3</b>
<b>Exploratory Data Analysis:</b> .....	<b>3</b>
<b>Feature engineering:</b> .....	<b>5</b>
1. <b>Aggregate:</b> .....	<b>5</b>
2. <b>Generating Lag Features:</b> .....	<b>5</b>
3. <b>Item Prices and Discount rate:</b> .....	<b>6</b>
<b>Data preparation:</b> .....	<b>6</b>
<b>Modelling:</b> .....	<b>7</b>
<b>Model 1:</b> .....	<b>7</b>
<b>Model 2:</b> .....	<b>8</b>
<b>Model 3:</b> .....	<b>10</b>
<b>Improvement:</b> .....	<b>11</b>
<b>Generalization Bound:</b> .....	<b>11</b>
<b>ScoreCard :</b> .....	<b>12</b>
<b>Conclusion:</b> .....	<b>12</b>
<b>Learning Outcomes:</b> .....	<b>12</b>
<b>References:</b> .....	<b>12</b>

## Abstract:

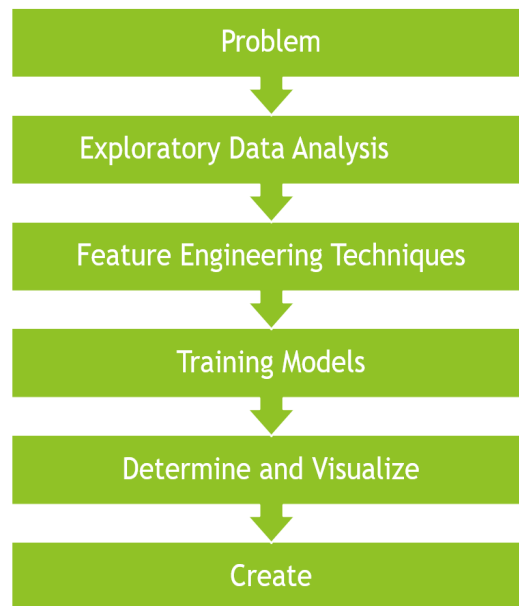
In this project, we propose our approach to future sales prediction based on the data analysis, Feature engineering techniques using the most conventional machine learning model as the baseline model entitled 'Linear regression' and compared with the gradient boosting algorithms namely LIGHTBM and XGBoost algorithms. It turned out to be, XGBoost outperformed other two models and the metric score that's observed to be 1.103.

## Introduction:

Predict future sales is a Kaggle competition. In this project a challenging time series data set was provided consisting of the sales data which is provided by one of the largest Russian firms- 1C company. The objective of this project is to predict the outcome of the sales for each product in the store based on the corresponding item\_id.

The main steps for predicting future sales include the Exploratory data analysis, Feature engineering techniques which include aggregate, lag features, data visualization and feature creation. Once, the data is prepared we split the data into the training and validation data. Then upon the selected modelling approaches we decide on selecting the best fit algorithm and predict the future sales based on it.

