

TATA STEEL

SNTI



**DATA ANALYSIS
USING PYTHON**

Submitted by :

VT20223143

HIMANSHU KUMAR

INSTITUTE OF TECHNICAL EDUCATION

(B.TECH CSE)

INDEX

HEADING	PAGES
Acknowledgement	3
Introduction	4
Challenges of the problem	5
Tech Stack Details	6-7
Data Description	8-9
Exploratory Data Analysis	11-18
Data Cleaning	19
Data Aggregation	20-22
Modelling	23-25
Model Evaluating	26
Summary	27

ACKNOWLEDGEMENT

I, **Himanshu Kumar, VT20223143**, would like to express my gratitude to my guide, **Mr. Adepu Sai Preetham**, and my sponsor, for allowing me to work on this project on "**Predict future sales**" using **Python** and for all of their assistance and guidance that enabled me to complete this project successfully. Finally, I'd like to express my gratitude to Tata Steel, SNTI, for providing this interesting internship opportunity for students like me to get industrial experience systematically .

Mr. Adepu Sai Preetham

INTRODUCTION

It is an integral part of the business to make an intelligent business decision to save cost or gain more profit. Good forecasting can help to ensure the retailers maintain adequate inventory levels, mitigate the chance of stock obsolete, or supply the right product at the right time and location. For example, if the forecast indicates a 25% increase in sales of products or services, the store can purchase those products ahead to meet the demand. Conversely, a forecast of shortfalls in sales can allow people to mitigate the effect by taking actions ahead .


Predicting the future is one of the most relevant and challenging tasks in applied sciences. Building effective predictors from historical data demands computational and statistical methods for inferring dependencies between past and short-term future values of observed values as well as appropriate strategies to deal with longer horizons.

Challenges of the problem

Our project is Predict Future Sales, which is a Kaggle competition relating to time series prediction. In this competition the participants work with a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company. We are asked to predict total sales for every product and store in the next month. This project inscribes in this context, it will essentially focus on Development and Deployment of a forecasting model that relies on Machine Learning and Time-Series analysis techniques.

Building a good model from observed time series data is challenging because time series data are usually unstable and continuous. For instance, future sales can be affected by new products launch, promotions available, seasonality, and other changes that make it very difficult to predict using past behaviors. Furthermore, there is only a single historical sample available in each data instance, hence, we might need to add lag terms to perform the forecasting task.

Tech Stack Details

1.  (Python) **Python** is a [high-level](#), [general-purpose programming language](#). Its design philosophy emphasizes [code readability](#) with the use of [significant indentation](#). Libraries such as [NumPy](#), [SciPy](#), and [Matplotlib](#) allow the effective use of Python in scientific computing,



2. (Anaconda) Anaconda is an open-source distribution for python and R. It is used for [data science](#), [machine learning](#), [deep learning](#), etc. With the availability of more than 300 libraries for data science, it becomes fairly optimal for any programmer to work on anaconda for data science . Anaconda helps in simplified package management and deployment. Anaconda comes with a wide variety of tools to easily collect data from various sources using various machine learning and AI algorithms. It helps in getting an easily manageable environment setup which can deploy any project with the click of a single button.



3. **Microsoft Excel** is a [spreadsheet](#) developed by [Microsoft](#) for [Windows](#), [macOS](#), [Android](#) and [iOS](#). It features [calculation](#) or [computation](#) capabilities, graphing tools, [pivot tables](#), and a [macro](#) programming language called [Visual Basic for Applications](#) (VBA). Excel forms part of the [Microsoft Office](#) suite of software.

✔ Why Use Pandas 📄 🔍

Pandas incorporates two additional data structures into Python, namely Pandas Series and Pandas DataFrame.

These data structures allow us to work with labeled and relational data in an easy and intuitive manner

The main differences between NumPy ndarrays:

- 1- Series can hold many data types, such as dict or strings.
- 2- you can assign an index label to each element in the Pandas Series.

Just like with NumPy ndarrays, we can perform element-wise arithmetic

Pandas also allows us to only apply arithmetic operations on selected items

✔ Why Keras 📄 🔍

Keras makes coding deep neural networks simpler.

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models.

Keras was created to be user friendly, modular, easy to extend, and to work with Python

Just like with NumPy ndarrays, we can perform element-wise arithmetic

Given that the TensorFlow project has adopted Keras as the high-level API for TensorFlow 2.0 release

✔ Why Numpy? 📄 🔍

Has a number of key features that give it great advantages over Python lists. One such feature is speed.

This speed comes from the nature of NumPy arrays being memory-efficient and from optimized algorithms used by NumPy for doing arithmetic, statistical, and linear algebra operations.

It has multidimensional array data structures that can represent vectors and matrices easily.

Has a large number of optimized built-in mathematical functions.

Dataset Description

We were provided with daily historical sales data .The task is to anticipate the total quantity of products sold in each test group store. We note that the list of stores and products changes slightly every month. We have a set of data stored in the following files.

File Description :

- ❖ sales_train.csv: the training set. Daily historical data from January 2013 to October 2015.
- ❖ items.csv: supplemental information about the items/products.
- ❖ item_categories.csv: supplemental information about the items categories.
- ❖ test.csv: the test set. We need to forecast the sales for these shops and products for November 2015.
- ❖ shops.csv: supplemental information about the shops.
- ❖ sample_submission.csv - a sample submission file in the correct format

Features Description :

❖ *Independent variable*

- item_cnt_day - number of products sold. We are predicting a monthly amount of this measure.

❖ Dependent variable:

- ID - an Id that represents a (Shop, Item) tuple within the test set.
- shop_id - unique identifier of a shop.
- item_id - unique identifier of a product.
- item_category_id - unique identifier of item category
- item_price - current price of an item
- date - date in format dd/mm/yyyy

- date_block_num - a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1, ..., October 2015 is 33

