[Project Name: Backorder Prediction]

Inception: High level Design(HLD)

Abstract:
Material backorder is a common supply chain problem under unpredictable demand and pertinent supply risk. We aim to use Machine Learning predictive models in the area of the business decision process. By predicting products backorder predictive models providing flexibility to the decision authority, better clarity of the process, and maintaining higher accuracy. The tree-based machine learning is chosen to predict material backorder the model. The backorders of products are predicted in this project using Random Forest (RF), sampling techniques and ensemble learning are employed in this particular task.

1.Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

1.1. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2. Project Overview

When a customer orders a product, which is not readily available due to lack of unavailability of the product in the store or not in the inventory but can guarantee the delivery of the ordered product a certain date in the future and the customer waits for the same. This scenario is called backorder of that specific product.

Backordering has good and bad faces in the business. So there has to be some balance between the demand and supply. ML Predictive analysis can schedule the production, supply chain and inventory helps to forecast product delivery delay which in return increase the customer satisfaction

If the back orders are not handled promptly it reflects high impacts on the company's revenue, share market price, and customer's trust that leads to losing the customer as well as the order. On the other hand, to satisfy backorders leads to enormous pressure on different stages of the supply chain which may break (exhaust) or can cause extra costs on production, shipment …etc.

Nowadays, most of the company's use Machine learning predictive analysis to predict products back orders to overcome the tangible and intangible costs of backorders.

We have performed some hypothesis tests considering backorder scenarios. The outcomes of the hypothesis's tests are helpful to choose the appropriate machine learning model for prediction.

3. Backorders vs. Out of Stock

An item is out of stock when the seller doesn't have the item in inventory and has no sure date to restock, or the item is seasonal or a limited run. Backordered items are expected to be available in a reasonable timeframe.

3.1. What Causes Backorders?

- **Unusual demand or demand exceeds supply:** Holiday seasons or other unforeseen circumstances, like extreme weather, may lead to unusual purchasing.
- **Inaccurate forecasting:** Less accurate forecasts could lead to low safety stock and a higher chance of backorders.
- **Supplier or manufacturing issue**: Supply chain challenges, such as factory shutdowns or raw material shortages, can lead to items being unexpectedly out of stock.
- **Delayed orders:** Companies that order based on safety stock formulas and require manual review of purchase orders may run into restocking delays, only to experience a sudden surge of orders.
- **Human errors:** An employee may enter an order as a backorder even if the item is available. Or, Warehouse management discrepancies: A glitch in an inventory management system may provide inaccurate data, or a breakdown in data entry may lead to stock being miscounted or misplaced. worse, a retailer may accept a backorder even though the item is out of stock.

3.2. Advantages of Backorders:

- **Offers market insights:** Backorders act like customer surveys, indicating what kinds of products buyers want, and when they're in the highest demand.
- **Improved cash flow:** Companies that avoid holding excess stock, with the associated costs, free up cash for other priorities. In some industries, less inventory also translates to reduced taxes.
- **Minimizing storage** and other inventory costs that come with holding extra stock.

3.3. Disadvantages of Backorders

- **Losing out on business:** Customers may not want to wait, or trust the company to fulfil their orders, causing them to cancel and purchase elsewhere.
- **Loss of market share:** If customers frequently encounter backorders or must wait a long time for fulfilment, their loyalty to your brand may wane, and they could turn to other brands.
- **Increased complexity:** Backorders increase the chances of a company having to resolve customer service issues, such as trying to update expired payment information.

3.4. How to avoid Backorders?

- A well planned Supply Chain Management, Warehouse management and inventory management can avoid or minimize Backorders to some extent.
- Manufacturers or suppliers can also hit peak demand. diversify suppliers or ordering from a wide variety of sources is good to some extent.
- Increasing inventory or stock of products is not a solution as it increases storage costs and extra costs means they have to be included in the product prices which might result in losing business to competitors**.**

4. Problem Statement:

- Classify the products whether they would go into Backorder (**Yes / No**) based on the available data from inventory, supply chain and sales.
- The task is to classifying whether a product would go to Backorder based on the given input data.
- The target variable to predict consists of two values, if it is "Yes" - the predicted product to be treated as Backorder. If the output is "No"- the predicted product to be not going to Backorder
- So we can have considered this is a **Binary Classification problem**.

