EXPLORATORY PROJECT:

BACKORDER PREDICTION

ABSTRACT
An Explanatory Project done by a team of 5 Mechanical Sophomores under the guidance of Prof Dr Prabhas Bharadwaj.

Preface

We are a team of 5 sophomores in the Mechanical Engineering Department. This document is a report for the exploratory project of 4th Semester. This project was done under the guidance of Prof. Dr. Prabhas Bharadwaj.

This project focuses on the domain of Inventory Management under the parent domain of Supply Chain Management which plays a very important role in today's corporate world. Without a proper Supply Chain, a company has no way to reach to any other level in the corporate world like Manufacturer, Retailer, Customer, etc. Inside the huge world of Supply Chain, Inventory Management is one of its biggest and most important management section through which the stock/inventory of a company is determined and managed. One of the most famous and frequent problems that occurs in inventories is the "**Backorder**" problem. Backorder need not always be a problem, but the trade-off between problematic and advantageous role of Backorder is very important for attaining proper Inventory Management.

This project mainly deals with this particular "Backorder" problem and we have tried to tackle the same through Supply chain concepts combined with advanced computing concepts like Machine Learning models and Data Science. To determine how good our model is performing, we have also taken care of the metrics that need to be looked at.

The project report starts with an introduction to what the problem of Backorder is, and then starts to go into detail on the same and then, goes on with explaining how data science (feature engineering) and Machine Learning Models deals with this problem with the help of a dataset.

Contents

Introduction

What is Backorder?

Also known as "Backlog", Backorder is what we call "out of stock" in layman language. To be elaborate, a backorder is an order for a product that cannot be fulfilled at the moment because of lack of available supply. The backorder indicates that the demand exceeds the supply of the company in question. Hence, we can say that Backorder is directly proportional to demand.

Company and Backorder:

So, from this we can understand that Backorder does not occur due to the substandard performance of the company, it occurs because the company has performed excellently on the product such that it has gone out of supplies. Backorder seems to be a problem, but no. Backorder is not always a problem; it can also be used tactically to boost demand and increase the number of customers, and increase the product value. The department/branch which deals with Backorders and Backorder Prediction is “Inventory Management”. The scale of how the company manages their inventory is indicated by the number of backordered products.

Backorder Accounting:

Backorders is a situation that deals with inventory management, it requires special accounting. This is because Inventory management comes in between buying the product from manufacturer and selling it to customer. The general procedure for accounting for Backorders is as follows. When the order is placed, the company informs the customer that the product they ordered is in backorder and has an approximate date of delivery. To avoid restoring all accounting records in case of cancellation, the sale is recorded in the company’s books specially as “backordered” instead of combining it with the other records of completed sales. After this, the company calls the manufacturer and urges them to deliver the goods as soon as possible so that the order is not cancelled. Once the company receives the product, it tracks the backordered record and delivers it to the customers. Finally, the sale can be recorded as completed.

Merits and Demerits of Backorder

Merits of Backorder:

1. Boosting the demand and thus creating hype about those products which customers buy regularly and which go out of stock very frequently.
2. The company can reduce the storage costs of products through Backorder. If they have items in their inventory that do not sell regularly, there is an unwanted addition to the storage costs. But if the order is placed after the customer orders the particular product, the storage cost can be optimized and the company can reduce unwanted expenses.
3. The company can sell customized products in case of a Backorder, because if the item is not in stock right now, it can accept customization requests from the customers, which is not the case if the product is in stock. This also creates hype and desire among the customer base and increases both the product and company's popularity.
4. When money is not wasted in the form of over-spaced warehouses and unnecessary carrying costs, part of it can be used to give discounts to customers. This will help the company eventually, which is ultimately the goal of any company.

Demerits of Backorder:

1. If the Backordered products of the company are very frequent over time, then it is an indication of inefficiency of inventory management and operations.
2. Frequent Backorders (even if the ones to create hype are frequent) lead to customers losing their trust on the company because every time they place an order for a product, the company reports it to be out of stock, and this ultimately leads to churning of customers.
3. Piling up backorders might result in high lead time, huge increase in storage/transportation costs and big lot size. These problems ultimately increase the expense of the company by a huge amount.
4. High Frequency of backorders lead to order cancellations and thus the spending done by company to supply that product is wasted.
5. All this leads to losing market share, which can be very negatively impactful on the company eventually.

