FUTURE SALES PREDICTION

Phase 5:project submission document

612721104041-Geetha A

Project Title: Future sales prediction

Phase5: Development part -2

Topic: continue building the IMDb score prediction model by:

- Feature engineering
- Model training
- Evaluation

Future sales prediction

Introduction:

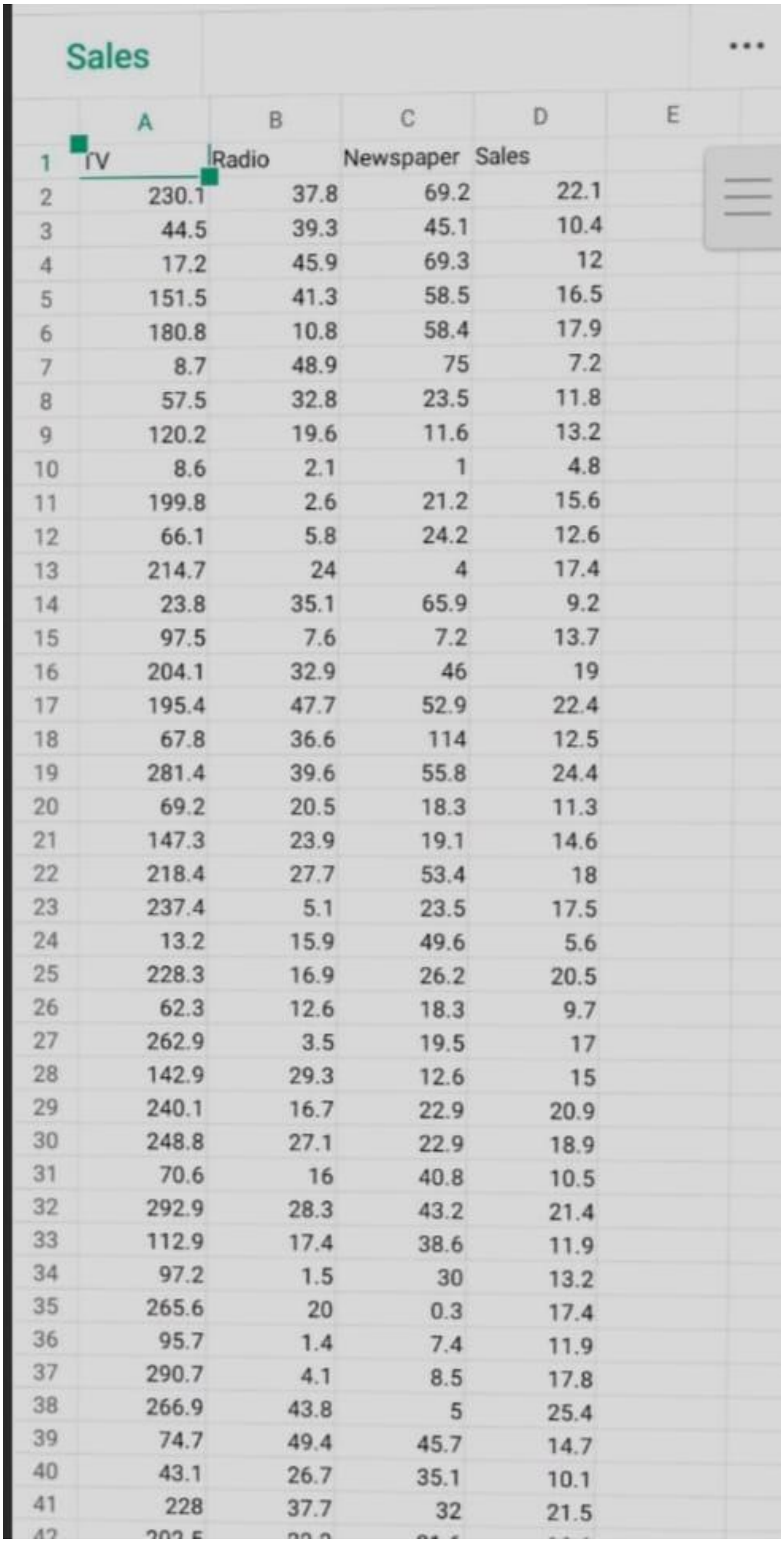
- Sales forecasting is the process of estimating future revenue by predicting how much of a product or service will sell in the next week, month, quarter, or year. At its simplest, a sales forecast is a projected measure of how a market will respond to a company's go-to-market efforts.
- Forecasts are about the future. It's hard to overstate how important it is for a company to produce an accurate sales forecast. Privately held companies gain confidence in their business when leaders can trust forecasts. For publicly traded companies, accurate forecasts confer credibility in the market.
- Sales forecasting adds value across an organization. Finance relies on forecasts to develop budgets for capacity plans and hiring, and production uses sales forecasts to plan their cycles. Forecasts help sales operations with territory and quota planning, supply chain with material purchases and production capacity, and sales strategy with channel and partner strategies.
- One of the most common methods used to predict sales is regression analysis. This method involves using historical sales data to train a model that can predict future sales. The model can take into account factors such as past sales, marketing campaigns, and economic indicators to make its prediction
- Sales forecast helps every business make better business decisions. It helps in overall business planning, budgeting, and risk management. Sales forecasting allows companies to efficiently allocate resources for future growth and manage its cash flow.

Sales prediction using in machine learning:

- Step 1: Identifying target and independent features. First, let's import Train. …
- Step 2: Cleaning the data set. First, we check for null values by running input. …
- Step 3: Exploratory Data Analysis. Descriptive Statistics. …
- Step 4: Building a model. ….
- Step 5: Check model accuracy. ….
- Step 6: Save the model into a pickle file.

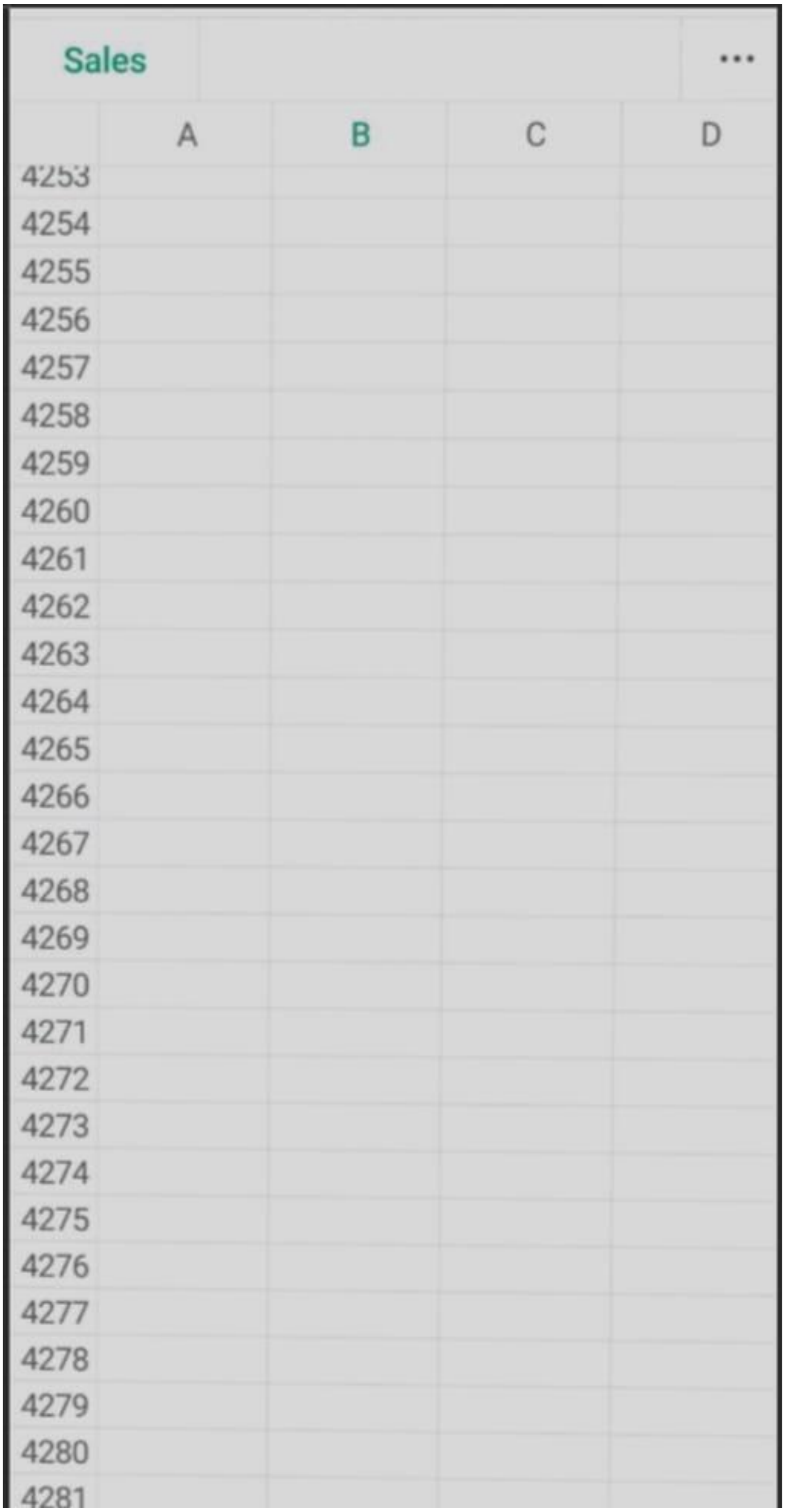
Given data set:

- One of the most common methods used to predict sales is regression analysis. This method involves using historical sales data to train a model that can predict future sales. The model can take into account factors such as past sales, marketing campaigns, and economic indicators to make its predictions



Sales

	A	B	C	D	E
1	TV	Radio	Newspaper	Sales	
2	230.1	37.8	69.2	22.1	
3	44.5	39.3	45.1	10.4	
4	17.2	45.9	69.3	12	
5	151.5	41.3	58.5	16.5	
6	180.8	10.8	58.4	17.9	
7	8.7	48.9	75	7.2	
8	57.5	32.8	23.5	11.8	
9	120.2	19.6	11.6	13.2	
10	8.6	2.1	1	4.8	
11	199.8	2.6	21.2	15.6	
12	66.1	5.8	24.2	12.6	
13	214.7	24	4	17.4	
14	23.8	35.1	65.9	9.2	
15	97.5	7.6	7.2	13.7	
16	204.1	32.9	46	19	
17	195.4	47.7	52.9	22.4	
18	67.8	36.6	114	12.5	
19	281.4	39.6	55.8	24.4	
20	69.2	20.5	18.3	11.3	
21	147.3	23.9	19.1	14.6	
22	218.4	27.7	53.4	18	
23	237.4	5.1	23.5	17.5	
24	13.2	15.9	49.6	5.6	
25	228.3	16.9	26.2	20.5	
26	62.3	12.6	18.3	9.7	
27	262.9	3.5	19.5	17	
28	142.9	29.3	12.6	15	
29	240.1	16.7	22.9	20.9	
30	248.8	27.1	22.9	18.9	
31	70.6	16	40.8	10.5	
32	292.9	28.3	43.2	21.4	
33	112.9	17.4	38.6	11.9	
34	97.2	1.5	30	13.2	
35	265.6	20	0.3	17.4	
36	95.7	1.4	7.4	11.9	
37	290.7	4.1	8.5	17.8	
38	266.9	43.8	5	25.4	
39	74.7	49.4	45.7	14.7	
40	43.1	26.7	35.1	10.1	
41	228	37.7	32	21.5	



Sales

•••

	A	B	C	D
4253				
4254				
4255				
4256				
4257				
4258				
4259				
4260				
4261				
4262				
4263				
4264				
4265				
4266				
4267				
4268				
4269				
4270				
4271				
4272				
4273				
4274				
4275				
4276				
4277				
4278				
4279				
4280				
4281				

Necessary step to follow :

1.Import Libraries :

Start by importing the necessary libraries.

Program :

```
Import pandas as pd

Import numpy as np

From sklearn.model_selection import train_test_split

From sklearn.preprocessing import StandardScaler
```

2.Load the dataset:

Load your dataset into a pandas dataframe. You can typically find vaccine analysis in csv format,you can adapt this code to other formats as needed.

Program :

```
Df=pd.read_csv('D:\world_vaccination.csv')

Pd.read()
```

3.Exporatory Data Analysis:

Perform EDA to understand your data better.This includes checking for missing values,exploring the data's statistics, and visualizing it to identify patterns.

Program:

```
#check for missing values

Print(df.isnull().sum())

#explore statistics

Print(df.describe())
```

```
#visualize the data (e.g.,histograms, scatter plots, etc,…)
```

4.Feature Engineering:

Depending on your dataset ,you may need to create new features or transform existing ones.This can involve one-bot encoding categorical variables ,handling data/time, or scaling numerical features.

Program :

```
#example:one-hot encoding for categorical variables.
Df = pd.get_dummies(df,coloumns=[‘Avg.total peoples vaccinated’,’ISO code’])
```

5.Split the Data:

Split your dataset into training and testing sets.This helps you evaluate your model’s performance later.

```
X = df.drop(‘iso code,axis=1)
Y= df[‘iso code’]
X_train,X test,Y_train,Y_test=train_test_split(X,Y,test_size=1,random_state=42)
```

6.Feature Scaling:

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean 0 and std=1) is a common choice.

Program:

```
Scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) X_test = scaler.transform(X_test)
```

DESIGN THINKING PROCESS PRESENT IN FORM OF DOCUMENT

- **Over the past few years, design thinking has quickly gained momentum in the business world. Some of the world’s leading brands—the likes of Apple, Google, HBO, Samsung, World Bank, and General Electric—have embraced design thinking as a means of optimizing product innovation. At its core, design thinking is a methodology for creative problem solving. In stark contrast to analytical thinking, which involves the breaking down of ideas, design thinking involves the building up of ideas.**
- **While design thinking has firmly implanted itself across product development teams, it has not secured a stronghold across sales teams—yet. Characterized by routinized activities, traditional sales methodologies tend to be at odds with the iterative methodology that underpins design thinking.**
- **Times are changing. The sales cycle is becoming increasingly complex and customers are demanding a more personalized experience. If you’re a sales rep, you know you need to up your game and become more innovative. Sales teams are recognizing the value of incorporating a design thinking approach into their**

daily activities. Salesforce's sales team, for example, has embraced design thinking in its sales discovery process and has realized a 100% increase in revenue growth as a result. It's time sales teams more broadly recognize the value of design thinking.

1.Empathize:

- **Empathy is at the core of design thinking. Empathy involves both a cognitive dimension—an ability to look at a situation from another person's perspective—as well as an affective dimenion—an ability to relate to relate to a person and develop an emotional bond with them.**
- **The importance of empathy in sales cannot be overstated. Empathy is a key predictor of sales success. A groundbreaking study published in The Journal of Marketing Theory and Practice found that there is a strong positive relationship between empathy and a buyer's level of trust, as well as his/her level of satisfaction. In our current sales landscape where a mere 3% of buyers trust reps—the only professions with less credibility include car sales, politics, and lobbying— seller trust is in short supply and high demand.**

2.Define:

- **The objective of the define stage is to craft a problem statement or, in design thinking speak, a point of view. So often salespeople define the problem before developing an empathetic understanding of a buyer's needs. The result is solution selling. Solution selling has long past its expiration date. At least 50% of sales reps' prospects are not good fits for their offering. Only by defining the buyer's problem can salespeople determine whether there is a lucrative fit.**
- **The define stage involves asking a lot of questions. Perhaps contrary to popular belief, this focus on questioning does not impair sales conversation, but rather enhances it. According to one analysis of 519,000 discovery calls, there's a clear relationship between the number of questions a sales rep asks a buyer and his/her likelihood of success.**

3.Ideate:

- **The ideate stage unlocks the true potential of design thinking, especially in the**

  Context of sales. This is when the focus shifts from problem identification to solution generation. And it's all about quantity—about generating a wide range of possible solutions, not necessarily the final solution. It involves thinking beyond the obvious and necessarily entails significant creativity. How can I craft an offering that is uniquely suited to my buyer?
- **While often pushed under the carpet in sales, creativity is essential to sales and a key predictor of success. Research from the Aston Business School, a highly-regarded business school in Europe, revealed that sales professionals who were more creative generated higher sales than their less creative counterparts. Another study by Adobe found that companies that foster creativity are 3.5 times more likely to outperform their peers in terms of revenue growth.**

- When crafting solutions to customers' problems, sales reps must dig deep for their creative juices. How can you craft a sales pitch that strikes a strong emotional chord with the customer? Which decision makers, in and beyond the C-Suite, should you involve? If the customer sells a free or inexpensive product or service, take it for a test run. Read through customer community forums and reviews. Don't let up in terms of stepping inside the shoes of your customer. Only by embracing these type of activities can ideation by optimized.