5. Exploratory analysis

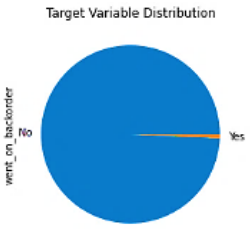
5.1Data Overview

- The dataset of this project work has been published in Kaggle (Currently the page is not available). The dataset is divided into the training and testing datasets.

Data Source: https://github.com/rodrigosantis1/backorder_prediction/blob/master/dataset.rar

- The dataset is highly imbalanced which should be addressed for accurate predictions by the model
- Each dataset contains 23 attributes with 1,687,862 and 242,077 observations for the training and testing sets, respectively.
- We have chosen inventory, lead time, sales, and forecasted sale as our predicting variables and 'went on backorder' as our response variable. Our target variable is labelled with two classes. Hence, this scenario falls under the binary classification problem.
- The inventory feature indicates an available stock of products, although it contains high numbers of negative records. The negative inventory may arise due to the machine or human error. It may also occur when a shipment is recorded as complete before it arrives.
- The 'lead time' feature indicates the elapsed time between the placement of products' orders
  and delivery of those products to the customers.
- The lead time in our dataset ranges from 0 to 52 weeks.
- The sales features are divided into four parts as one-month sale, three months' sale, six months' sale, and nine months' sale.

- The forecasted sale is divided into three columns showing the forecast of three months, six months, and nine months.
- It is observable that the data points of different features have many outliers with different ranges. In both training and testing datasets, a large number of missing values across the predicting variables are observed. Moreover, our response variable is highly imbalanced with 0.669% data from 'Yes' class, and 99.33% data from 'No' class.



- The data samples are distributed among two classes, where 0 indicates 'No' class or no backorder items and 1 indicates 'Yes' class or backorder items.
- Highly Class imbalance: Most of the samples did not went to back order (99.83%) of the time, while actual back orders occurs (0.17%) of the time.
- The data points which are responsible for extreme positive skewedness may not actually be outliers.

Suppose where the sales of a particular product are very high in the past 12 months. This implies that forecast for coming 6 and 9 months will be high. If forecast and sales is high this implies inventory and in transit quantity for the particular product would be high. Also the amount of sales would be very high for a few products compared to others. Considering these the positive skewedness present in the data may not truly represent outliers.

5.2. In the Train dataset we are provided with 23 columns(Features) of data.

1) sku — Random ID for the product
2) national_inv — Current inventory level for the part
3) lead-time — Transit time for product (if available)
4) in_transit_qty — Amount of product in transit from source
5) forecast_3_month — Forecast sales for the next 3 months
6) forecast_6_month — Forecast sales for the next 6 months
7) forecast_9_month — Forecast sales for the next 9 months
8) sales_1_month — Sales quantity for the prior 1-month time period
9) sales_3_month — Sales quantity for the prior 3-month time period
10) sales_6_month — Sales quantity for the prior 6-month time period
11) sales_9_month — Sales quantity for the prior 9-month time period
12) min_bank — Minimum recommend amount to stock
13) potential issue — Source issue for part indented
14) pieces_past_due — Parts overdue from source
15) perf_6_month_avg — Source performance for prior 6-month period
16) perf_12_month_avg — Source performance for prior 12-month period
17) local_bo_qty — Amount of stock orders overdue
18) deck risk — Part risk flag
19) oe_constraint — Part risk flag

20) ppap_risk — Part risk flag
21) stop_auto_buy — Part risk flag
22) rev_stop — Part risk flag
23) went_on_backorder — Product actually went on backorder. This is the target value.