- item_name - name of item

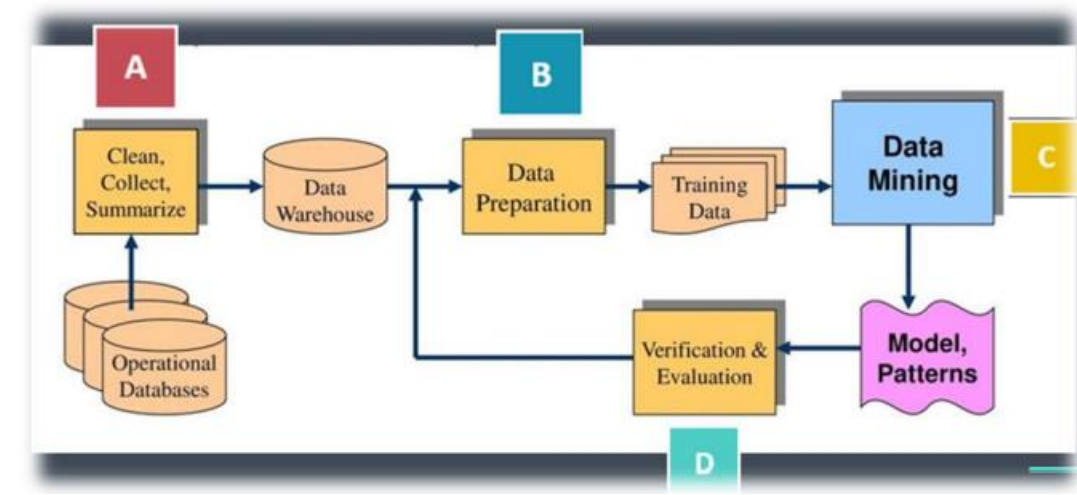
- shop_name - name of shop

- item_category_name - name of item category

Experimental Processes

Knowledge Discovery and Data Mining (KDD) is a multidisciplinary field focused on methodologies for extracting useful knowledge from data. The challenge of extracting knowledge from data draws upon research in statistics, databases, pattern recognition, machine learning, data visualization, optimization, and high-performance computing to provide advanced intelligence and useful solutions. The continuous rapid growth of data has created a huge need for KDD methodologies. The KDD process is classified in Four main stages which are:

- A. Clean, Collect, Summarize
- B. Data Preparation
- C. Data Mining
- D. Evaluation & Validation



Data Import :

```
train_data=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\sales_train.csv')
test_data=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\test.csv')
submission=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\sample_submission.csv')
items=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\items.csv')
items_category=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\item_categories.csv')
shops=pd.read_csv(r'D:\Datasets\competitive-data-science-predict-future-sales\shops.csv')
```

Exploratory Data Analysis :

Out[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
5	10.01.2013	0	25	2564	349.00	1.0
6	02.01.2013	0	25	2565	549.00	1.0
7	04.01.2013	0	25	2572	239.00	1.0
8	11.01.2013	0	25	2572	299.00	1.0
9	03.01.2013	0	25	2573	299.00	3.0

To take a closer look at the data took help of “.head()”function of pandas library which returns first five observations of the data set. Similarly “.tail()” returns last five observations of the data set.

```
In [5]: train_data.shape
```

Out[5]: (2935849, 6)

- Dataset comprises of 2935849 observations and 6 characteristics.
- Out of which one is dependent variable and rest 5 are dependent variables .