4.prototype:

- The fourth stage of the design thinking process is prototyping—developing more fleshed-out and scaled solutions. Prototyping shouldn't be done in a black box—otherwise you are sure to lose momentum. Prototyping is an opportunity to have a more directed conversation with your customer after the discovery calls. The most effective sales reps will involve champions and other affiliates from the customer's organization in the prototyping process and vet ideas with them. Involving tangential stakeholders in the solution process goes a long way in terms of making them feel valued and invested in the final solution.

5.Test:

- The final stage of the design thinking process is to test the final offering. This necessarily involves unveiling the fully fleshed-out pitch to all key stakeholders. During the test phase, salespeople need to be strategic and see themselves on the same team as the customer. They should use collaborative words and phrases—words like "we" and "together". The "you versus us" mentality is dangerous. Not surprisingly, research has revealed that top-performing salespeople are up to 10 times likelier to use collaborative words and phrases, as compared to their low-performing counterparts.
- Forrester predicts that one million US B2B sales reps will be out of a job by 2020. Salespeople can no longer afford to rely on so-called tried and true approaches. Nearly six in 10 salespeople say that when they figure out what works for them, they don't change it. In a world where each customer yearns personalized selling wants, this mindset is problematic. Design thinking—which is especially well suited for solving ambiguously defined problems—is key to establishing a genuine connection with customers and engaging them throughout the sales process. It's key to sales success.

PHASES OF DEVELOPMENT:

6 Sales Forecasting Methods To Help Predict Future Sales:

- **Sales forecasting is a process companies use to predict the future profitability of their business. Effective sales forecasting methods can help sales leaders properly allocate resources, adhere to a budget and make plans to expand.**
- **In this article, we define sales forecasting, explain why these methods are important, describe six key sales forecasting methods and share tips for choosing sales forecasting methods to predict future sales.**

1.Sales cycle length forecasting:

- **This forecasting method ranks opportunities based on how long a potential customer has been communicating with the company. By using historical data, sales professionals can identify the average amount of time it takes for a deal to close in their company**
- **Then, they can compare sales opportunities in their system with the average customer journey. For example, if a typical sales cycle for an HR software program is three months, this forecasting method might rank a lead that's been in the system for one month higher than one that's been in the system for five months.**

2.Intuitive Forecasting:

- **In intuitive forecasting, individual sales representatives predict whether a sale might close. If they predict a sale, they can add the opportunity size to the forecasted sales revenue.**

- **This method is one of the simpler forecasting types and allows sales representatives to use their expertise and intuition, which might make it a good fit for a smaller company with experienced sales professionals. It's also the most subjective method, so it might be an effective method to use along with a more quantitative forecasting process.**

3.Historical forecasting:

- **While most forecasting methods use historical data, the historical forecasting method uses it more extensively. This method compares the current lead count to past years' counts and then extrapolates a predicted sales value based on past years' revenue.**

- **Using historical forecasting might be effective in companies where the revenue from year to year has followed a consistent pattern. If the company is experiencing economic changes or has released new product lines, then this method might not be fully accurate on its own.**

4.opturnuity stage forecasting:

- **The opportunity (or deal) stage model evaluates deals based on where they are in the sales funnel. In this model, when a sales representative speaks to a potential customer, they estimate the likelihood that the deal might close at each stage of the process and enter that information into the company's sales management or customer service management (CRM) program.**

- **Using this information, the forecasting tool multiplies the probability of closing by the potential size of the deal to get a value estimate. This method relies on accurate and current information from the sales team.**

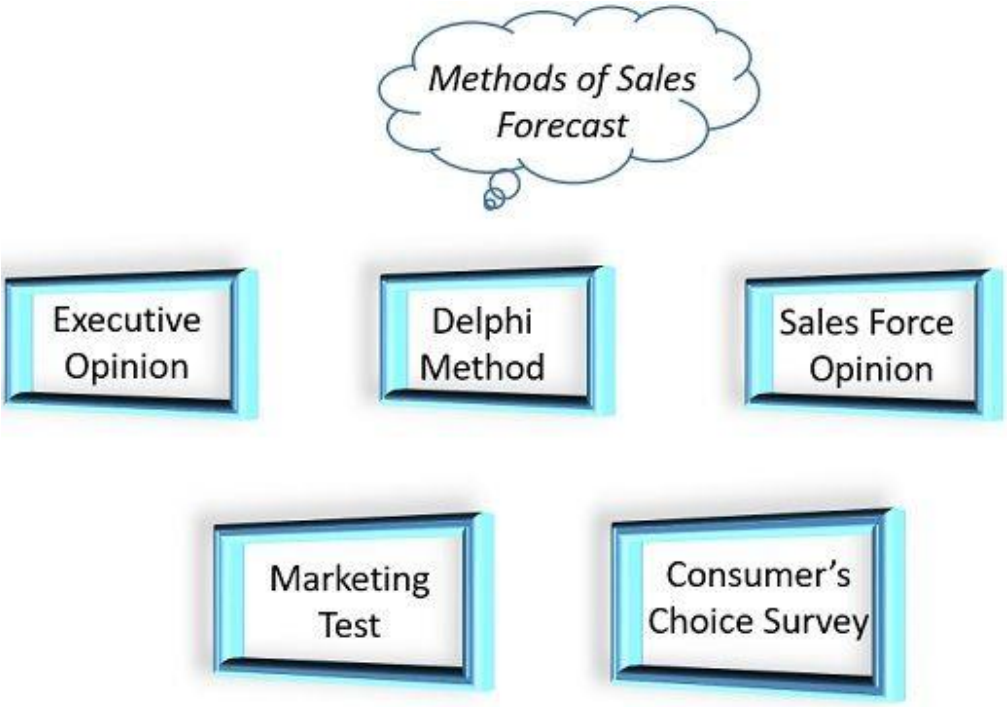
5.Pipeline forecasting:

- **A sales pipeline is a model of the company's sales cycle, populated by sales leads who are at different stages in the process. Pipeline forecasting is a method that analyzes each sales opportunity in the company's pipeline and assigns a value to each lead that shows profitable it might be.**

- **By evaluating each lead, this method gives an estimate of how much potential revenue currently exists in the pipeline. This method requires data about each lead, along with historical sales trends. Because of the data requirement, it might be a good fit for a larger or more established company.**

6.Multivariable forecasting:

- **A multivariable approach combines several of the other forecasting methods to create a customized plan for a company. Many sales management tools allow you to create multivariable analysis methods that account for pipeline, historical data and other factors.**



- **This approach might be effective for companies that are growing rapidly or that rely on a combination of sales data and qualitative information. It also allows you to add external factors, like price increases, into the calculation.**

Python

File descriptions

Sales_train.csv — the training set. Daily historical data from January 2013 to October 2015.

Test.csv — the test set. You need to forecast the sales for these shops and products for November 2015.

Sample_submission.csv — a sample submission file in the correct format.

Items.csv — supplemental information about the items/products.

Item_categories.csv — supplemental information about the items categories.

Shops.csv- supplemental information about the shops

Predict Future Sales using Machine Learning

Arnav Andraskar

Arnav Andraskar

-

Follow

Listen

Share

All industries aim to manufacture just the right number of products at the right time, but for retailers this issue is particularly critical as they also need to manage perishable inventory efficiently.

Too many items and too few items are both scenarios that are bad for business. (Estimates suggest that poor inventory management costs US retailers close to two billion dollars per year.)

Business Problem

Demand forecasting is the process of predicting what the demand for certain products will be in the future. This helps manufacturers to decide what they should produce and guides retailers toward what they should stock.

Demand forecasting is aimed at improving the following processes:

Supplier relationship management

Customer relationship management

Order fulfillment and logistics

Marketing campaigns

Manufacturing flow management

Use of Machine Learning

Machine learning techniques allows for predicting the amount of products/services to be purchased during a defined future period. In this case, a software system can learn from data for improved analysis. Compared to traditional demand forecasting methods, a machine learning approach allows you to:

Accelerate data processing speed

Provide a more accurate forecast

Automate forecast updates based on the recent data

Analyze more data

Identify hidden patterns in data

Create a robust system

Increase adaptability to changes

Reference: If you want to learn above topics in more detail, here's a blog by Liudmyla Taranenko from where it is abstracted.

Data Gathering

Dataset is taken from kaggle competition and can be downloaded from here.

Data Description:

You are provided with daily historical sales data. The task is to forecast the total amount of products sold in every shop for the test set. Note that the list of shops and products slightly changes every month. Creating a robust model that can handle such situations is part of the challenge.

File descriptions

Sales_train.csv — the training set. Daily historical data from January 2013 to October 2015.

Test.csv — the test set. You need to forecast the sales for these shops and products for November 2015.