- **From** the 23 features given in the dataset ***15 are numerical*** and ***8*** *(including the target variable)* ***are categorical features***. The first column 'sku' corresponds to product identifier which is unique for each data point in the dataset. So this feature can be dropped as it adds no value in output prediction.

```
In [8]: 1 data.describe()
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
national_inv	1929935.0	496.568259	29573.434344	-27256.0	4.00	15.00	80.00	12334404.0
lead_time	1814318.0	7.878627	7.054212	0.0	4.00	8.00	9.00	52.0
in_transit_qty	1929935.0	43.064397	1295.420493	0.0	0.00	0.00	0.00	489408.0
forecast_3_month	1929935.0	178.539864	5108.770174	0.0	0.00	0.00	4.00	1510592.0
forecast_6_month	1929935.0	345.465893	9831.562085	0.0	0.00	0.00	12.00	2461360.0
forecast_9_month	1929935.0	506.606748	14345.430866	0.0	0.00	0.00	20.00	3777304.0
sales_1_month	1929935.0	55.368164	1884.377009	0.0	0.00	0.00	4.00	741774.0
sales_3_month	1929935.0	174.663858	5188.855851	0.0	0.00	1.00	15.00	1105478.0
sales_6_month	1929935.0	341.565349	9585.030376	0.0	0.00	2.00	31.00	2146625.0
sales_9_month	1929935.0	523.577094	14733.265629	0.0	0.00	4.00	47.00	3205172.0
min_bank	1929935.0	52.776366	1257.968257	0.0	0.00	0.00	3.00	313319.0
pieces_past_due	1929935.0	2.016193	229.611242	0.0	0.00	0.00	0.00	146496.0
perf_6_month_avg	1929935.0	-6.899870	26.599879	-99.0	0.63	0.82	0.96	1.0
perf_12_month_avg	1929935.0	-6.462343	25.883429	-99.0	0.66	0.81	0.95	1.0
local_bo_qty	1929935.0	0.653704	35.432295	0.0	0.00	0.00	0.00	12530.0

5.3. Observations:

- The feature distribution is extremely right skewed because the features mean value is greater than 75th percentile value which depicts the same.
- Also the difference between the 75th percentile value and max value is for each feature is very high which might be due to the presence of outliers.
- The columns perf_6_month_avg and perf_12_month_avg has max. value as 1 and min. value as -99. It seems that the missing values are replaced with -99.
- Out of 1929937, only 13981 i.e., 0.5% of data is Out of Stock (Flag as Yes... as it went to Backorder), and more than 99.5% is in Stock (Flag as No). So its result bias.

6.Data Cleaning

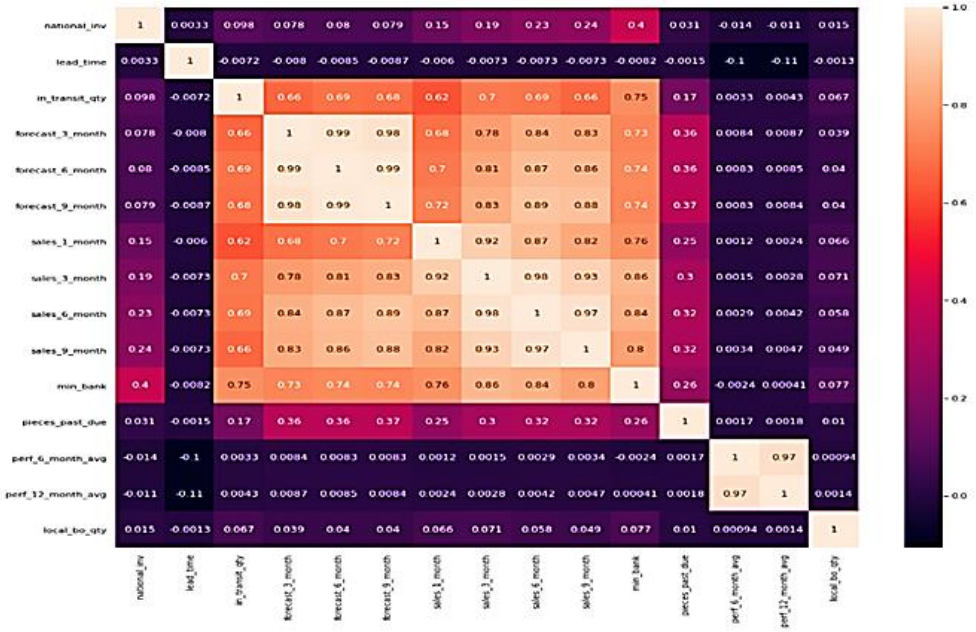
- Clean missing values.
- Dropped the first column 'sku' as it as an identifier and is unique for all the rows in dataset so dropped 'sku' column as it is random product id.
- Replace all categorical columns with 1 for yes and 0 for no.

- Scale the numerical columns**.**

6.1. Handling Missing values:

- Used model based imputation to impute the missing values
- Few null values observed in the column lead-time.
- Nan values observed in the last row of the training dataset it should be dropped.
- Replaced missing values from the categorical features ['national_inv','lead_time','in_transit_qty','forecast_3_month','forecast_6_month', 'forecast_9_month', 'sales_1_month','sales_3_month', 'sales_6_month', 'sales_9_month', 'min_bank','potential_issue', 'pieces_past_due', 'perf_6_month_avg', 'perf_12_month_avg', 'local_bo_qty', 'deck risk', 'oe_constraint','ppap_risk', 'stop_auto_buy', 'rev_stop', 'went_on_backorder'], dtype='object']
- Replaced all the categorical columns with 1 for ‘yes’ and 0 for ‘No’: [potential_issue, deck_risk, oe_constraint, ppap_risk, stop_auto_buy, rev_stop, went_on_backorder],

6.2. Correlation matrix



- All the significant correlations observed are positive.
- forecast_3_month, forecast_6_month and forecast_9_month are very strongly correlated with each other to a degree of 0.99.
- Forecast and sale columns are correlated with each other with a minimum degree of 0.62 varying up to 0.88. It is obvious that when the sales for a certain product is high in the past sales the forecast for the same in the coming months will be higher and vice versa.
- perf_6_month_avg and perf_12_month_avg are very highly correlated with each other to a degree of 0.97.