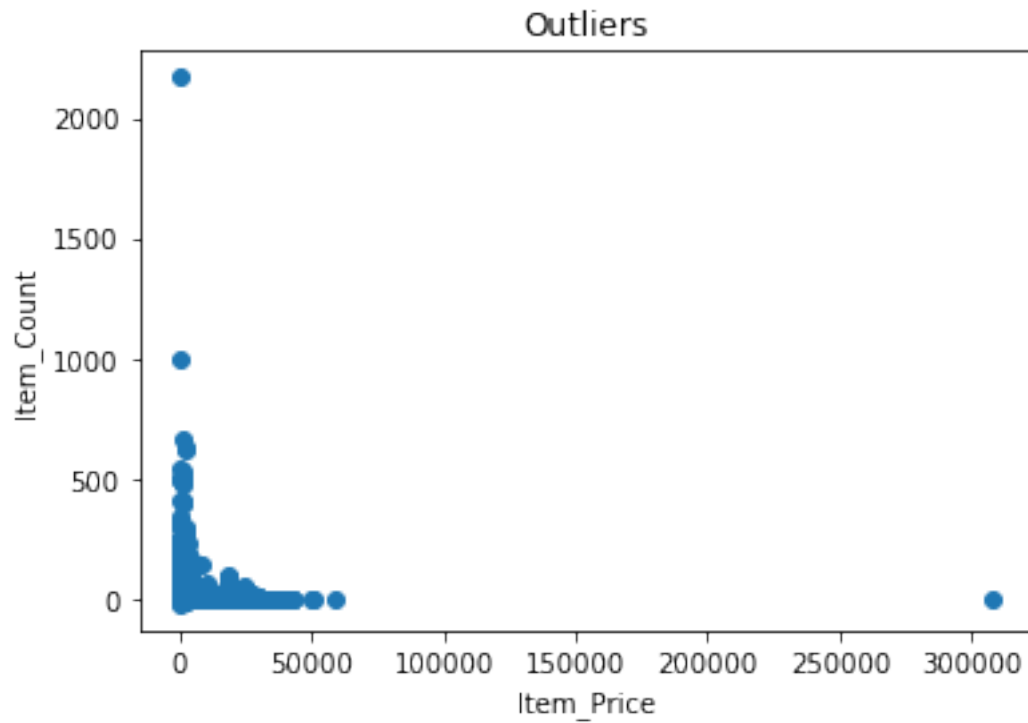
Here's the method i.e., followed for the project.



## Exploratory Data Analysis:

This is the first step followed in the project. In this method we analyse the data by exploring through the data sets. Initially, we check the train and test data and compare the shapes, I.e, if all the necessary data that's in the train data is in test data and viceversa. If this isn't the case, we add the missing values in the test and train data as zeros. Upon preprocessing we compare if there are any missing values and outliers

in the data set. From the visualization it is observed that the item\_price is too high for a particular item and later it is observed that the value is present in the data set and verified it's not a outlier.



Here, the data is grouped by each month after training and testing phase and merged later into the training data. Here's how head of the training data

```
out[15]:
```

	date_block_num	shop_id	item_id	date	item_price	item_cnt_day
0	0	5	5037	NaN	nan	0.000
1	0	5	5320	NaN	nan	0.000
2	0	5	5233	NaN	nan	0.000
3	0	5	5232	NaN	nan	0.000
4	0	5	5268	NaN	nan	0.000
...	...	...	...	...	...	...
7907065	33	45	18454	30.10.2015	99.000	1.000
7907066	33	45	16188	NaN	nan	0.000
7907067	33	45	15757	NaN	nan	0.000
7907068	33	45	19648	NaN	nan	0.000
7907069	33	45	969	NaN	nan	0.000

7757668 rows × 6 columns

## Feature engineering:

### 1. Aggregate:

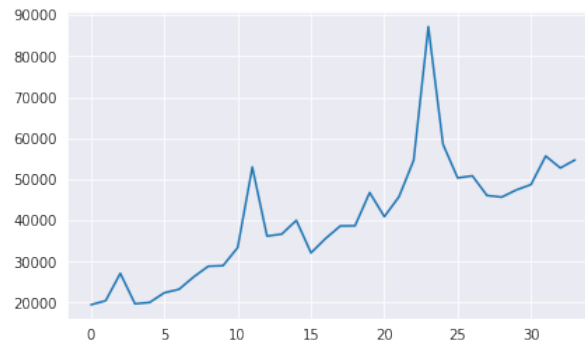
As the future prediction is per each month, we need to consider all the shop\_ids, item\_ids, item\_cnt\_month and take the aggregate and sum of the data for each month. To add more features and to make dataset robust I've also included the std\_deviation features for all the above data sets provided. Finally, merge the data into the data set. Here's the overview of how my dataset looks after classifying with shop and item.