Sample_submission.csv — a sample submission file in the correct format.

Items.csv — supplemental information about the items/products.

Item_categories.csv — supplemental information about the items categories.

Shops.csv- supplemental information about the shops.

Data fields

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_cnt_day — number of products sold. You are predicting a monthly amount of this measure

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

Exploratory Data Analysis

Requirement gathering

```
Import pandas as pd
Import numpy as np
Import matplotlib.pyplot as plt
Import seaborn as sns
```

EDA

```
Df_train = pd.read_csv(“sales_train.csv”)
Df_items = pd.read_csv(“items.csv”)
Df_item_cat = pd.read_csv(“item_categories.csv”)
Df_shops = pd.read_csv(“shops.csv”)
Df_test = pd.read_csv(“test.csv”)
```

```
Df_train.shape #shape of data
```

```
(2935849, 6)
```

```
Df_train.head() #reading top 5 rows
```

```
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
```

```
Df_train.info() #Data types and other info
```

```
<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    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
```

```
Df_train.describe() #statistical info
```

```
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  4.700
Max    3.300000e+01  5.900000e+01     2.216900e+04     3.079800e+05      2.169000e+03
```

```
Df_train.isnull().sum()
```

```
Df_train_3 = df_train.groupby(['date_block_num'], as_index=False)['item_cnt_day'].sum()
Plt.figure(figsize=(15,3))
Sns.barplot(x="date_block_num",y="item_cnt_day", data = df_train_3)
Plt.show()
```

EDA on shop_id and item_id exist in test data

```
Test_shop = df_test['shop_id'].unique()
Test_item = df_test['item_id'].unique()
# Only shops that exist in test data
_train = df_train[df_train['shop_id'].isin(test_shop)]
# Only items that exist in test data
_train = _train[_train['item_id'].isin(test_item)]
```

```
Print("shape after exluding unwanted shop and item ids ",_train.shape)
```

```
Shape after exluding unwanted shop and item ids  (1224439, 6)
```

```
Print("Number of Duplicates" ,len(_train[_train.duplicated()]))
```

```
Number of Duplicates 5
```

Outlier Analysing

```
Plt.figure(figsize=(10,4))
Sns.boxplot(x=_train.item_cnt_day)
For I in range(90,101):
  Print("{}th percenile value of item count:
{}".format(I,np.percentile(_train.item_cnt_day.values,i)))
Outlier_item_cnt = np.percentile(_train.item_cnt_day.values,100)
```

```
90th percenile value of item count: 2.0
91th percenile value of item count: 2.0
92th percenile value of item count: 2.0
93th percenile value of item count: 2.0
94th percenile value of item count: 2.0
95th percenile value of item count: 2.0
96th percenile value of item count: 3.0
97th percenile value of item count: 3.0
98th percenile value of item count: 4.0
99th percenile value of item count: 7.0
100th percenile value of item count: 2169.0
```

99% dataset has 7 or less sellings

```
Plt.figure(figsize=(10,4))
Sns.boxplot(x=_train.item_price)
For I in range(90,101):
  Print(“{}th percenile value of item price:
{}”.format(I,np.percentile(_train.item_price.values,i)))
Outlier_item_price = np.percentile(_train.item_price.values,100)
```

```
90th percenile value of item price: 2499.0
91th percenile value of item price: 2599.0
92th percenile value of item price: 2599.0
93th percenile value of item price: 2599.0
94th percenile value of item price: 2799.0
95th percenile value of item price: 2999.0
96th percenile value of item price: 3190.0
97th percenile value of item price: 3490.0
98th percenile value of item price: 3790.0
99th percenile value of item price: 4990.0
```

100th percenile value of item price: 59200.0

99% dataset has 4990 or less item price

Shop wise selling

```
Df_train_3 = _train.groupby([‘shop_id’], as_index=False)[‘item_cnt_day’].sum()
Plt.figure(figsize=(20,4))
Sns.barplot(x=”shop_id”,y=”item_cnt_day”, data=df_train_3)
Plt.show()
```

Catogery wise selling

```
_train_2 = _train.merge(df_items, on=’item_id’, how=’left’ )
Df_train_3 = _train_2.groupby([‘item_category_id’], as_index=False)[‘item_cnt_day’].sum()
Plt.figure(figsize=(23,4))
Sns.barplot(x=”item_category_id”,y=”item_cnt_day”, data = df_train_3)
Plt.show()
```

Month wise selling

```
Df_train_3 = _train.groupby([‘date_block_num’], as_index=False)[‘item_cnt_day’].sum()
Plt.figure(figsize=(15,3))
Sns.barplot(x=”date_block_num”,y=”item_cnt_day”, data = df_train_3)
Plt.show()
```

Data processing and feature engineering

Here I’ am removing outliers and negative item price and item counts

I’ am considering all the ‘item prices’ >50000 and ‘item count a day ’ >1000 as an outliers.

```
#Join data sets

Train = sales.join(items, on=’item_id’, rsuffix=’_’).drop([‘item_id_’,’item_name’], axis=1)

# Remove outlier

Train = train[(train.item_price < 50000 )& (train.item_cnt_day < 1000)]

# remove negative item price and item count day

Train = train[(train.item_price > 0) & (train.item_cnt_day >= 0)].reset_index(drop = True)


# Cleaning shops data

# We have different shop ids for same shop names like 0-57, 1-58, 11,10, 40-39

# We don’t have 0,1,11 and 40 shop_id in test data so we are replacing these with shop_id which shares a similar names

Train.loc[train.shop_id == 0, ‘shop_id’] = 57


Train.loc[train.shop_id == 1, ‘shop_id’] = 58


Train.loc[train.shop_id == 11, ‘shop_id’] = 10


Train.loc[train.shop_id == 40, ‘shop_id’] = 39
```

Here I’ am converting daily historical data into monthly sales data .Also extracting (‘item_cnt_month’, ‘mean_item_cnt’, ‘mean_item_price’, ‘revenue_month’) features using aggregation function.

Aranging columns

```
Train_monthly = train[[‘date’, ‘date_block_num’, ‘shop_id’, ‘item_category_id’, ‘item_id’, ‘item_price’, ‘item_cnt_day’]]

Train_monthly[‘revenue’] = train_monthly[‘item_price’] * train_monthly[‘item_cnt_day’]


# Group by month in this case “date_block_num” and aggregate features.

Train_monthly = train_monthly.sort_values(‘date’).groupby([‘date_block_num’, ‘shop_id’, ‘item_category_id’, ‘item_id’], as_index=False)

Train_monthly = train_monthly.agg({ ‘item_cnt_day’:[‘sum’, ‘mean’],’item_price’:[‘mean’], ‘revenue’:[‘sum’]})
```

```
# Rename columns

Train_monthly.columns = ['date_block_num', 'shop_id', 'item_category_id', 'item_id', 'item_cnt_month', 'mean_item_cnt', 'mean_item_price', 'revenue_month']
```

Machine Learning

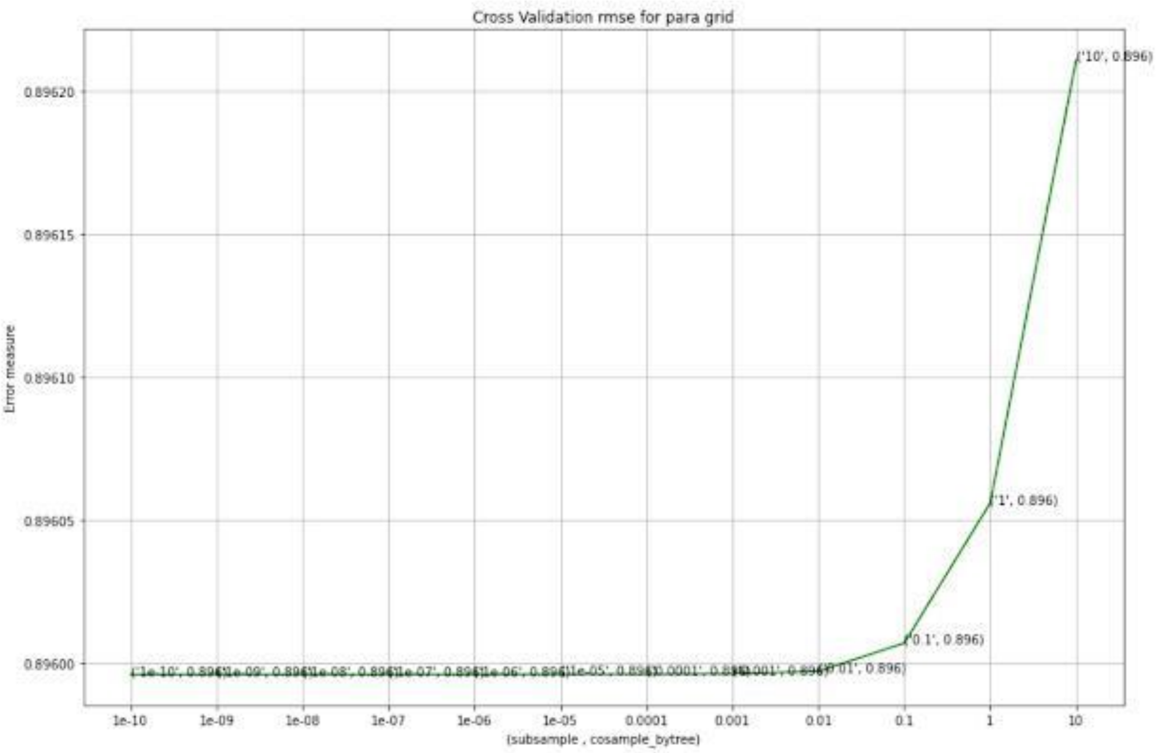
We are done with feature extraction and data preprocessing. In this section I' am training different models , starting from very basic models like lasso regression to more complex models like XGBoost and LightGBM.