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
#   Column          Dtype
---  -
0   date            object
1   date_block_num  int64
2   shop_id         int64
3   item_id         int64
4   item_price      float64
5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB

```

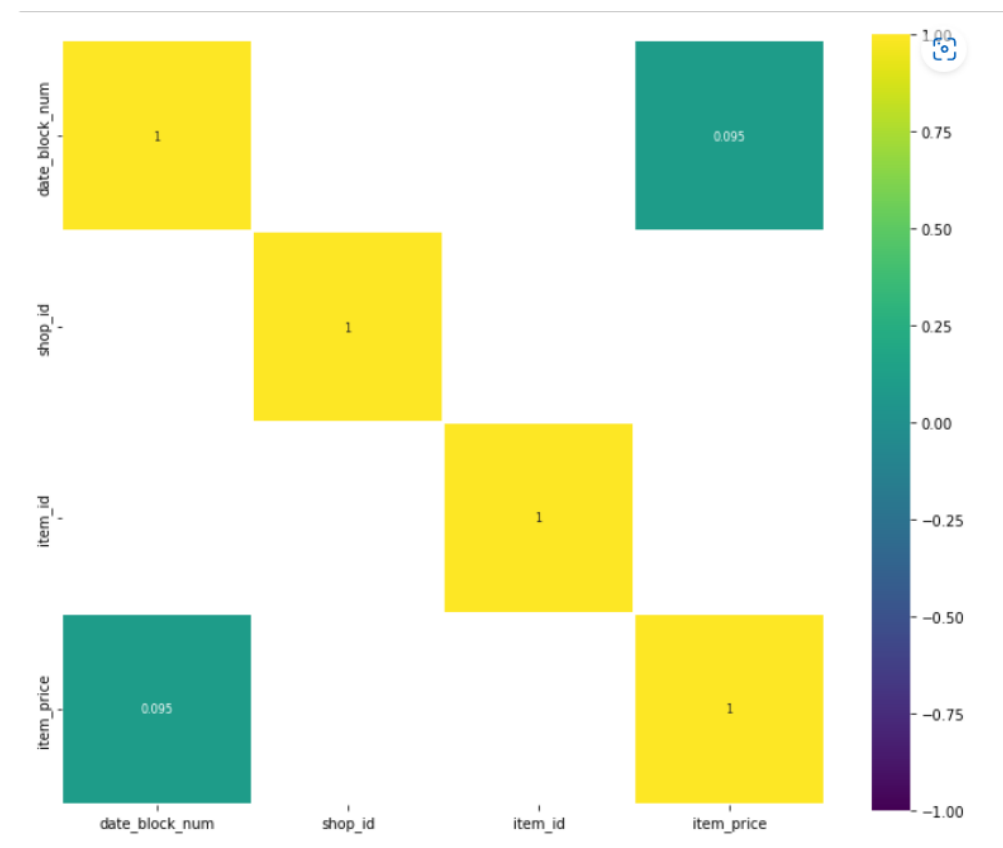
It is also a good practice to know the columns and their corresponding data types, along with finding whether they contain null values or not.

- Data has only object and integer values.
- No variable column has null/missing values.

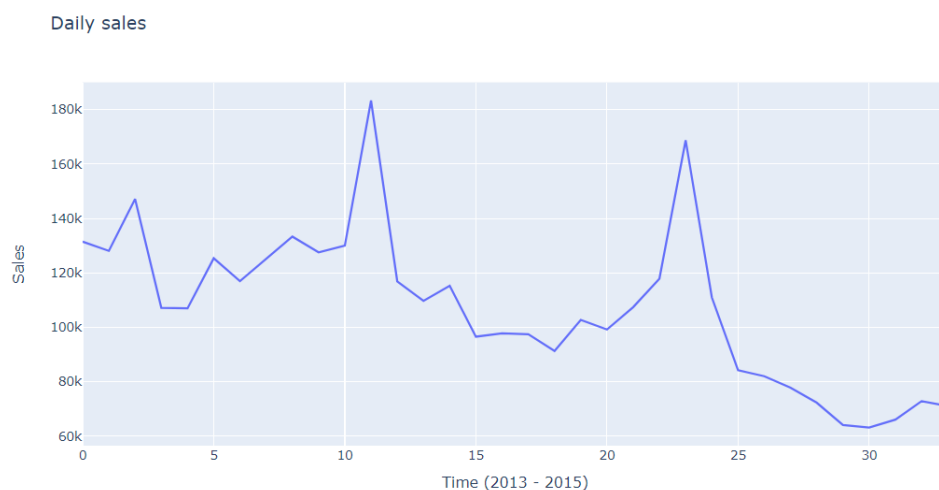
	date_block_num	shop_id	item_id	item_price	item_cnt_day
count	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06	2.935849e+06
mean	1.456991e+01	3.300173e+01	1.019723e+04	8.908532e+02	1.242641e+00
std	9.422988e+00	1.622697e+01	6.324297e+03	1.729800e+03	2.618834e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	-2.200000e+01
25%	7.000000e+00	2.200000e+01	4.476000e+03	2.490000e+02	1.000000e+00
50%	1.400000e+01	3.100000e+01	9.343000e+03	3.990000e+02	1.000000e+00
75%	2.300000e+01	4.700000e+01	1.568400e+04	9.990000e+02	1.000000e+00
max	3.300000e+01	5.900000e+01	2.216900e+04	3.079800e+05	2.169000e+03

The describe() function in pandas is very handy in getting various summary statistics. This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data.

- date_block_num, item price, and item_cnt_day have a high standard deviation because of the large difference between their minimum and maximum values. This is also the reason why outliers can be seen in item price and item count below.

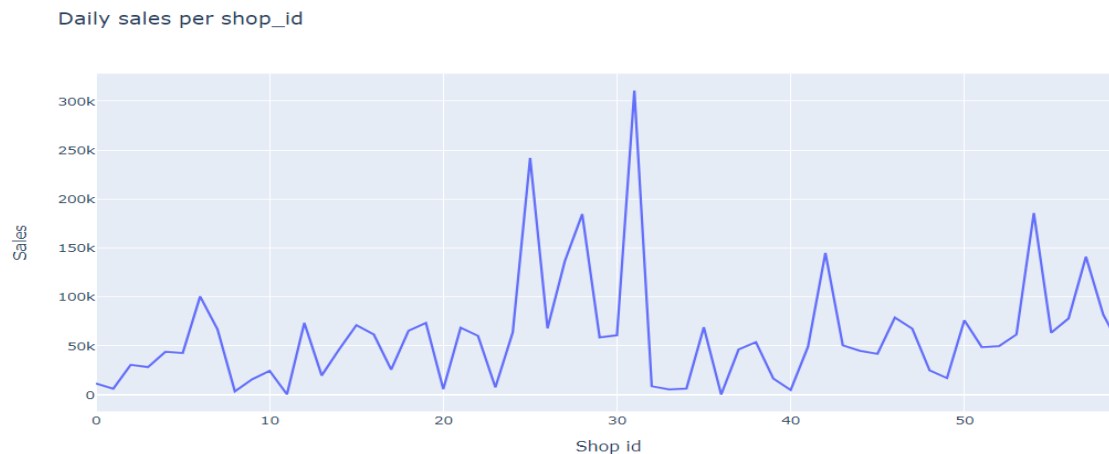


The above snippet tells us the correlation between different columns .



We use plotly library to plot sales v/s Time graph . In this code snippet we group train_data with 'date_block_num' and sum each 'item_cnt_day' . We observe that Generally, sales from shops show a downward trend towards 2015. Peaks in sales can be seen between the days 20 and 25 probably because this is the time that most customers receive their monthly salaries.

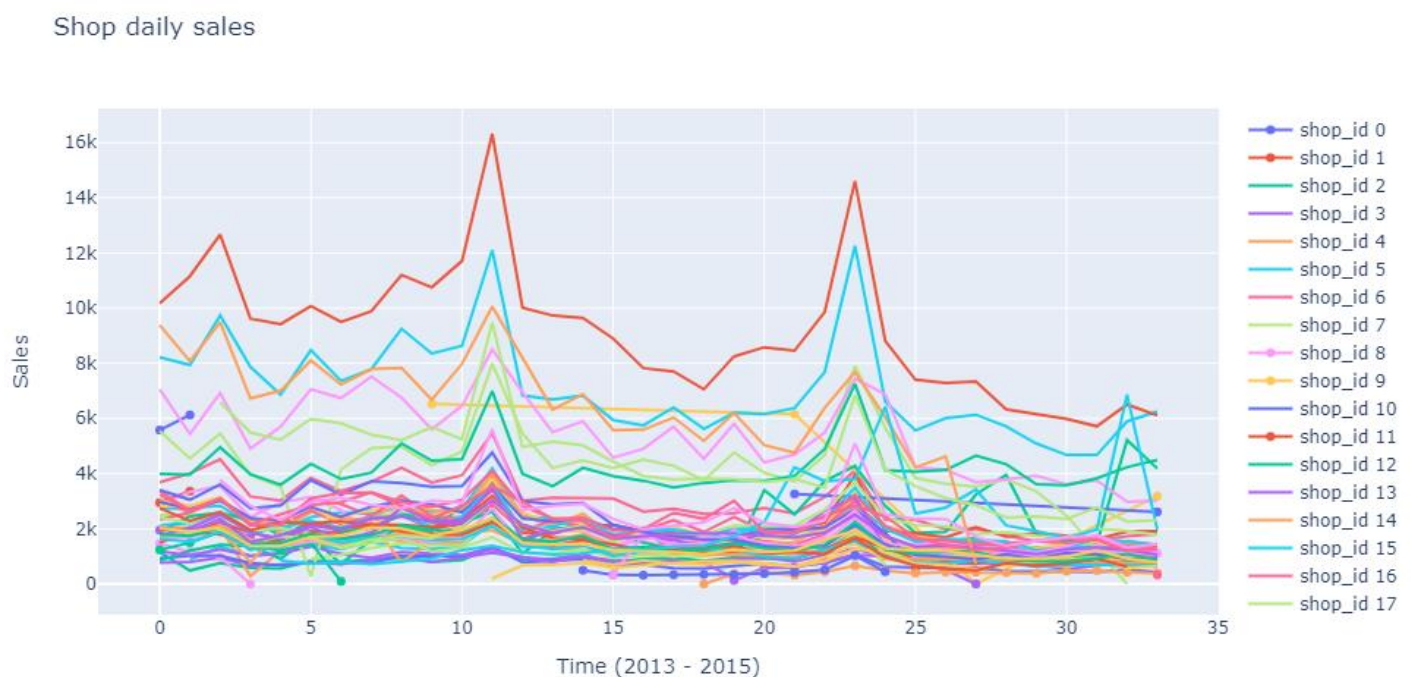
Another peak can be seen between the days 10 and 12 which could be influenced by special offers provided by shops.



We group data by shop_id and sum each item_cnt_day .

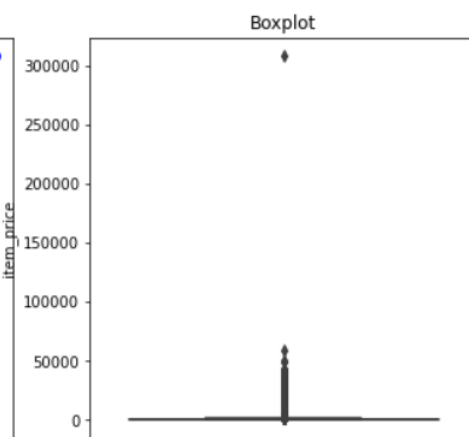
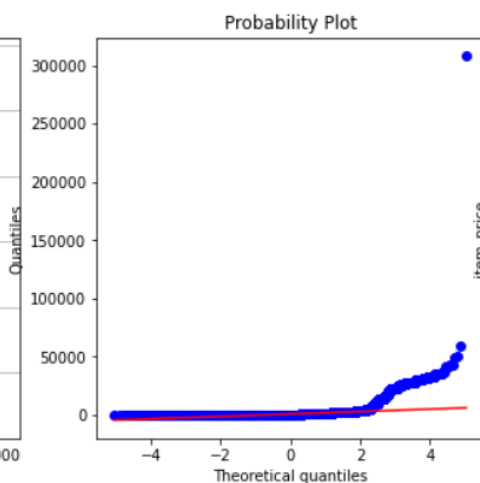
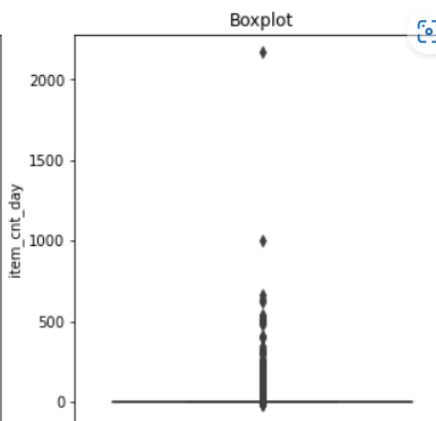
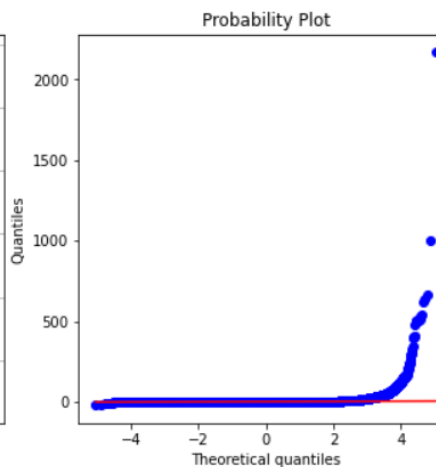
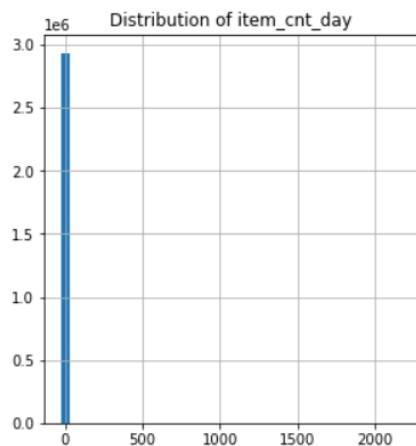
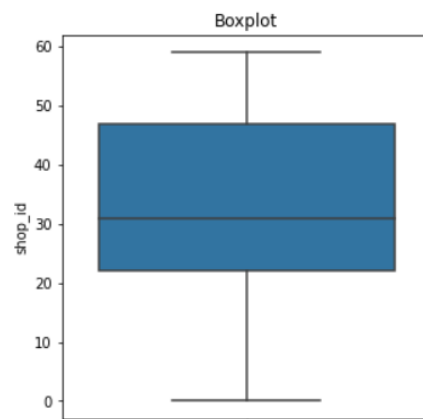
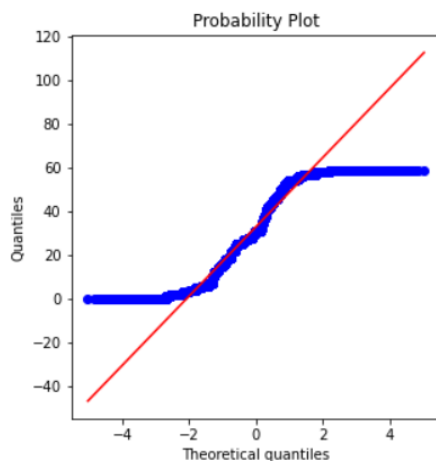
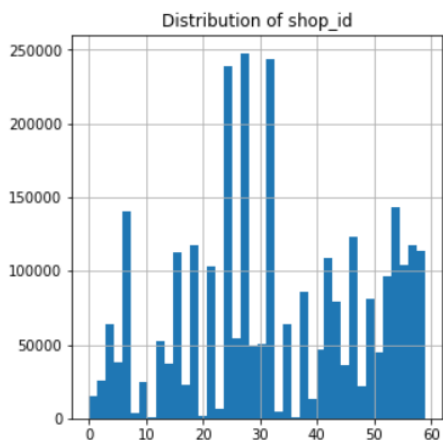
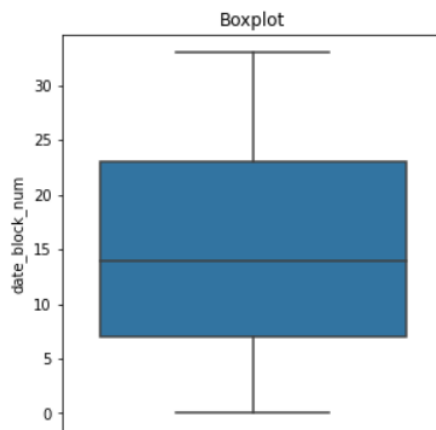
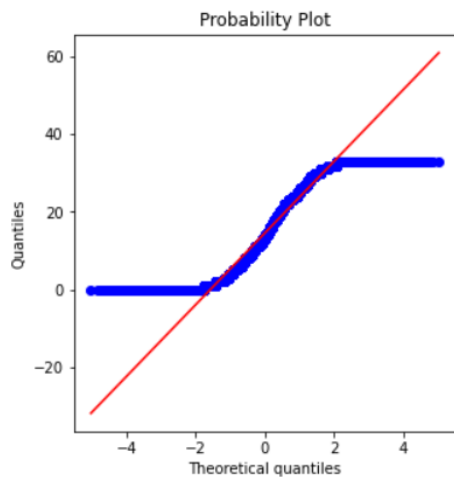
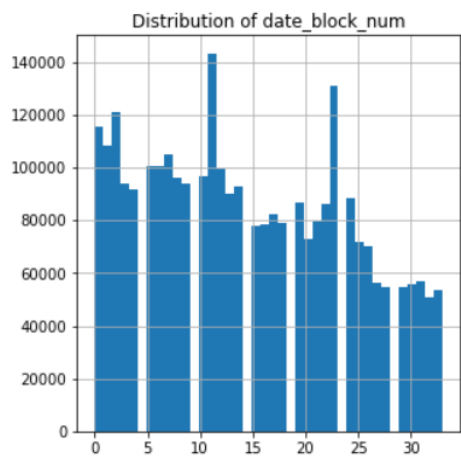
We observe that shop_id 31 has the highest number of sales .

we group by shop_id and date_block_num . Then for each unique shop_id a plot is drawn .



We define a dataframe with name 'data' which consist of only three column names('date_block_num','shop_id','item_price', 'item_cnt_day').

We then define a function 'num_plot' which has argument df and data . The num_plot function makes three subplot namely , **distribution plot** , **probability plot** and **boxplot**.

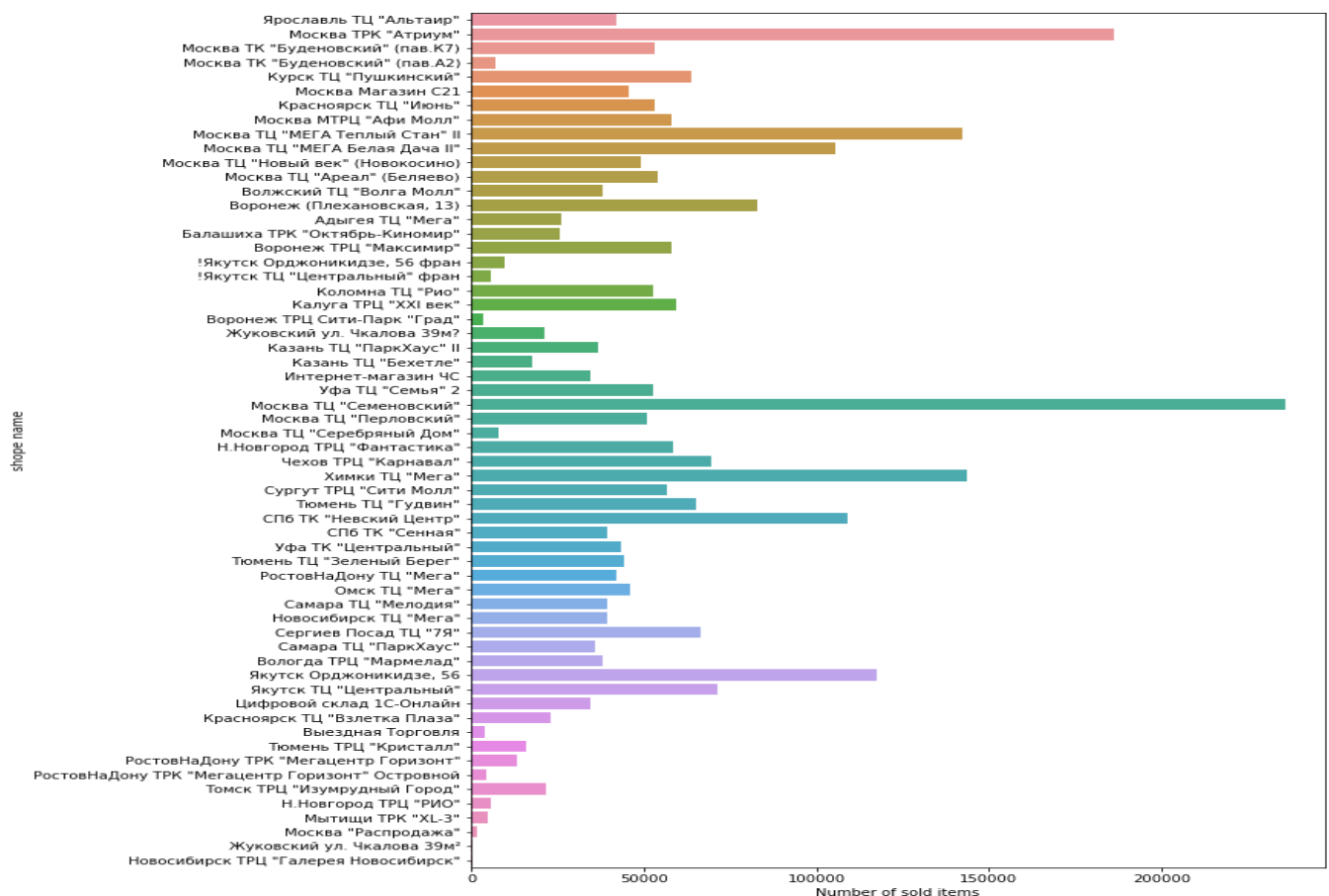


The probability plot is a graphical technique for assessing whether or not a data set follows a given distribution such as the normal or Weibull.

The data are plotted against a theoretical distribution in such a way that the points should form approximately a straight line. Departures from this straight line indicate departures from the specified distribution.

We merge different dataset together to form df .