	cnt_mean_shop	cnt_med_shop	cnt_std_shop
count	214,200.000	214,200.000	214,200.000
mean	0.188	0.054	0.381
std	0.608	0.509	0.773
min	0.000	0.000	0.000
25%	0.000	0.000	0.000
50%	0.029	0.000	0.171
75%	0.147	0.000	0.431
max	20.000	20.000	10.055

Out[17]:

	shop_id	item_id	cnt_mean_shop	cnt_med_shop	cnt_std_shop	order_mean_shop
0	2	30	0.118	0.000	0.327	1.000
1	2	31	0.235	0.000	0.741	1.088
2	2	32	0.324	0.000	0.638	1.088
3	2	33	0.324	0.000	0.535	1.029
4	2	38	0.000	0.000	0.000	1.000

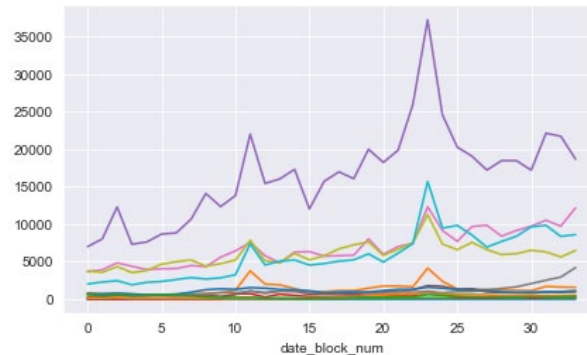
### 2. Generating Lag Features:



From the above graph it's evident that the itemsales for each month are varying for each month. This clearly suggest that the, item sales for the date\_block\_num ranging from 20-25 has a uptrend when compared with others. Which by means suggests that there is huge sale due to holidays or there's a lot of discount during that period due to the festival. Hence, to make the graph stationary we are ommitting the sales before 2014 because there would be no lag features before 2014 as ther's a 12 month lag. Here the graph would be stationary through out the considered months.

nt_rate	city	cnt_mean_shop	cnt_med_shop	cnt_std_shop	order_mean_shop	cnt_mean_shop_cat	cnt_prev	...	order_prev12
0.000	0	0.118	0.000	0.327	1.000	0.039	0.000	...	1.000
0.000	0	0.235	0.000	0.741	1.088	0.039	0.000	...	1.000
0.659	0	0.324	0.000	0.638	1.088	0.039	0.000	...	1.000
0.601	0	0.324	0.000	0.535	1.029	0.039	1.000	...	1.000
0.000	0	0.000	0.000	0.000	1.000	0.039	0.000	...	1.000

Sales for each item\_category for each month,



### 3. Item Prices and Discount rate:

Based on the item\_prices mean and the discount rate the the mean for each month based on the shop and item were calculated and these are meaningfully merged onto the above data set.

### Data preparation:

After the feature engineering techniques the data was merged onto the original train data that consist of the mean, standard deviation, order mean, lag features and discount rates and item prices mean for each month. The final data set consists of all the above features.

Upon merging all the above features and removing the repetitive parameters. The final data set consist of 25 columns and 4712400 rows with a final shape of ( 4712400, 24)

```

In [41]: train_set.shape
Out[41]: (4712400, 25)

In [33]: X_train.shape
Out[33]: (4712400, 24)

In [34]: test_set = test.copy()
test_set['date_block_num'] = 34

test_set = pd.merge(test_set, test_price_a, on=['shop_id', 'item_id'], how='left')
test_set = mergeFeature(test_set)
test_set['item_order'] = test_set['cnt_ema_s_prev'] #order_prev
test_set.loc[test_set['item_order'] == 0, 'item_order'] = 1

X_test = test_set.drop(['cnt_ema_s_prev'])

```

From the train data we take separate the X\_train and Y\_train. Where X\_train acts as the input and Y\_train acts as the output for the corresponding input. Here the Y\_train a column vector containing item\_counts.