Now we know Backorder has both nearly equal amount of merits and demerits and thus we can come to a conclusion that we cannot certainly say backorder brings in complete profit or complete loss, rather state that everything about Backorder is a trade-off between cost, intuition, and hype.

This is why the backorders (especially the unintentional ones i.e., when the company does not plan on backordering, but the product is genuinely out of stock) should be predicted so that the backorder does not lead to any of the above-mentioned demerits like customer dissatisfaction, losing market share, etc. If the backorders are predicted, the company will be able to focus on the trade-off between increasing demand unwantedly and deliberately backordering the products to lower the costs and storage space.

Now that we have completed the introduction and gone to the details of what the topic is and why it needs to be predicted, let us proceed to the next section of Introduction to Dataset and its briefing.

About the Dataset

Data Source:

Below is the link from where we downloaded the Data.
https://www.kaggle.com/chandanareddy12/back-order-prediction

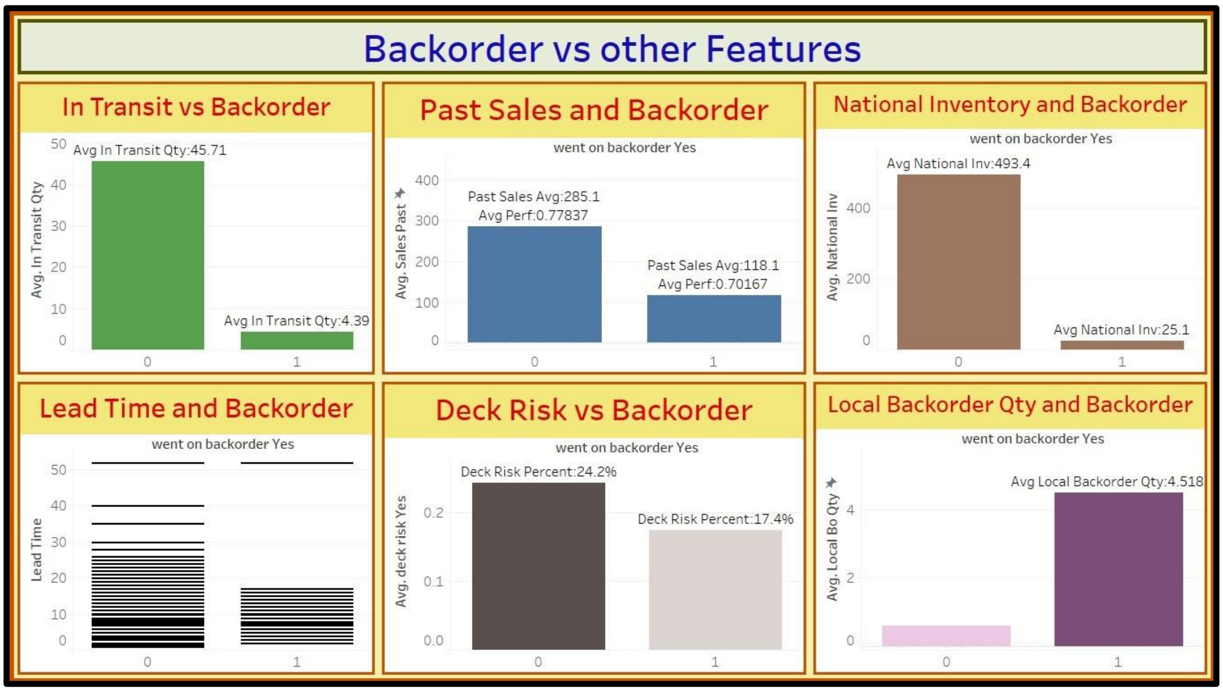
Metadata:

Data Fields	**Meaning**
sku	Stock Keeping Unit. (Unique number for every different kind of product.)
national_inv	Current inventory level of component. National Inv here means Inventory Level of all suppliers in a region, from where all other companies like ours purchase products.
lead_time	Purchase order lead time is the number of days from when a company places an order for supplies, to when those items arrive.
in_transit_qty	Quantity in transit.
forecast_x_month	Forecast sales for the next 3, 6, 9 months.
sales_x_month	Sales quantity for the prior 1, 3, 6, 9 months.
min_bank	Minimum recommended amount in stock.
potential_issue	Indicator variable noting potential issue with item.
pieces_past_due	Parts overdue from source.
perf_x_months_avg	Source performance in the last 6 and 12 months.
local_bo_qty	Amount of stock orders overdue.
"Remaining Columns before target column"	General Risk Flags of the product shipped/manufactured for predicting backorder (deck risk, oe constraint, ppap risk, stop_auto_buy, rev_stop,etc)
went_on_backorder	Product went on backorder or not.