1. **Lasso Regression**

Here I' am tuning "alpha" parameter

```
Alpha = [10**I for I in range(-6,2)]



Train_score = []

Val_score = []

For I in tqdm(alpha):


    Print("alpha = {} ".format(i))

    _model = Lasso(alpha= i)

    _model.fit(X_train, y_train)

    Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train, squared=False)

    Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val, squared=False)


    If val_score:

        If sorted(val_score)[0] > rmse_val:

            Print("model saving…..")

            With open('D:/Case Study/Predict future sales/best_lasso','wb') as loc:

                Pickle.dump(_model,loc)

    Else:

        Print("model saving…..")

        With open('D:/Case Study/Predict future sales/best_lasso','wb') as loc:

            Pickle.dump(_model,loc)
    Train_score.append(rmse_train)
    Val_score.append(rmse_val)
    Print("Training Loss is {} ".format(rmse_train))
    Print("Validation Loss is {} ".format(rmse_val))
    Print("-"*50)

With open('D:/Case Study/Predict future sales/lasso_log','wb') as loc:
    Pickle.dump((train_score,val_score,alpha),loc)
```

Alpha vs score



Lasso regression Best score train: 0.77688rmse test:0.896rmse.

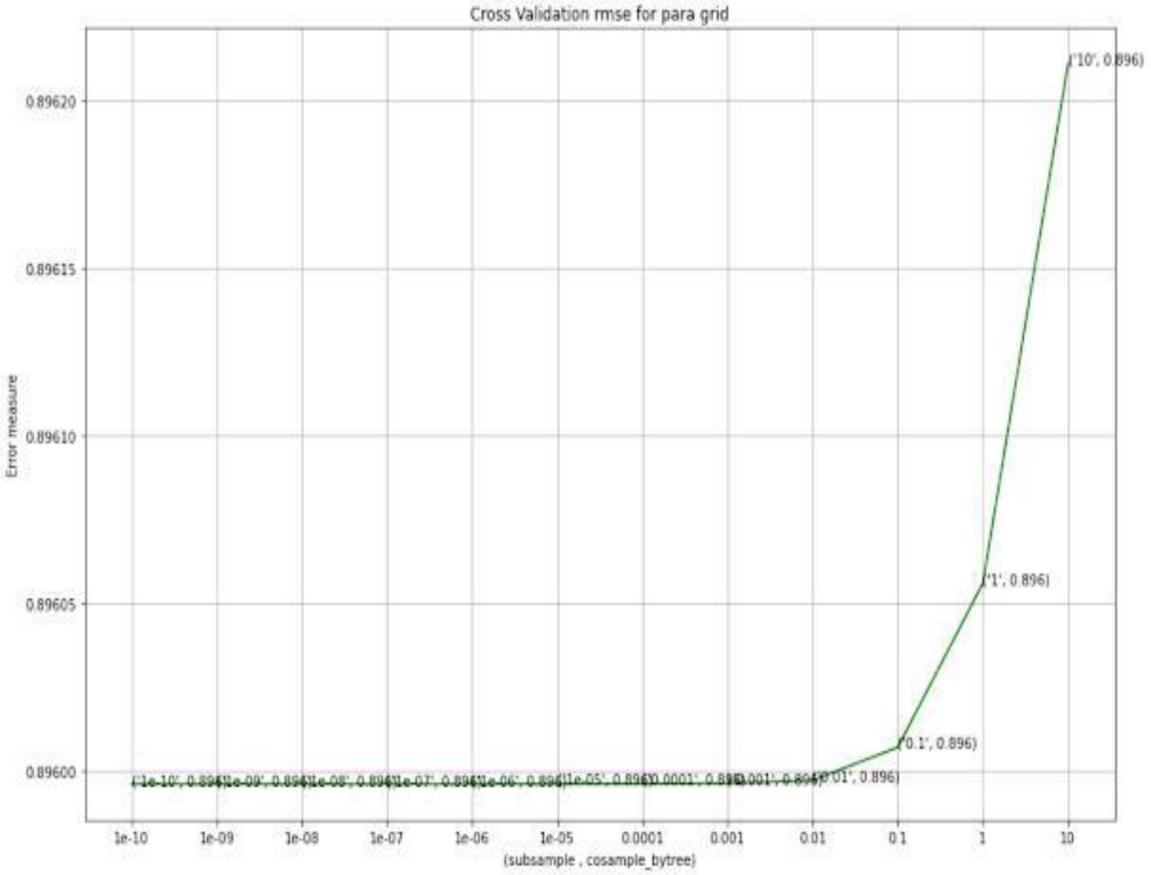
2. **Ridge Regression**

Here I' am tuning "alpha " parameter

```
Alpha = [10**I for I in range(-10,2)]
```

```
Train_score = []
Val_score = []
For I in tqdm(alpha):

   Print(“alpha = {} ”.format(i))
   _model = Ridge(alpha= i)
   _model.fit(X_train, y_train)
   Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train, 
squared=False)
   Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val, squared=False)

   If val_score:
      If sorted(val_score)[0] > rmse_val:
         Print(“model saving.....”)
         With open(‘D:/Case Study/Predict future sales/best_ridge’,’wb’) as loc:
            Pickle.dump(_model,loc)
   Else:
      Print(“model saving.....”)
      With open(‘D:/Case Study/Predict future sales/best_ridge’,’wb’) as loc:
         Pickle.dump(_model,loc)

   Train_score.append(rmse_train)
   Val_score.append(rmse_val)
   Print(“Training Loss is {} ”.format(rmse_train))
   Print(“Validation Loss is {} ”.format(rmse_val))
   Print(“-“*50)

With open(‘D:/Case Study/Predict future sales/ridge_log’,’wb’) as loc:
   Pickle.dump((train_score,val_score,alpha),loc)
```

Alpha vs score



Ridge Regression best score train:0.77689 **rmse test**: 0.896**rmse**.

3. **Dicision Tree**

Here I' am tuning "max_depth " parameter

Alpha = [3,5,7,9,12]