Data	Shape
All_train	( 4712400, 25)
X_train	( 4712400, 24)
Y_train	( 4712400, 1)
Test set	( 214200, 25)
X_test	(214200, 24)
Y_test	(214200, 1)

Using the inbuilt sklearn model **train\_test\_split()**, the train data is itself split into the train and validation data.

## Modelling:

### Model 1:

#### BaselineModel – Linear regression

1. Data was split into Train and Validation data.
2. From sklearn import mean squared error and Linear regression.
3. Consider the metric as Root mean squared error ( RMSE)
4. Now, predict the train and validation values for the given train and val models.
5. Observe the insample error and choose the model
6. Save the files and submit the prediction. Check the out of sample error for the submission.

#### Prediction with Linear Regression

```
In [61]:
lr = LinearRegression()
lr.fit(trn_x, trn_y)

preds_trn = lr.predict(trn_x)
preds_val = lr.predict(val_x)

print('RMSE_TRAIN:', np.sqrt(mean_squared_error(trn_y, preds_trn)))
print('RMSE_VAL:', np.sqrt(mean_squared_error(val_y, preds_val)))

preds_test = lr.predict(X_test).clip(0,20)

linear_reg_df = pd.DataFrame()
linear_reg_df['ID'] = test['ID']

linear_reg_df['item_cnt_month'] = preds_test

linear_reg_df.to_csv('linear_submission.csv', index=False)

del linear_reg_df

RMSE_TRAIN: 0.3660696
RMSE_VAL: 0.36980003
```

For this model, The final RMSE scored that's observed is 1.2448 and the rank observed was of 4681.

Number	Value
Error in sample	0.3660696
Error out of sample	1.2488
Error_Validation	0.3698003
Overall Standing	4681

## Model 2:

### LightBGM:

LightBGM is one of the gradient boosting Machine learning model. LightBGM will train a Gradient Boosted Decision Tree. Basically, lightBGM work under the influence of hyper parameters. I've trained this model for 3000 estimators. Initially, we set LGB parameters as a dict and import the LGBMRegressor by sending the hyper parameters and the iterations into it.

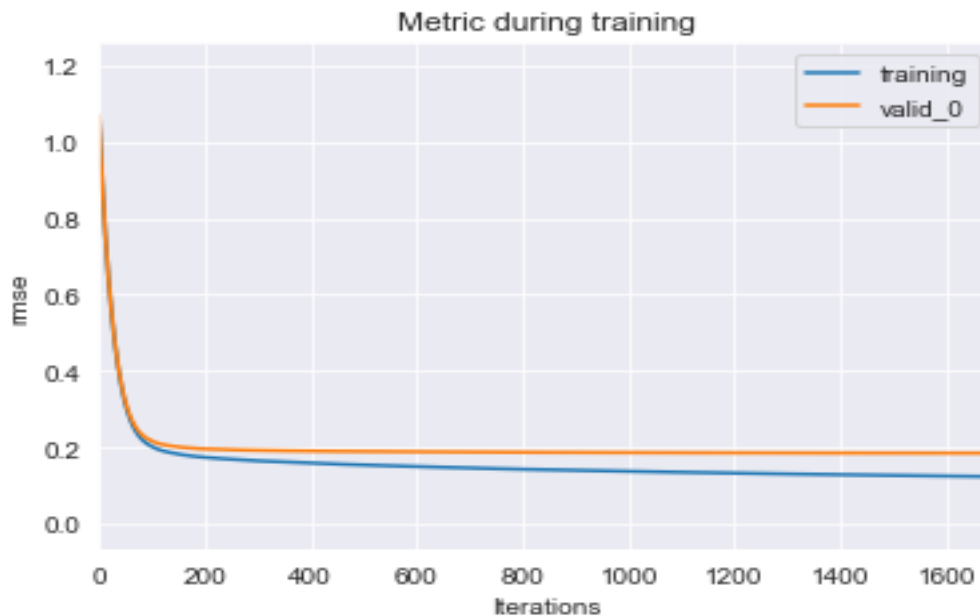
Params for the Lightbgm are training parameters, trainset is training data, num\_boost\_round is number of boosting iterations, valid sets is data to be evaluated during training and early stopping rounds is the max rounds number where validation score needs to improve to let the training continues. Therefore, if validation score does not improve after early stopping rounds, then training stops.