Exploratory Data Analysis (EDA)

This section will be consisting of the Data Analysis part of this project. The Dataset has been connected to Tableau and the screen shots of the Dashboards created have been inserted below and the explanation for the same has been given.

1. Backorder vs Other Features



1. **In Transit vs Backorder:** As we can clearly see from the top left corner graph, for the items that are not backordered, the "In transit quantity" is much higher than if the backordered ones.

2. **Past Sales vs Backorder:** In the top center viz, we can observe that opposite to what we expected (higher performance increases demand and thus increases backordering), the backordered quantities have sold lesser and have had lesser performance as compared to the ones which aren't backordered. A plausible reason for this might be that the products which are backordered in this time period have also been backordered in the past continuously, hence leading to lesser

performance and lesser sales compared to the ones which aren't backordered.

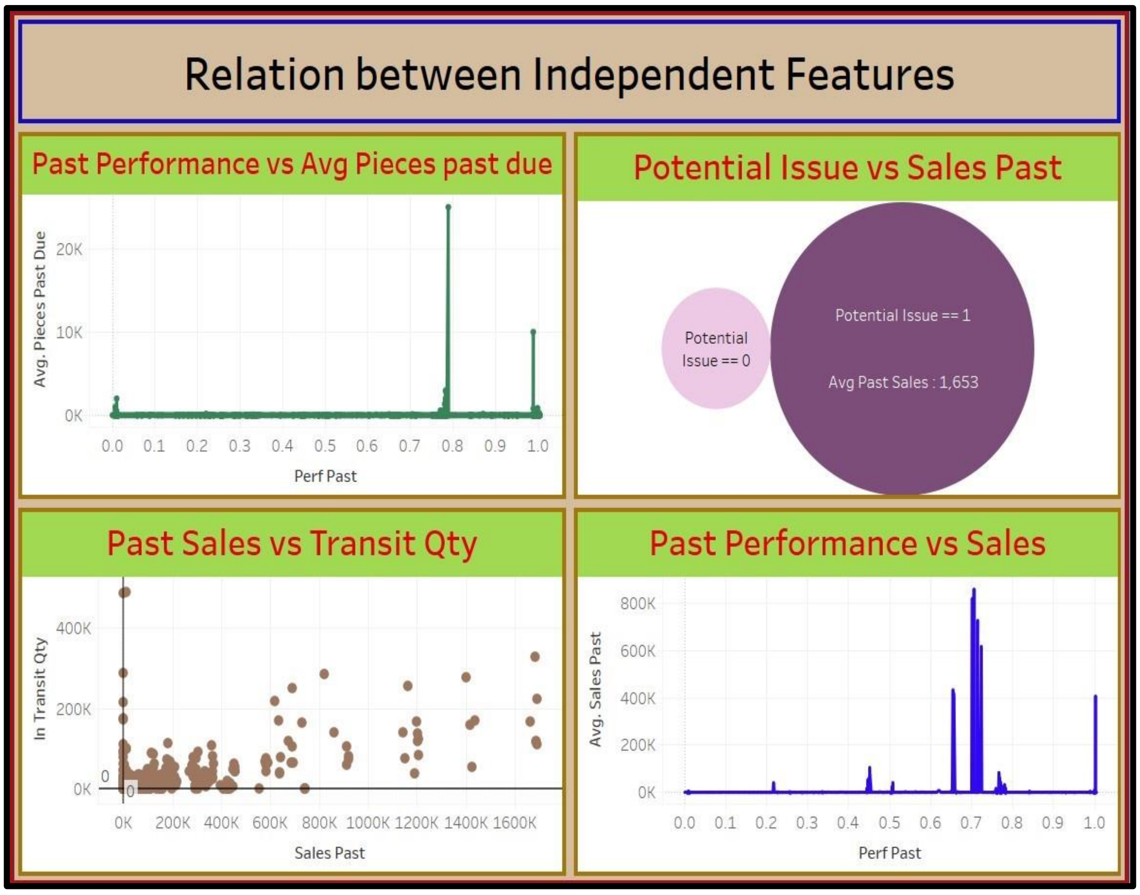
3. **National Inventory vs Backorder:** From the top right corner bar chart, we can easily notice that there's a significant difference between the average national inventories of backordered and non-backordered sku(s). The average national inventory of those which aren't backordered is way higher (493) than those which are backordered (25). There are some negative values also. Negative Inventory is caused by entering sales transactions before entering the corresponding purchase transactions, i.e., you sell inventory items that you do not have in stock. Out of total products that have negative value of "national_inv" 15% went on backorder, while overall only 0.66% products went on backorder.