```
In [14]: plt.figure(figsize = (10, 16))
sns.countplot(y=df['shop_name'])
plt.ylabel('shop name')
plt.xlabel('Number of sold items')
```



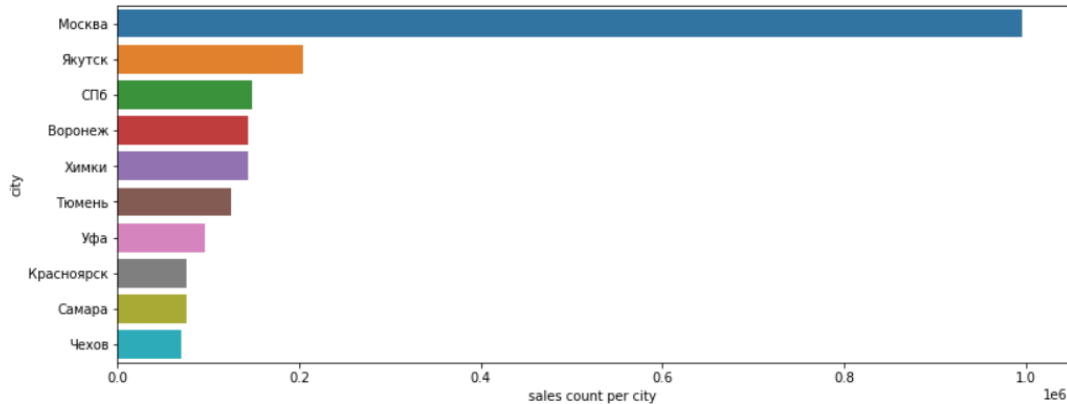
Seeing the above plot we get an idea of number of sales per shop.