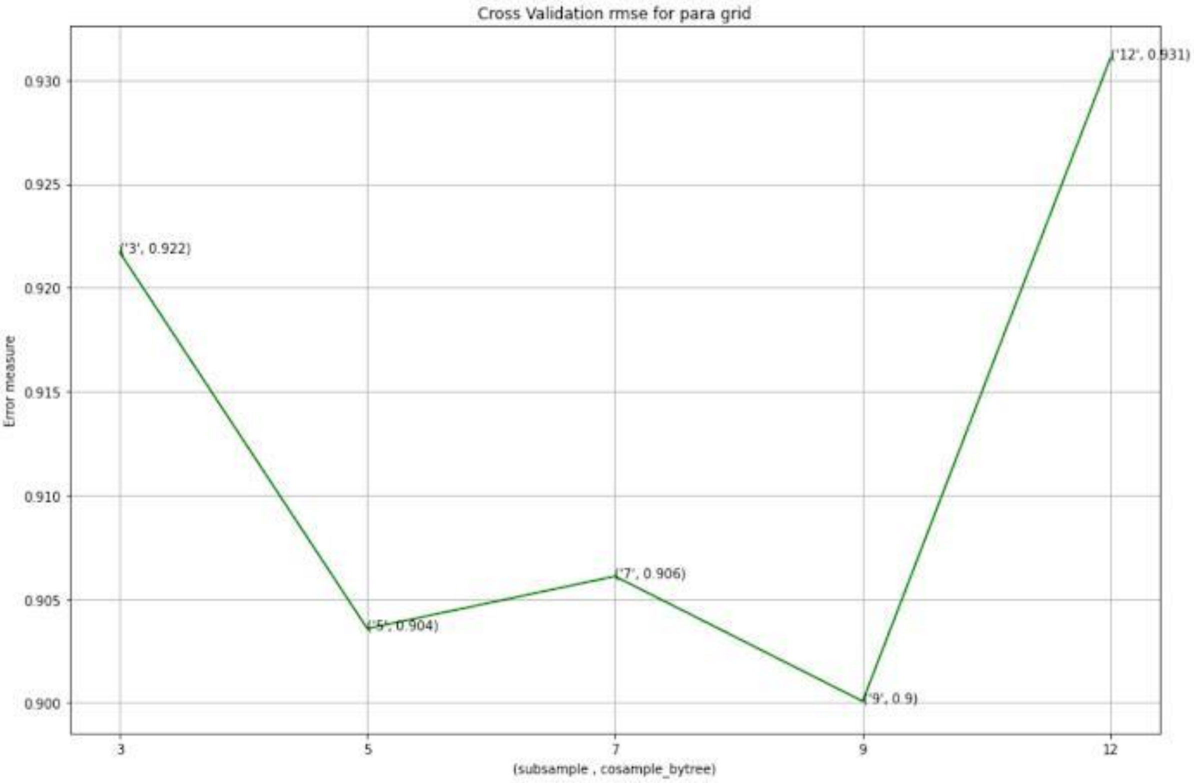
Train_score = []

Val_score = []

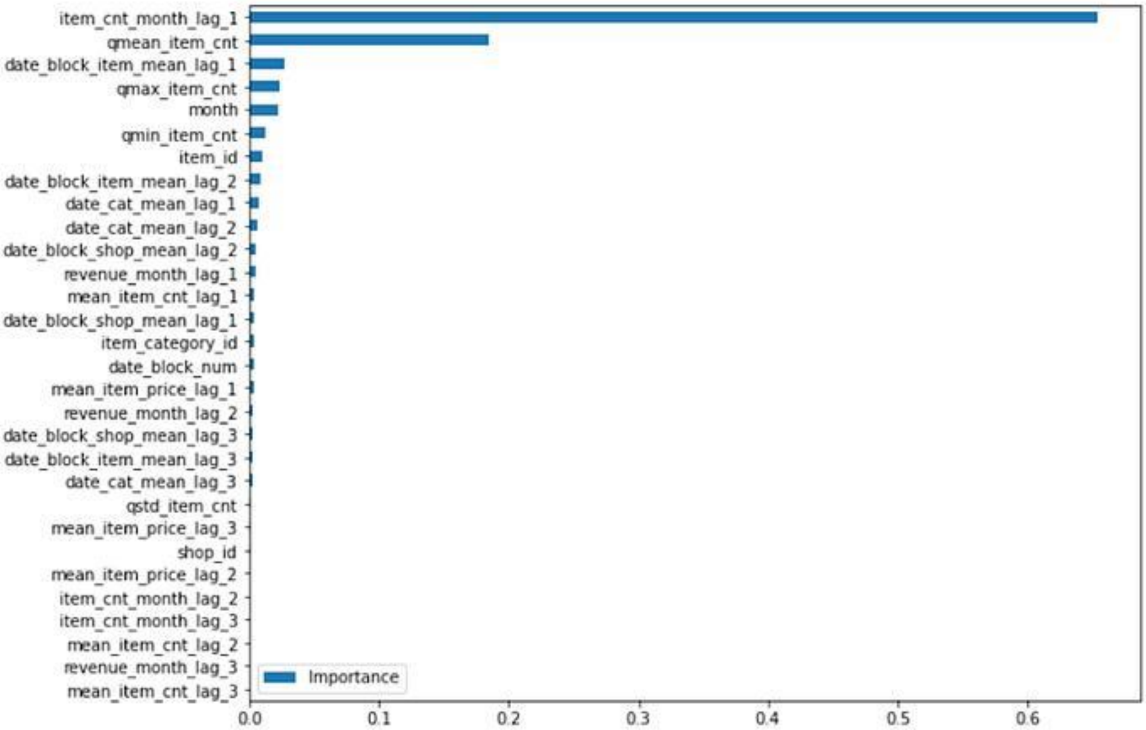
For I in tqdm(alpha):

```
    Print(“alpha = { } ”.format(i))
    _model = DecisionTreeRegressor(max_depth = i)
    _model.fit(X_train, y_train)
    Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train, squared=False)
    Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val, squared=False)

    If val_score:
        If sorted(val_score)[0] > rmse_val:
            Print(“model saving.....”)
            With open(‘D:/Case Study/Predict future sales/best_dt’,’wb’) as loc:
                Pickle.dump(_model,loc)
    Else:
        Print(“model saving.....”)
        With open(‘D:/Case Study/Predict future sales/best_dt’,’wb’) as loc:
            Pickle.dump(_model,loc)

    Train_score.append(rmse_train)
    Val_score.append(rmse_val)
    Print(“Training Loss is { } ”.format(rmse_train))
    Print(“Validation Loss is { } ”.format(rmse_val))
    Print(“-“*50)

With open(‘D:/Case Study/Predict future sales/dt_log’,’wb’) as loc:
    Pickle.dump((train_score,val_score,alpha),loc)
```

Here’s a visualization max_depth vs score



Here's feature importance graph of decision tree



Decision tree Regressor best score train:0.74485**rmse test:**0.90008**rmse**

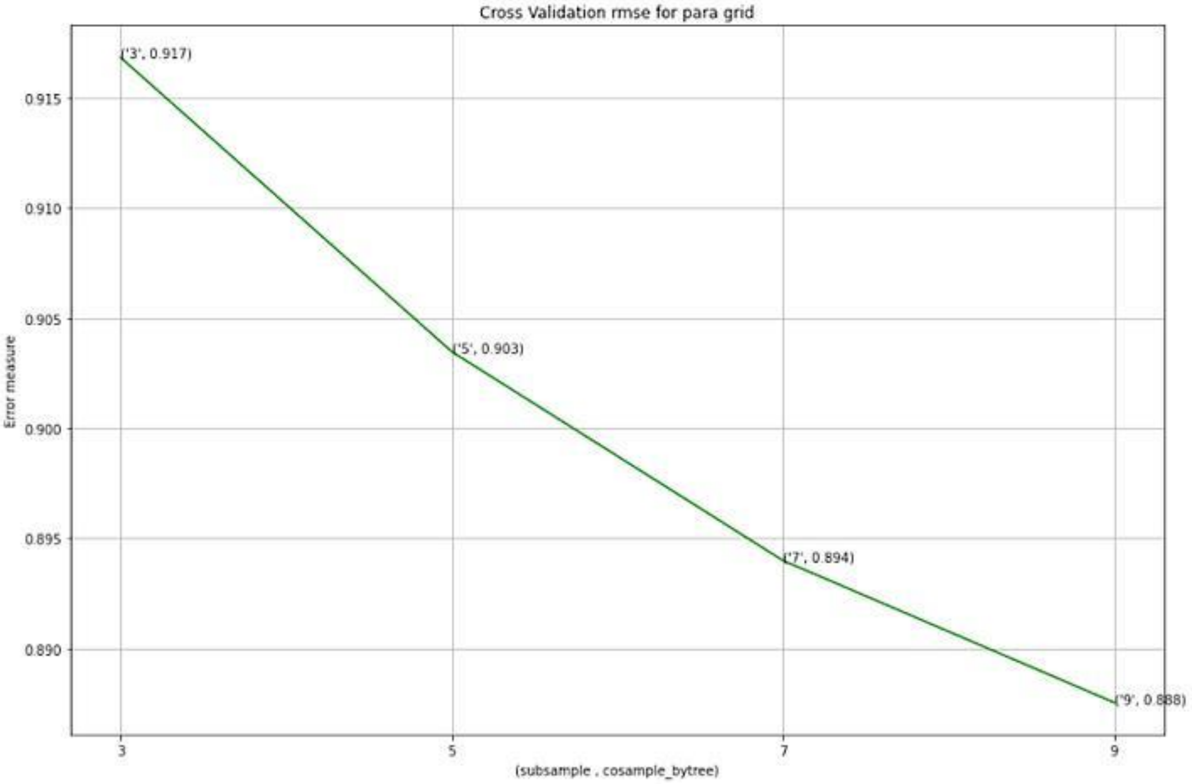
4. **Random Forest**

Here I' am tuning "max_depth " parameter

```
Alpha = [3,5,7,9]



Train_score = []

Val_score = []

For I in tqdm(alpha):


   Print("alpha = {} ".format(i))

   _model = RandomForestRegressor(max_depth = i)

   _model.fit(X_train, y_train)

   Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train,
squared=False)

   Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val, squared=False)
```