4. **Lead Time vs Backorder:** In the bottom left corner graph, it seems like a bar chart, but it's a scatter plot with lines instead of points i.e., in tableau, each record is shown as a separate one, instead of aggregating like the other graphs. Because of this, it's easy to notice that there is an outlier in the backordered section. One of the backordered products has had a high lead time compared to others. So, there is a trend that the non-backordered items will have higher lead time than the backordered ones.

5. **Deck Risk vs Backorder:** This bottom center graph is an interesting plot to show that the backordered products have lesser deck risk than the ones that aren't backordered while we expect it to show the exact opposite characteristics.

6. **Local BO Qty vs Backorder:** From the last graph of the dashboard, the one in the bottom right corner, we are able to see that the local backorder quantity is way higher for the backordered products of this data than the non-backordered products. It's pretty obvious when we think about it, but still it's needed to check if the data follows the usual trend of that place/town/country.

2. <u>Relation between Independent Columns:</u>



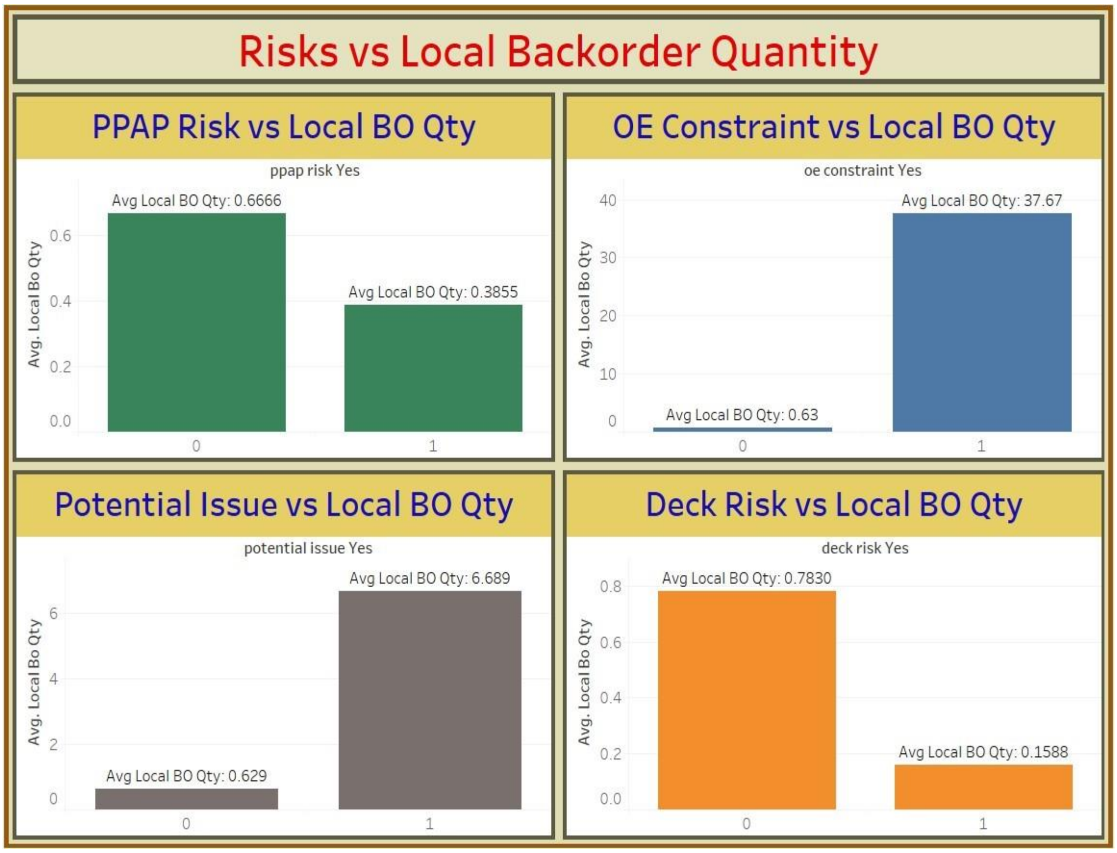
1. **Avg Past Performance vs Avg Pieces past due:** As we can see from the top left graph, the average pieces past due almost remains at 0 most of the times even if past performance has changed but there are a few peaks at a couple of points like around 0.8 and around 1. This means that the avg pieces past due has increased around the higher performances and that's because of the fundamental fact of backorder that as the past performance increases, demand also increases thus leading to backorder.