```
Out[17]:
```

	city	count
0	Москва	996636
1	Якутск	204404
2	СПб	148535
3	Воронеж	144151
4	Химки	143480

Москва is the city with highest count. Lets represent this with graph.

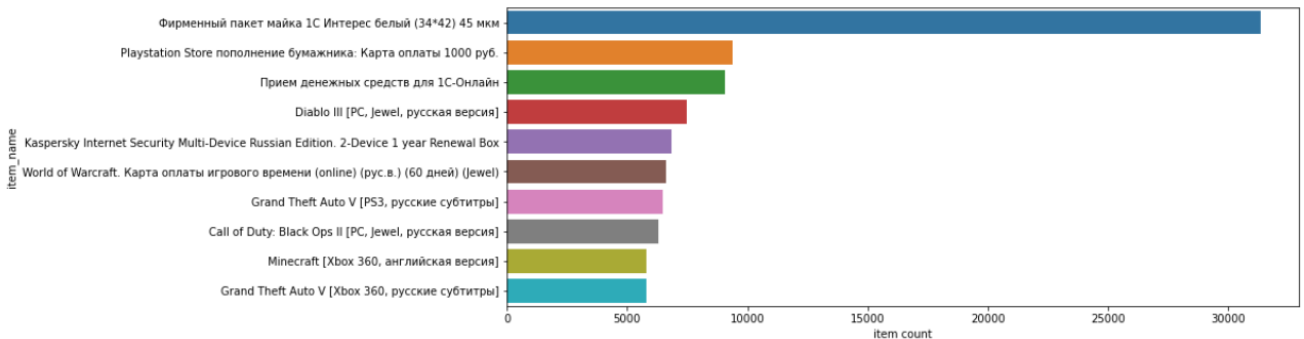
```
Out[18]: Text(0.5, 0, 'sales count per city')
```