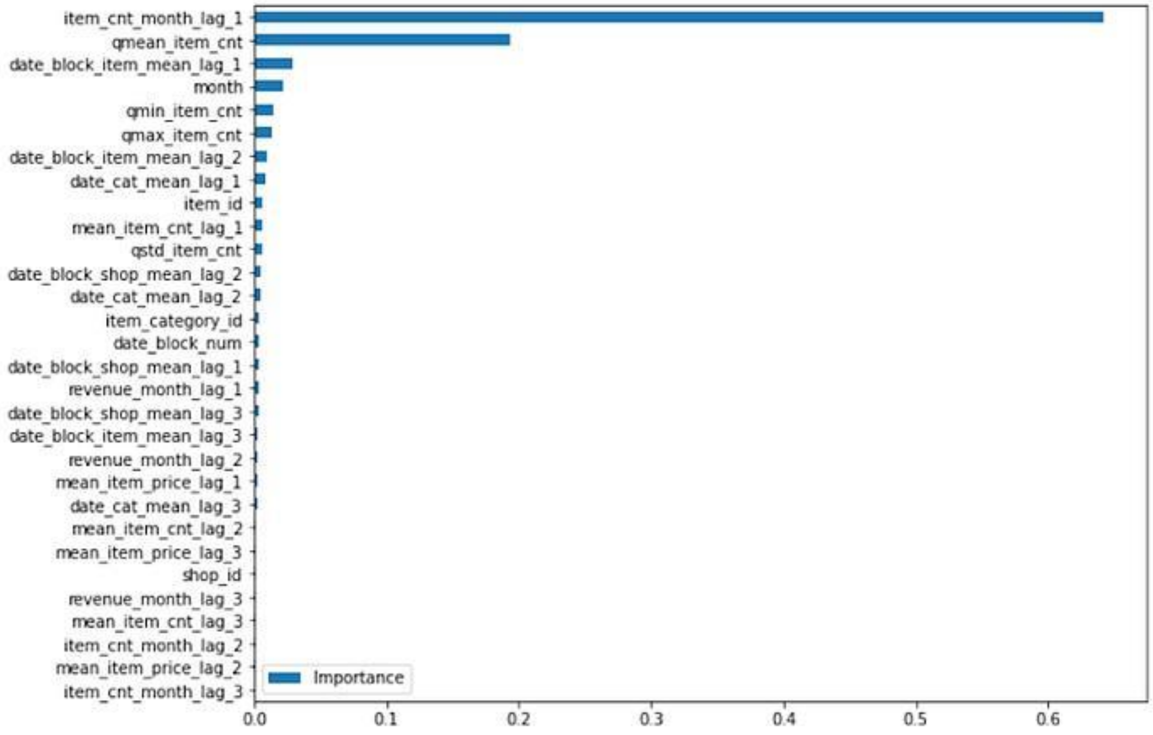
```
    If val_score:
        If sorted(val_score)[0] > rmse_val:
            Print("model saving.....")
            With open('D:/Case Study/Predict future sales/best_rf','wb') as loc:
                Pickle.dump(_model,loc)
    Else:
        Print("model saving.....")
        With open('D:/Case Study/Predict future sales/best_rf','wb') as loc:
            Pickle.dump(_model,loc)

    Train_score.append(rmse_train)
    Val_score.append(rmse_val)
    Print("Training Loss is { } ".format(rmse_train))
    Print("Validation Loss is { } ".format(rmse_val))
    Print("-"*50)

With open('D:/Case Study/Predict future sales/rf_log','wb') as loc:
    Pickle.dump((train_score,val_score,alpha),loc)
```

Here's a graph showing of parameter vs score



Here's a bar graph showing feature importance of best model

Random Forest Regressor best scoretrain:0.73291**rmse test:**0.88755**rmse.**



5. **XGBoost Regressor**
I did hyperparameter tuning of XGB in two parts
In first part I have tuned max_depth and min_child_weight parameter and in second part I have tuned subsample and cosample_bytree parameter

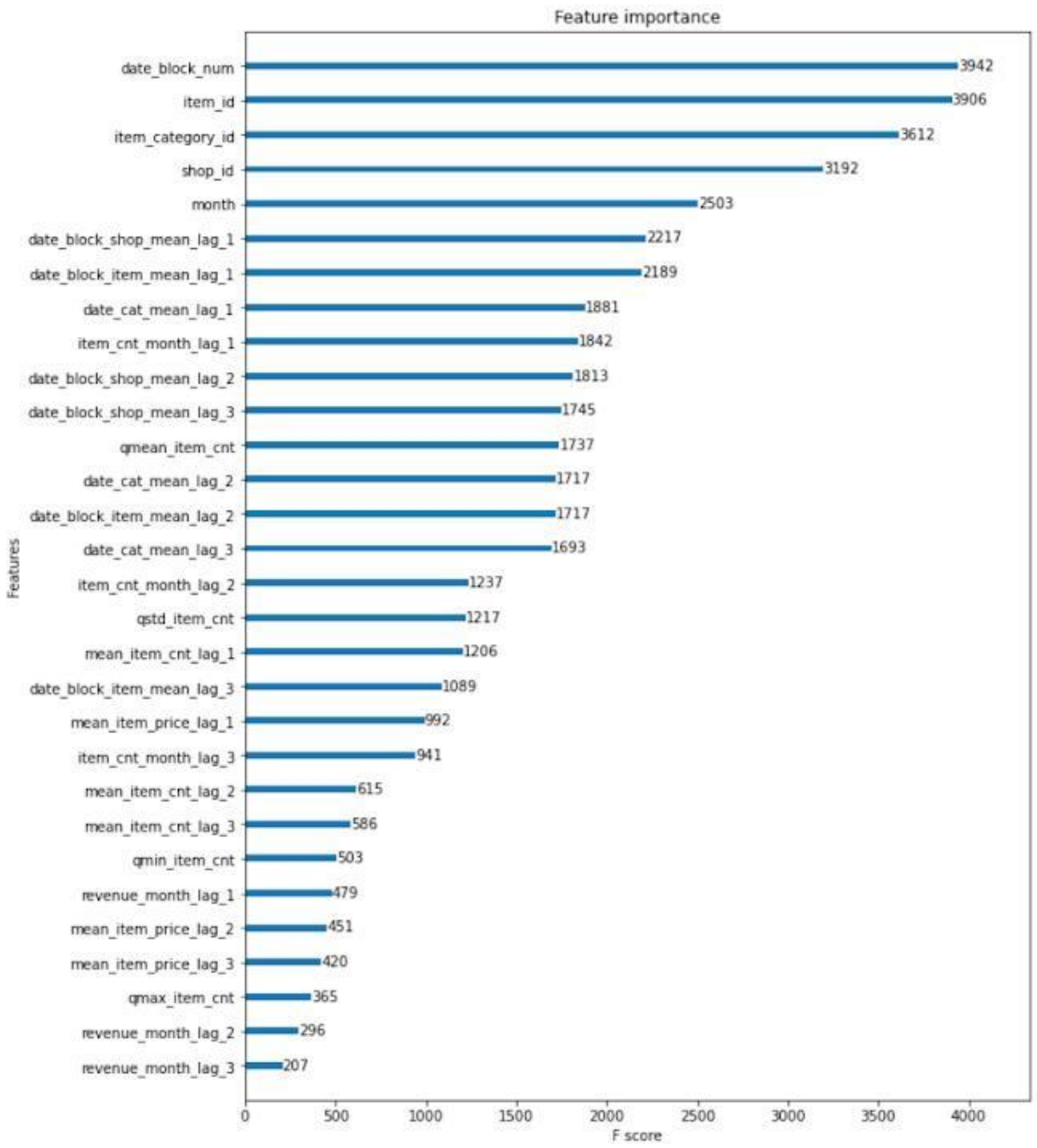
Part 1:
This is a custom grid search code

```
Max_depth = [4, 6, 8, 10, 12]
Min_child_weight = [1, 10, 100]

Fit_params={"early_stopping_rounds":20,
            "eval_metric" : "rmse",
            "eval_set" : [[X_val, y_val]] }

Train_score = []
Val_score = []
Params = []
For I in max_depth:
    For j in min_child_weight:
        Print("Tuning parameter subsamle and cosample_bytree")
        Print("set subsamle = {} and cosample_bytree = {}".format(I,j))
        _model = XGBRegressor(eta = 0.1, n_estimators=1000,max_depth=I
,min_child_weight=j, verbose=100)
        _model.fit(X_train, y_train, verbose=True, **fit_params)
        Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train,
squared=False)
        Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val,
squared=False)

        If val_score:
            If sorted(val_score)[0] > rmse_val:
                Print("model saving.....")
                With open('D:/Case Study/Predict future sales/best_xgb_1','wb') as loc:
                    Pickle.dump(_model,loc)
        Else:
            Print("model saving.....")
            With open('D:/Case Study/Predict future sales/best_xgb_1','wb') as loc:
                Pickle.dump(_model,loc)

        Train_score.append(rmse_train)
        Val_score.append(rmse_val)
        Params.append((I,j))

        Print("Training Loss when max_depth={} and min_child_weight={} is {}
".format(I,j,rmse_train))
        Print("Validation Loss when max_depth={} and min_child_weight={} is {}
".format(I,j,rmse_val))
        Print("-"*50)

With open('D:/Case Study/Predict future sales/xgb_log_1','wb') as loc:
    Pickle.dump((train_score,val_score,params),loc)
```