After tuning initially with the default values for the parameters, I've observed a Estimated out of sample error to be 1.1995. After changing the learning rate to 0.03 and number of estimates. There wasn't significant change in the error in sample.

The Error in sample and Error in validation converged after 200 iterations and the validation curves for the same has been plotted below.

Number	Value
Error in sample	0.366
Error out of sample	1.1918
Error_Validation	0.2134
Overall Standing	4829





### Implementation:

The hyper parameter tuning was done first by setting up the initial parameters to default parameters with learning rate 0.01 and tried various learning rates with 0.05, 0.1. For the particular learning rate due to high data, the computational power of PC wasn't sufficient and would crash in the middle. Observed almost similar values of the Errors for different learning rates.

### Model2- Lightgbm

```
In [68]: from sklearn import linear_model, preprocessing
from sklearn.model_selection import GroupKFold
import lightgbm as lgb

params={'learning_rate': 0.03,
        'objective': 'regression',
        'metric': 'rmse',
        'num_leaves': 64,
        'verbose': 1,
        'random_state': 42,
        'bagging_fraction': 1,
        'feature_fraction': 1
        }

reg = lgb.LGBMRegressor(**params, n_estimators=3000)
reg.fit(trn_x, trn_y, eval_set=[(val_x, val_y), (trn_x, trn_y)], early_stopping_rounds=50, verbose=500)
# print(reg.evals_result_)
lgb.plot_metric(reg)

preds_test = reg.predict(X_test).clip(0,20)

lgb_reg_df = pd.DataFrame()
lgb_reg_df['ID'] = test['ID']

lgb_reg_df['item_cnt_month'] = preds_test

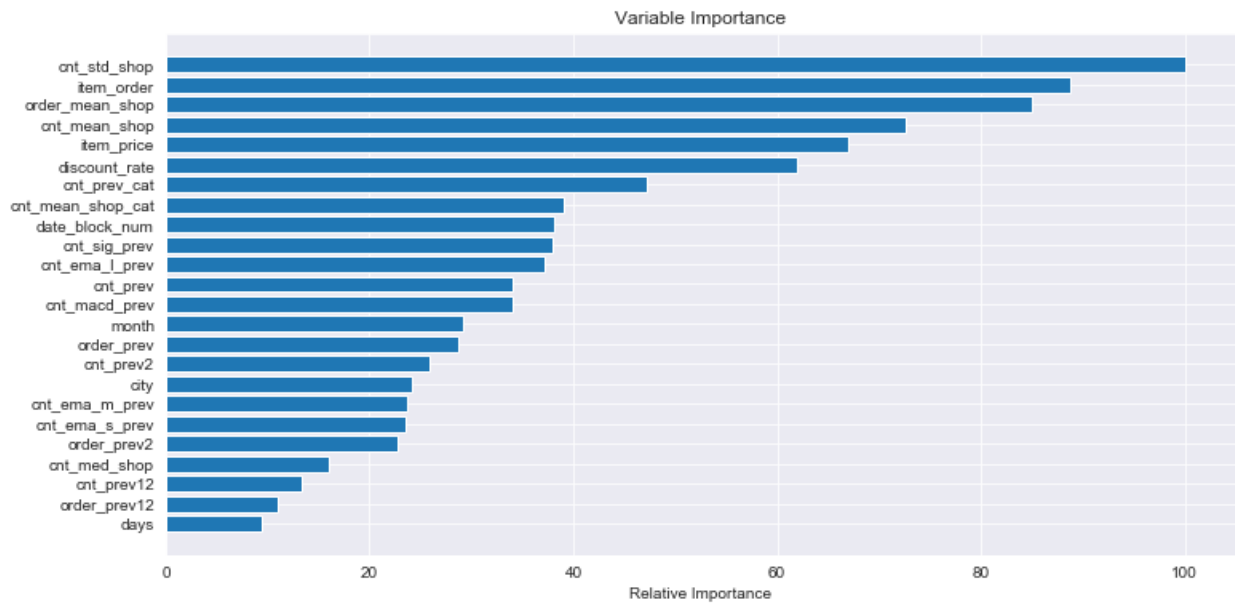
lgb_reg_df.to_csv('lgb_submission.csv', index=False)
del lgb_reg_df
```

```
Training until validation scores don't improve for 50 rounds
[500] training's rmse: 0.153645    valid_0's rmse: 0.189075
[1000] training's rmse: 0.137058    valid_0's rmse: 0.1861
[1500] training's rmse: 0.12641    valid_0's rmse: 0.184898
Early stopping, best iteration is:
[1637] training's rmse: 0.124007    valid_0's rmse: 0.184695
```