2. **Potential Issue vs Sales Past:** In the top right plot, the type of visualization done is called a bubble plot. Its size indicates the magnitude of the column in that section. From the plot, we can see the surprising fact that the sales for the products which have potential issue have been really high compared to those which don't have potential issue.

3. **Past Sales vs Transit Quantity:** We can see from the bottom left corner that the plot between past sales and transit quantity doesn't indicate much information, but there's one important observation to be made. There are 2-3 data points which have 0 past sales but have high value of transit quantity. One reason could be that the particular order might be a bulk order but there might be another reason for this; maybe the ones which were ordered from the source haven't been received for a long time now.

4. **Past Performance vs Past Sales:** In the bottom right plot, we can see that the past sales is almost zero till the performance reaches about 0.6. After that, the sales have been really good. So, we can say that 'sales' is correlated to 'performance'.

3. **Relation between Local BO Qty and Risks:**



1. **PPAP Risk vs Local BO Qty:** The top left graph is a bar chart between PPAP Risk and Local BO Quantity. This bar graph indicates that the average local backorder quantity is higher for products that **don’t** have any risk in PPAP (Production Part Approval Process). One of the ways to explain this might be that PPAP risk contributes to lesser performance, and those products which don’t have any PPAP risk, have better performance than the others that have PPAP risk=1. Since performance is high, there will be larger demand and lesser stock, which ultimately leads to backordering.

2. **OE Constraint vs Local BO Qty:** As we can see clearly from the top right visualization in this dashboard, the average local backorder quantity is way higher for those products with OE constraint than those with no OE constraint.

3. **Potential Issue vs Local BO Qty:** In the bottom left bar graph, we can see that there is a greater local backorder quantity for products with potential issue than those without potential issue. So, we can say that potential issue has a high impact on local backorder quantity and thus backordering. One possible way to explain this phenomenon would be: “If there’s a potential issue on an **ordered** item, that particular item will be removed from stock, but since it has already been ordered, it would be out of stock now and thus leads to backorder.

4. **Deck Risk vs Local BO Qty:** In the bottom right graph, we would be able to notice that there is a higher local backorder quantity for products with **no deck risk** compared to those with deck risk.

Feature Engineering

- The Dataset we have is highly imbalanced Data, with 99.34% data points in class 0 and only 0.66% data of class 1.

- There are 100893 Null Values in “lead_time” column.

- Columns “perf_6_month_avg” and “perf_12_month_avg” have most of values ranging from 0 to 1, but there are 1,20,000 plus data points for which “perf_6_month_avg” or “perf_12_month_avg” is ‘-99’. We interpreted this irregularity that the products for which the “perf_6_month_avg” or “perf_12_month_avg” was ‘-99’, this means that product was not supplied by the current source for the last 6 or 12 months, which means instead of “Null” values ‘-99’ was kept there.

- The Dataset has many Categorical variables, therefore we One Hot Encoded the data using the pandas getdummies() function.

- Chi Squared Test: To perform Feature Selection, we used Chi Squared Test between Categorical variables and the dependent variable (went_on_backorder_Yes) and dropped the variables having p value more than 0.02.

  Note: The Chi-square test of independence is a statistical hypothesis test used to determine whether two categorical or nominal variables are likely to be related or not. Here we used the Chi-Squared Test to quantify the relation between the categorical variables and the dependent variable.

- Next, we made a heatmap which displayed correlation among all features. Many features were correlated which we either dropped or merged some features together by taking average.

  i) "min_bank" was dropped as it was highly correlated with many features.

  ii) "forecast_3_month", "forecast_6_month", "forecast_9_month" were merged by taking average, and the new column formed was named as "forecast_sales".

  iii) Same was done with columns like "sales_x_month" and with columns like "perf_x_month_avg"

  iv) AVG("sales_x_month") = sales_past"

  v) AVG("perf_x_month_avg") = "perf_past"

  vi) The column "forecast_sales" was dropped, as it was highly correlated with "sales_past".

  vii) Next, we Imputed the Null values in "lead_time" using "Iterative Imputer" function of sklearn library.