```
Out[19]:
```

	item_name	count
0	Фирменный пакет майка 1С Интерес белый (34*42)...	31340
1	Playstation Store пополнение бумажника: Карта ...	9408
2	Прием денежных средств для 1С-Онлайн	9067
3	Diablo III [PC, Jewel, русская версия]	7479
4	Kaspersky Internet Security Multi-Device Russi...	6853
5	World of Warcraft. Карта оплаты игрового време...	6602
6	Grand Theft Auto V [PS3, русские субтитры]	6475
7	Call of Duty: Black Ops II [PC, Jewel, русская...	6320
8	Minecraft [Xbox 360, английская версия]	5811
9	Grand Theft Auto V [Xbox 360, русские субтитры]	5805

```
Out[20]: Text(0.5, 0, 'item count')
```

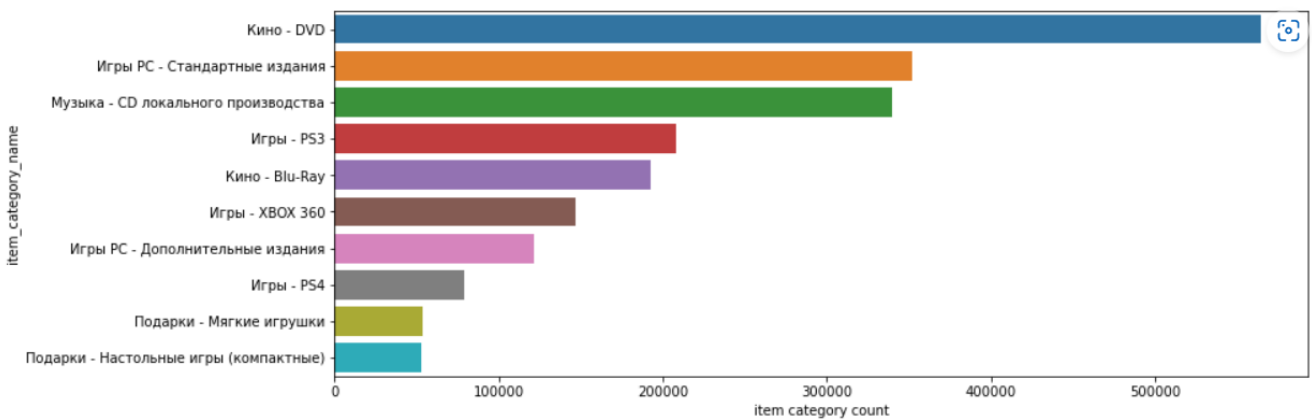


Thus, Фирменный пакет майка 1С Интерес белый (34*42) is the most bought after item.

Out[21]:

	item_category_name	count
0	Кино - DVD	564652
1	Игры PC - Стандартные издания	351591
2	Музыка - CD локального производства	339585
3	Игры - PS3	208219
4	Кино - Blu-Ray	192674
5	Игры - XBOX 360	146789
6	Игры PC - Дополнительные издания	121539
7	Игры - PS4	79058
8	Подарки - Мягкие игрушки	53845
9	Подарки - Настольные игры (компактные)	53227

Out[22]: Text(0.5, 0, 'item category count')



Кино - DVD category had the highest sells.

Summary of primary EDA :

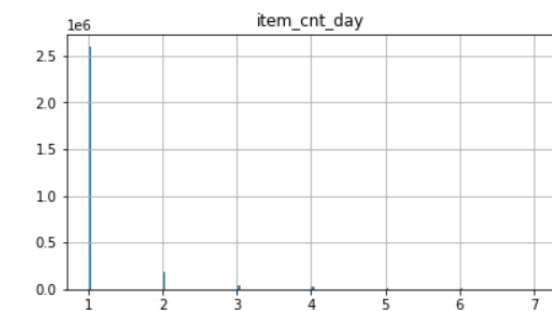
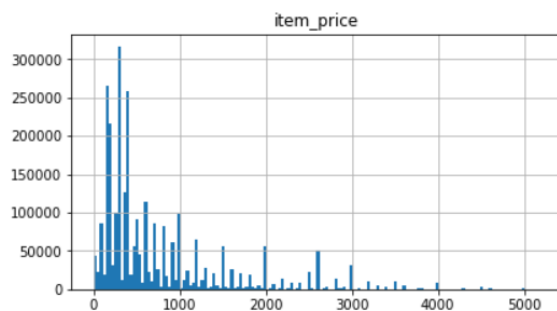
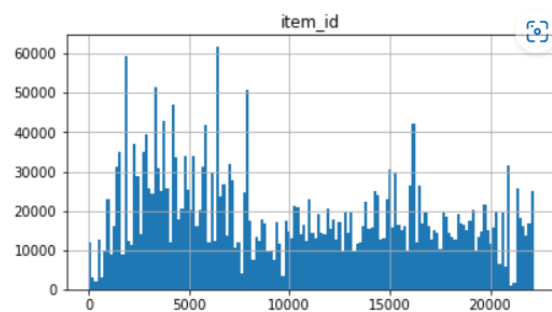
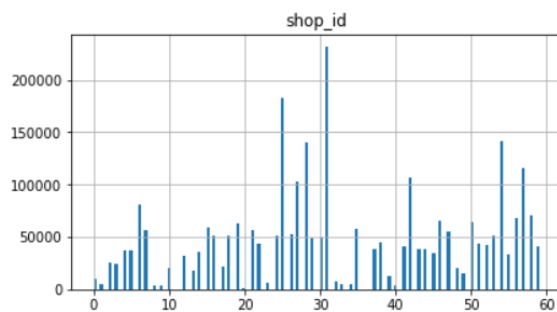
- Using `isnull().sum()` , we found that there are no null values.
- From `.describe()` , we found that there are negative values.
- Clearly from above histograms and boxplot , we see that `item_price` and `item_cnt_day` have outliers.

Data Cleaning :

Data cleaning, or data cleansing, is the important **process of correcting or removing incorrect, incomplete, or duplicate data within a dataset**. Data cleaning **should be the first step in the workflow**. When working with large datasets and combining various data sources, there's a strong possibility of duplicate or mislabel data. If one have inaccurate or incorrect data, it will lose its quality, and ML algorithm's outcomes become unreliable.

Objectives :

- Increase the reliability of overall dataset.
- Remove data items with higher standard deviation .
- Remove data items with negative value.



Data Aggregation :

Data aggregation refers to a process of collecting information from different sources and presenting it in a summarized format so that business analysts can perform statistical analyses of business schemes. The collected information may be gathered from various data sources to summarize these data sources into a draft for data analysis. This step is the major step taken by any business organization because the accuracy of insights from data analysis majorly depends on the quality of data they use. It is very necessary to collect quality content in huge amounts so that they can create relevant outcomes. Data aggregation plays a vital role in finance, product, operations, and marketing strategies in any business organization.

Aggregated data is present in the data warehouse that can enable one to solve various issues, which helps solve queries from data sets.

Objective :

- Create Monthly count
 - Merge Shop name with cities.
 - Group By "month", "date_block_num", "shop_id", "item_id", "item_price", "item_cnt_day"
 - Aggregate each of newly formed columns.