Part 2:

```
Subsample = [0.6,0.7,0.8,0.9]
Cosample_bytree = [0.6,0.7,0.8,0.9]

Fit_params={"early_stopping_rounds":10,
          "eval_metric" : "rmse",
          "eval_set" : [[X_val, y_val]] }

Best_param_1 = params[np.argmin(val_score)]

Train_score_2 = []
Val_score_2 = []
Params_2 = []
For I in subsample:
   For j in cosample_bytree:
      Print("Tuning parameter subsamle and cosample_bytree")
      Print("set max_depth = {} and min_child_weight =
{}".format(best_param_1[0],best_param_1[1]))
      Print("set subsamle = {} and cosample_bytree = {}".format(I,j))
      _model = XGBRegressor(eta = 0.1,
n_estimators=1000,max_depth=best_param_1[0]
,min_child_weight=best_param_1[1],subsample=I,cosample_bytree=j)
      _model.fit(X_train, y_train, verbose=True, **fit_params)
      Rmse_train = mean_squared_error(_model.predict(X_train).clip(0,20),y_train,
squared=False)
      Rmse_val = mean_squared_error(_model.predict(X_val).clip(0,20),y_val,
squared=False)

      If val_score_2:
         If sorted(val_score_2)[0] > rmse_val:
            Print("saving model.....")
            With open('D:/Case Study/Predict future sales/best_xgb_2','wb') as loc:
               Pickle.dump(_model,loc)
      Else:
         Print("saving model.....")
         With open('D:/Case Study/Predict future sales/best_xgb_2','wb') as loc:
            Pickle.dump(_model,loc)

      Train_score_2.append(rmse_train)
      Val_score_2.append(rmse_val)
      Params_2.append([I,j])

      Print("Training Loss when subsample={} and cosample_bytree={} is {}
".format(I,j,rmse_train))
      Print("Validation Loss when subsample={} and cosample_bytree={} is {}
".format(I,j,rmse_val))
      Print("-"*50)

With open('D:/Case Study/Predict future sales/xgb_log_2','wb') as loc:
   Pickle.dump([train_score_2,val_score_2,params_2],loc)
```

Here's bar graph showing feature importance of best scored model



Model training proces:

- One of the most common methods used to predict sales is regression analysis. This method involves using historical sales data to train a model that can predict future sales. The model can take into account factors such as past sales, marketing campaigns, and economic indicators to make its predictions

TIME SERIES FORSTING ALGORITHM AND EVALUATION METRICS

A Guide to Different Evaluation Metrics for Time Series Forecasting Models:

- Measuring the performance of any machine learning model is very important, not only from the technical point of view but also from the business perspective.
  Table of Contents

Measuring Time Series Forecasting Performance

- The fact that the future is wholly unknown and can only be predicted from what has already occurred is a significant distinction in forecasting. The ability of a time series forecasting model to predict the future is defined by its performance.

Evaluation Metrics to Measure Performance

- Now, let us have a look at the popular evaluation metrics used to measure the performance of a time-series forecasting model.

R-Squared

- The stationary R-squared is used in time series forecasting as a measure that compares the stationary part of the model to a simple mean model. It is defined as,

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Where SSres denotes the sum of squared residuals from expected values and SStot denotes the sum of squared deviations from the dependent variable's sample mean. It denotes the proportion of the dependent variable's variance that may be explained by the independent variable's variance. A high R2 value shows that the model's variance is similar to that of the true values, whereas a low R2 value suggests that the two values are not strongly related.

Mean Absolute Error (MAE)

- The MAE is defined as the average of the absolute difference between forecasted and true values. Where yi is the expected value and xi is the actual value (shown below formula). The letter n represents the total number of values in the test set.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

- The MAE shows us how much inaccuracy we should expect from the forecast on average. MAE = 0 means that the anticipated values are correct, and the error statistics are in the original units of the forecasted values.

Mean Absolute Percentage Error (MAPE)

- MAPE is the proportion of the average absolute difference between projected and true values divided by the true value. The anticipated value is Ft, and the true value is At. The number n refers to the total number of values in the test set.

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

- It works better with data that is free of zeros and extreme values because of the in-denominator

Mean Squared Error (MSE)

- MSE is defined as the average of the error squares. It is also known as the metric that evaluates the quality of a forecasting model or predictor. MSE also takes into account variance (the difference between anticipated values) and bias (the distance of predicted value from its true value).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Where y’ denotes the predicted value and y denotes the actual value. The number n refers to the total number of values in the test set. MSE is almost always positive, and lower values are preferable. This measure penalizes large errors or outliers more than minor errors due to the square term (as seen in the formula above).

Root Mean Squared Error(RMSE)

- This measure is defined as the square root of mean square error and is an extension of MSE. Where y’ denotes the predicted value and y denotes the actual value. The number n refers to the total number of values in the test set. This statistic, like MSE, penalizes greater errors more.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

- This statistic is likewise always positive, with lower values indicating higher performance. The RMSE number is in the same unit as the projected value, which is an advantage of this technique. In comparison to MSE, this makes it easier to comprehend.

Normalized Root Mean Squared Error (NRMSE)

- The normalized RMSE is used to calculate NRMSE, which is an extension of RMSE. The mean or the range of actual values are the two most used methods for standardizing RMSE (difference of minimum and maximum values). The maximum true value is ymax, while the smallest true value is ymin.

$$\mathrm{NRMSD} = \frac{\mathrm{RMSD}}{y_{\max} - y_{\min}} \text{ or } \mathrm{NRMSD} = \frac{\mathrm{RMSD}}{\bar{y}}$$

- NRMSE is frequently used to compare datasets or forecasting models with varying sizes (units and gross revenue, for example). The smaller the value, the better the model's performance. When working with little amounts of data, this metric can be misleading. However, Weighted Absolute Percentage Error and Weighted Mean Absolute Percentage Error can help.

Weighted Mean Absolute Percentage Error (WMAPE)

- WMAPE (sometimes called wMAPE) is an abbreviation for Weighted Mean Absolute Percentage Error. It is a measure of a forecasting method's prediction accuracy. It is a MAPE version in which errors are weighted by real values (e.g. in the case of sales forecasting, errors are weighted by sales volume).

$$\mathrm{WMAPE} = \frac{\sum_{t=1}^{n} |A_t - F_t|}{\sum_{t=1}^{n} |A_t|}$$

- Where A is the current data vector and F is the forecast This metric has an advantage over MAPE in that it avoids the 'infinite error' problem.

- The higher the model's performance, the lower the WMAPE number. When evaluating forecasting models, this metric is useful for low volume data where each observation has a varied priority. The weight value of observations with a higher priority is higher. The WMAPE number increases as the error in high-priority forecast values grows.

CONCLUSION:

- Sales forecasting is mainly required for the organizations for business decisions. Accurate forecasting will help the companies to enhance the market growth. Machine learning techniques provides the effective mechanism in prediction and data mining as it overcome the problem with traditional techniques.
- Sales forecasting is a critical part of the strategic planning process and allows a company to predict how their company will perform in the future. It allows them to not only plan for new opportunities, but also allows them to avert negative trends that appear in the forecast. A mission statement is important because it allows an organization to know exactly why they exist and serves as a guide for decisions. Both concepts are important to the success of the company and should not be overlooked throughout the strategic planning process.