Metric during training

1.2

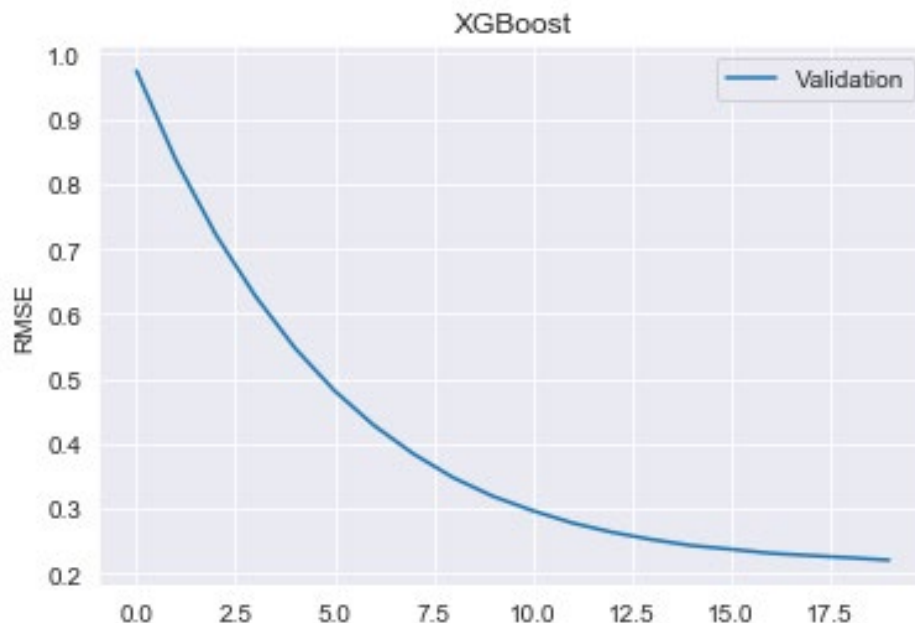
training



### Model 3: XGBOOST:

Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications [2]. I've implemented this expecting a better score when compared with other two algorithms.

I've used this method expecting it to preventing overfitting through Regularization. Also, it's observed that the, XGBoost employs that it's effectively find the optimal split points among the weighted data sets.