  Note: Iterative Imputer uses a Regression Algorithm that is trained on all features other than the one that is Null and then that algorithm predicts the Null values. This process is done several times until the new predicted values are remarkably close to the previous predicted values.

  viii) Now we have Clean Data with No Null values in any of the columns, and where no two features are highly correlated, but still our data is highly imbalanced, and therefore it is unfit to be fed into any Machine Learning model.

Dealing with Imbalance in Dataset

The Dataset we have is highly imbalanced Data, with 99.34% data points in class 0 and only 0.66% data of class 1.

“went_on_backorder_Yes” in the Imbalanced Data.

0	:	1676567	1	:	11293

We did Random Under sampling of the Majority Class, after Under Sampling the class frequency was: -

0	:	800921	1	:	11293

Further to Increase the frequency of Minority Class, we used Synthetic Over Sampling method (SMOTE), after Over Sampling the frequency of classes was: -

0	:	800921	1	:	400460

Note: We did Under Sampling first and then Over Sampling, instead of directly Over Sampling the minority class, to prevent creation of false data at a large scale. If we directly did Over Sampling then we would have to Over Sample minority class from 11000 to 8 lacs or 16 lacs, which would have created a large amount of false data, which can reduce our model performance on test data.

So finally, the class 1 vs class 0 ratio was 0.5, which is decent enough to use data for training ML model.

Choosing Metrics:

We need to maximize “recall” score as we want our Model to identify all products that will go on backorder. So, our metrics would be “recall” and not “accuracy”.

Model Selection and Training

The Model we chose to work with in this particular project is “Random Forest”. We chose this particular model out of all others because:

- Many features in our Data have highly skewed distributions, having many outliers, and tree-based models are very robust to outliers.

- Tree based models work better for Imbalanced Data, as compared to linear models.

- Bagging technique, on which RandomForestClassifier is based, makes our model perform good enough on test data (unseen data).

Let’s now study look upon the modelling and features and hyperparameters of Random Forest in Detail.

Understanding the working of RandomForest

As mentioned earlier, RandomForest works on the Bagging principle.

Bagging:

Bagging, also known as **Bootstrap Aggregation** is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as **row sampling**. This step of row sampling with replacement is called **bootstrap**. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as **aggregation**.

Steps involved in random forest algorithm:

Step 1: In RandomForest, n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on **Majority Voting** for Classification respectively.

Significant Features of RandomForest

1. **Train-Test split-** In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

2. **Diversity-** Not all attributes/variables/features are considered while making an individual tree, each tree is different.

3. **Parallelization-**Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

4. **Stability-** Stability arises because the result is based on majority voting.

5. **Immune to dimensionality-** Since each tree does not consider all the features, the feature space is reduced.

Hyperparameters of RandomForest

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

For Increasing the Predictive Power:

- **n_estimators**– number of trees the algorithm builds before averaging the predictions.
- **max_features**– maximum number of features random forest considers splitting a node.
- **mini_sample_leaf**– determines the minimum number of leaves required to split an internal node.

For Increasing the Speed:

- **n_jobs:** it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
- **random_state:** controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
- **oob_score:** OOB means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

Summary of our Model and its Output on Test data

1. Summary:
   - i) Model – RandomForestClassifier
   - ii) Hyper parameters –
     - max_samples = 0.1
     - n_estimators = 800
   - iii) max_depth=10
   - iv) min_samples_split=100

2. Outcome on Test Data:
   - i) Recall Score: 0.643 = 64.3%
   - ii) Roc_Auc_Score: 0.791 = 79.1%
   - iii) Accuracy_Score: 0.937 = 93.7%

Reason for comparatively less Recall Score:

Even if we used a really good model like Random Forest Classifier, it's clearly visible that the recall score is abnormally low. This is mainly because of the undersampling and oversampling we did for rectifying the imbalanced data. Imbalanced data would have induced more errors and thus less Recall score. One might think Undersampling and Oversampling can't induce errors, but it's again predicting and inserting or deleting rows into the dataset which induces some errors when it comes to metrics. Secondly, even if we wanted to perform a GridSearchCV on this for more enhancement, we would need a super computer or a large amount of time for running a single cell. The Reason is that this dataset contains lakhs of rows and using a GridSearch on it is practically impossible with a normal laptop with 8 or 16 GB Ram as it crashes the kernel.

- Anand Rangwani
- Bhavesh Attarde
- Navya Yadav
- Niyati Srivastava
- Sriram V

Mechanical Exploratory Project Group
Under the guidance of Prof Dr. Prabhas Bharadwaj