```
In [40]: dataframe.head(10)
```

```
Out[40]:
```

	shop_id	item_id	city	item_category_id	item_cnt_month																	
date_block_num					0	1	2	3	4	5	...	24	25	26	27	28	29	30	31	32	33	
0	0	30	0	40	NaN	22.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0	31	0	37	NaN	11.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	0	32	0	40	6.0	10.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	0	33	0	37	3.0	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	0	35	0	40	1.0	14.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5	0	36	0	37	NaN	1.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
6	0	40	0	57	NaN	1.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
7	0	42	0	57	NaN	1.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	0	43	0	40	1.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9	0	49	0	57	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

10 rows × 38 columns

Making test data and train data equal dimension:

We make test data equal size with train data by merging test data with items dataset and shopshop dataset. In item dataset we take 'item_id','item_category_id' and in shop dataset we take 'shop_id' and 'city' .

```
Out[83]:
```

	ID	shop_id	item_id	item_category_id	city
0	0	5	5037	19	4
1	1	5	5320	55	4
2	2	5	5233	19	4
3	3	5	5232	23	4
4	4	5	5268	20	4

We then, merge test_data with dataframe on 'shop_id','item_id','city','item_category_id' .

Dataframe looks like this ,

	ID	shop_id	item_id	item_category_id	time0	time1	time2	time3	time4	time5	...	time25	time26	time27	time28	time29	time30	time31	time32	time33
0	0	5	5037	19	4	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	1.0	1.0	1.0	3.0	1.0
1	1	5	5320	55	4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2	5	5233	19	4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	2.0	0.0	1.0	3.0
3	3	5	5232	23	4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	4	5	5268	20	4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Defining Lag :

Lag : The steps that are considered to shift the data backward in the time(sequence), called lag times or lags.

Why Lag is important ?

To transform a time series model to supervised ml problem lag is used.

Our df dataframe after adding sufficient lag .

	time0	time1	time2	time3	time4	time5	time6	time7	time8	time9	...	time25_lag7	time26_lag7	time27_lag7	time28_lag7	time29_lag7	time30_lag7	time
0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	1.0	
1	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
5	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	2.0	0.0	0.0	
6	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
7	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0
8	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	5.0
9	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

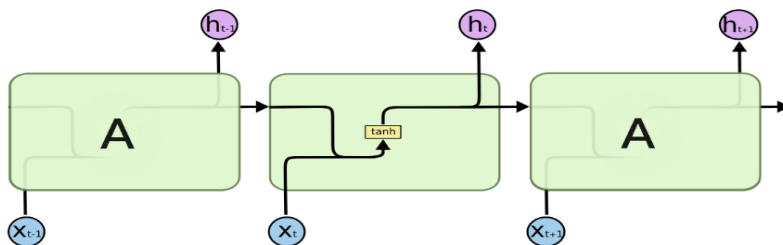
10 rows × 280 columns

Modelling :

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.

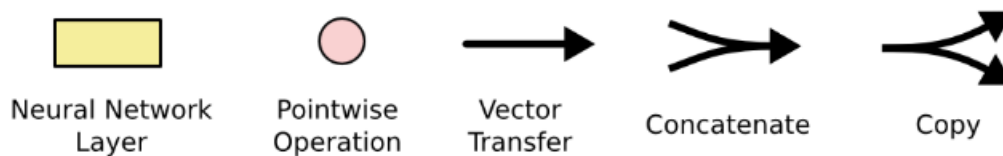
LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

Implementing LSTM

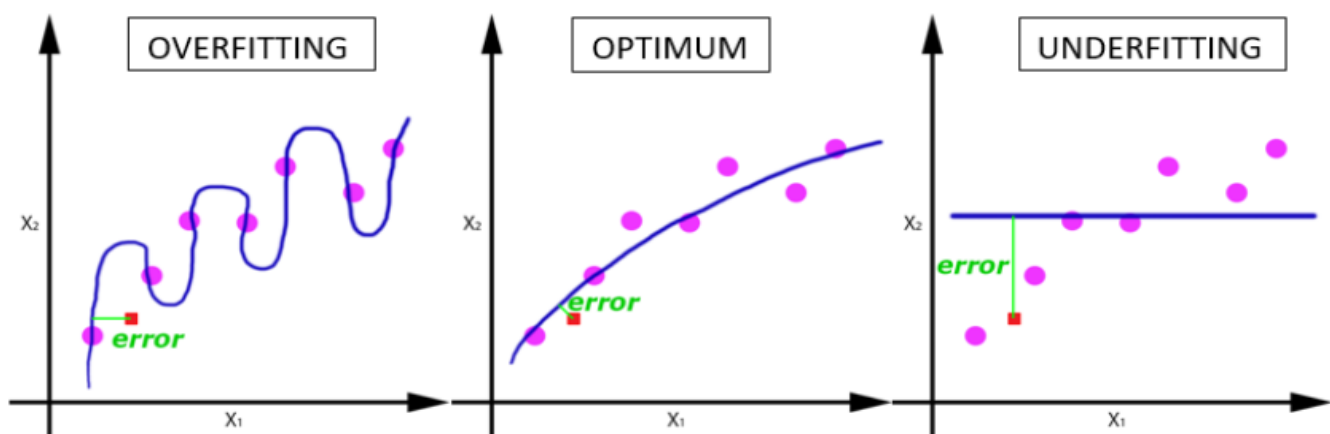
We use keras from tensorflow. Here to implement LSTM we first import Sequential() , then import layers ; LSTM, Dense , Dropout.