- min_bank (minimum amount of stock recommended) is highly correlated with sales and forecast columns as stock in inventory is directly proportional to sales.
- in_transit_qty is highly correlated with sales, forecast and min_bank columns. This is obvious because high sales of a product => more of that product in transport for inventory replenishing high sales of a product => high forecast.
- pieces_past_due is meekly correlated with sales and forecast columns.
- national_inv is meekly correlated with min_bank and weekly correlated with sale columns.
- Many features are correlated the linear models like logistic regression, Linear SVM and other linear models may not perform well as the coefficients of separating plane change.
- By checking VIF (Variance Inflation Factor) value between the correlated features the redundant features can be removed if needed or using PCA we can reduce dimensions if feature interpretability of model is not important
- Dropped 11 features from the dataset [national_inv, lead_time, sales_1_month, pieces_past_due, perf_6_month_avg, local_bo_qty, deck_risk, oe_constraint, ppap_risk, stop_auto_buy, rev_stop].

6.3. Change scale of data

- Machine learning algorithms perform better when numerical input variables are scaled to a standard range**.** Used MinMaxScalere.

6.4. Handling class imbalance

- From EDA it can be concluded that dataset is extremely right skewed and also is highly imbalanced. Class imbalance can be handled in two ways. Balancing the dataset using different Oversampling or Under sampling techniques or while fitting model itself different ensembles can fit their base learners on balanced sub-samples.

6.4.1. Under sampling techniques

- **Imblearn** library helps and provides best implementation for different sampling techniques also different classifiers which tackle the problem of class imbalance.
- To resolve the imbalanced class problem, we used ***Nearmiss Under sampling*** on the target class is selected.

6.4.2. Near miss under sampling

- In this method the algorithm starts by calculating distances between all instances of majority and minority classes. There are 3 variations in this method. **Near miss — 1:** Choses the majority class instances whose avg. distance is smallest to their minority class instances. **Near miss -2:** Same as near miss — 1 but farthest instead of smallest. **Near miss — 3:** picks the given number of instances from majority class which are closest to minority class.

7. Splitting the Dataset

- Split whole data into train and test (80%–20%)

8. Hyper Parameter Tuning (Modelling)

- RandomizedSearchCV used for Hyper parameter tuning is choosing a set of optimal hyper parameters for a learning algorithm. A hyper parameter is a model argument whose value is set before the learning process begins. ***Used random grid***. The key to machine learning algorithms is hyper parameter tuning. The optimal hyper parameters help to avoid under-fitting and over-fitting.
- Training and test sets were split using an 80:20 proportion, generating 50 different combinations. A stratified 2-fold cross validation scheme is adopted in training to avoid overfitting in training.
- Random search of parameters, using 2-fold cross validation, search across 50 different combinations.

8.1. Random Forest

- Random forest estimators are used because it performs well in the imbalanced dataset.
- Robust to outliers and the skewed predictors
- Random forest is a tree based ensemble built using a bootstrap sample, where training set batches are drawn with replacement.
- During the construction of the tree, the split selected is the best among a random subset of the attributes leading to a randomness that leans the performance of the forest over a single non-random tree. The bias increase is offset by averaging variance decrease, achieved by the combination of the probabilistic prediction of the base classifiers.
    - ***Model Accuracy: 89.39***

8.1.1 Predicting.

Checking For yes.