### Model 3- XGBoost

```
In [*]: import xgboost as xgb
from xgboost import XGBClassifier
from xgboost import XGBRegressor

Train = xgb.DMatrix(trn_X, trn_y)
Val = xgb.DMatrix(val_X, val_y)

model = XGBRegressor(
    max_depth=6,
    n_estimators=500,
    min_child_weight=30,
    colsample_bytree=0.8,
    subsample=0.8,
    eta=0.15,
    seed=42)

model.fit(
    trn_X,
    trn_y,
    eval_metric="rmse",
    eval_set=[(trn_X, trn_y), (val_X, val_y)],
    verbose=True,
    early_stopping_rounds = 10)

preds_test = model.predict(X_test).clip(0,20)

xg_boost = pd.DataFrame()
xg_boost['ID'] = test['ID']

lgb.plot_metric(model)

xg_boost['item_cnt_month'] = preds_test

xg_boost.to_csv('xgb_submission.csv',index=False)

fig, ax = pyplot.subplots()
ax.plot(x_axis, results['validation_0']['logloss'], label='Train')
ax.plot(x_axis, results['validation_1']['logloss'], label='Test')
ax.legend()
pyplot.ylabel('Log Loss')
pyplot.title('XGBoost Log Loss')
pyplot.show()

fig, ax = pyplot.subplots()
ax.plot(x_axis, results['validation_0']['error'], label='Train')
ax.plot(x_axis, results['validation_1']['error'], label='Test')
ax.legend()
pyplot.ylabel('Classification Error')
pyplot.title('XGBoost Classification Error')
pyplot.show()

[86] validation_0-rmse:0.18595 validation_1-rmse:0.19759
[87] validation_0-rmse:0.18582 validation_1-rmse:0.19757
[88] validation_0-rmse:0.18546 validation_1-rmse:0.19725
[89] validation_0-rmse:0.18540 validation_1-rmse:0.19724
[90] validation_0-rmse:0.18521 validation_1-rmse:0.19719
[91] validation_0-rmse:0.18512 validation_1-rmse:0.19719
```

Number	Value
Error in sample	0.2041
Error out of sample	1.10503
Error_Validation	0.1823
Overall Standing	3845

### Future work:

- Few things that can be improved were, to add more features Like holidays, including more lag features by taking the difference between current month and holiday month.
- Implement neural networks using deep learning frame works
- More Data preprocessing and preprocessing to be done.

### Generalization Bound:

Our main objective was to minimize  $E_{out}$ , and to minimize we must consider the epsilon and the bound to be small,


earlier with the previous bound, to get the **VC generalization bound**:

$$R(h) \leq R_{\text{emp}}(h) + \sqrt{\frac{8 \ln \Delta_{\mathcal{H}}(2m) + 8 \ln \frac{4}{\delta}}{m}}$$

or, by using the bound on growth function in terms of  $d_{\text{vc}}$  as:

$$R(h) \leq R_{\text{emp}}(h) + \sqrt{\frac{8d_{\text{vc}}(\ln \frac{2m}{d_{\text{vc}}} + 1) + 8 \ln \frac{4}{\delta}}{m}}$$

## ScoreCard :

3845	Maverick2020		1.10503	6	3h
<b>Your Best Entry ↑</b> Your submission scored 1.19831, which is not an improvement of your best score. Keep trying!					

## Conclusion:

Modelled three approaches to predict future sales, XGBoost model performed better when compare with that of the other two models. XGBoost model outperforms because it formulates a strong relation between item and consecutive months number for a given shop. To devise a better approach for strong model training mechanism, we need to generate more features and consider the regression model for each shop and item combination as well as the holiday features. Can be made more robust if neural networks are modelled using a deep learning models. Generated the excel files for the three models as the part of submission.

## Learning Outcomes:

- In this project I've learnt the exploration of data and enhanced my skills in leveraging large data
- Learnt the data analysis and explored the better feature engineering techniques
- Implemented the gradient boosting models and leveraged my skills in the field of data science.

## References:

1. (Morde, n.d.)
2. <https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d>
3. <https://jip.dev/notes/machine-learning/>
4. <https://www.kaggle.com/plasticgrammer/future-sales-prediction-playground>