```
Epoch 1/15
53/53 [=====] - 85s 2s/step - loss: 1.7305 - mean_squared_error: 1.7305
Epoch 2/15
53/53 [=====] - 93s 2s/step - loss: 1.7018 - mean_squared_error: 1.7018
Epoch 3/15
53/53 [=====] - 94s 2s/step - loss: 1.7004 - mean_squared_error: 1.7004
Epoch 4/15
53/53 [=====] - 44s 836ms/step - loss: 1.6983 - mean_squared_error: 1.6983
Epoch 5/15
53/53 [=====] - 44s 823ms/step - loss: 1.7009 - mean_squared_error: 1.7009
Epoch 6/15
53/53 [=====] - 44s 834ms/step - loss: 1.6988 - mean_squared_error: 1.6988
Epoch 7/15
53/53 [=====] - 42s 797ms/step - loss: 1.6994 - mean_squared_error: 1.6994
Epoch 8/15
53/53 [=====] - 43s 815ms/step - loss: 1.6988 - mean_squared_error: 1.6988
Epoch 9/15
53/53 [=====] - 43s 809ms/step - loss: 1.6986 - mean_squared_error: 1.6986
```

Epochs :

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.

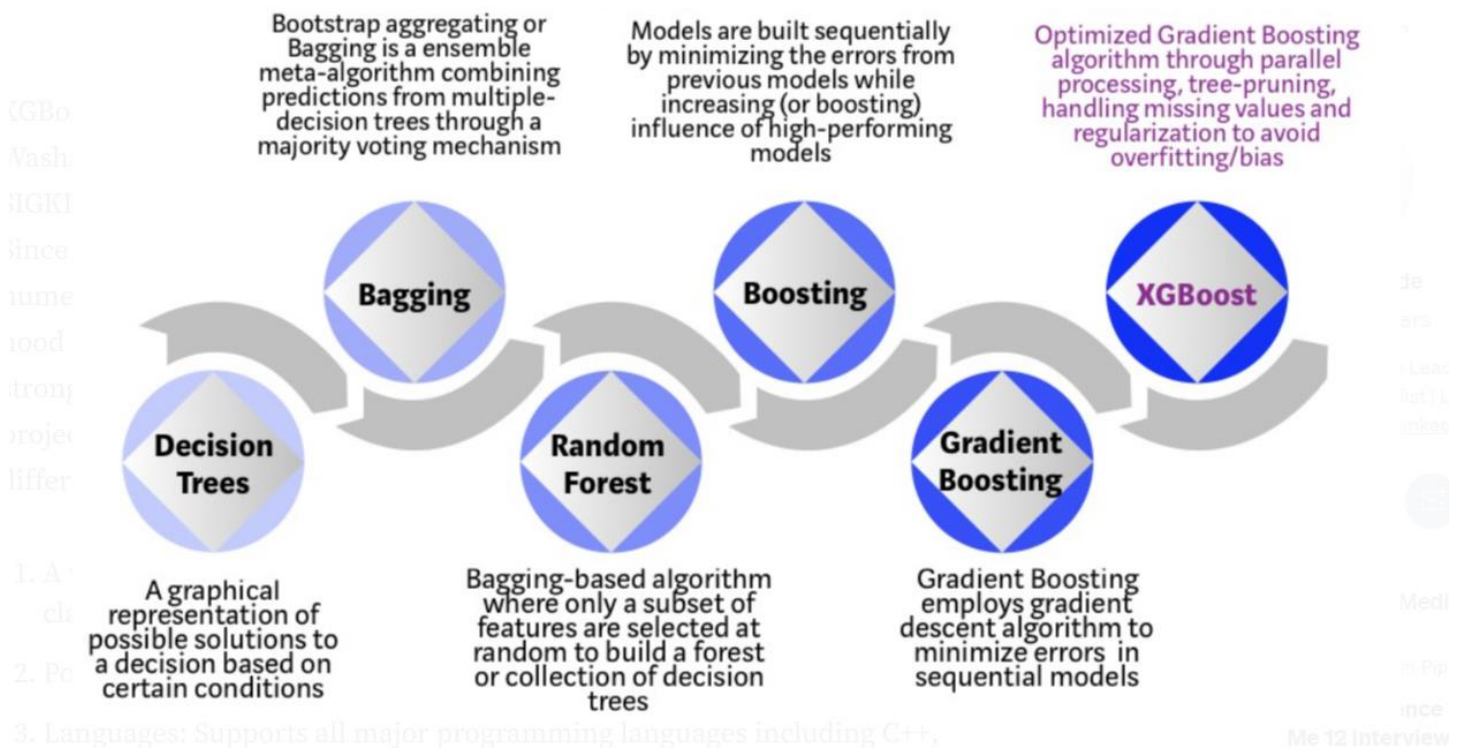
we need to pass the full dataset multiple times to the same neural network. But keep in mind that we are using a limited dataset and to optimise the learning and the graph we are using **Gradient Descent** which is an *iterative* process. So, *updating the weights with single pass or one epoch is not enough.*



One Epoch leads to underfitting. As the number of epochs increases, more number of times the weight are changed in the neural network and the curve goes from **underfitting** to **optimal** to **overfitting** curve.

XGBoost:

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.



Model Evaluating:

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ are predicted values

y_1, y_2, \dots, y_n are observed values

n is the number of observations

RMSE for LSTM is 1.20096

RMSE for XGBoost is 1.18939

Summary

This project has provided me with an excellent opportunity to study because it has taught me a lot about machine learning , exploratory data analysis, time series, and data preparation. Throughout the course of this project, I have encountered challenges and found a variety of solutions, which have given me new perspectives on the subjects.

This training helped me to improve various aspects of a students and a professionals life which are:

- Time management: how much of a wrok done everyday that counts as progress.
- Improve Self-learning: Through practical implementation of anything, an individual can learn anything faster and more efficiently.
- Improve my debugging skills
- Understand a practical approach to data analysis and implementation of

Neural network in a real world problem.

My theoretical understanding of machine learning was put into practice, and I was able to apply it in the actual world. This project taught me the value of self learning, problem-solving, debugging, and identifying the best practices to increase the efficiency of model and present data in more appeasing way. Additionally, working on this project gave me invaluable experience and understanding of the industry's operations, both of which were very helpful. It, in my opinion, played a crucial role in the advancement of my profession as a data scientist .

[Github Repository Link](#)