```
[64]: test=[[0.002206,1.000000,0.000065,0.0,0.0000,0.00008,1.0,0.0,1.0,0.0,0.0]]

[65]: answer=rdf_clf.predict(test)
      print(answer)=[[0.002206,1.000000,0.000065,0.0,0.0000,0.00008,1.0,0.0,1.0,0.0,0.0]]answer)

[1.]

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 495 out of 495 | elapsed:    0.0s finished
```

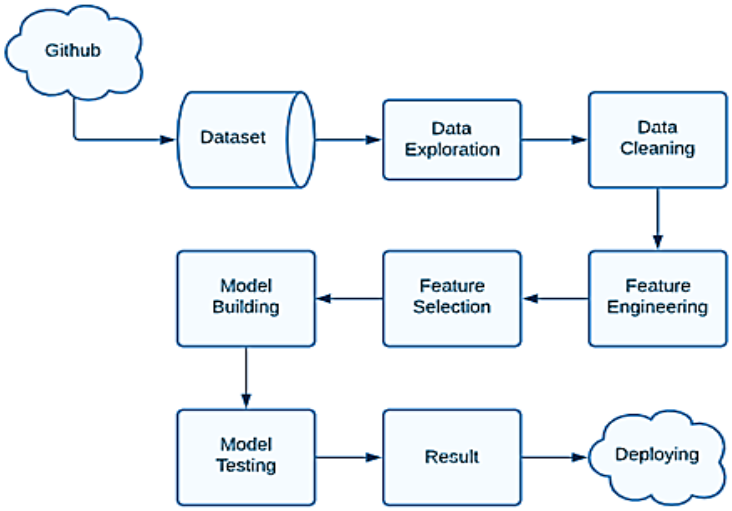
Checking For NO

```
[66]: test1=[[0.002205,0.153846,0.000000,0.0,0.0000,0.0,0.0,0.0,0.0,1.0,0.0]]

[67]: answer=rdf_clf.predict(test1)
      print(answer)

[0.]
```

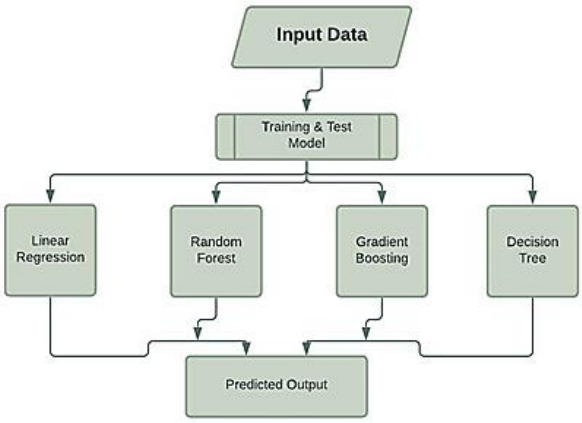
9. Future Work

- Business experts can suggest or with more other effective features.
- Try with more complex ensemble models.
- Deep Learning models tends to works better, we can fine tune the hyper parameters of the existing architecture or can come up with different deep learning architecture.

10. Working procedure of proposed model.

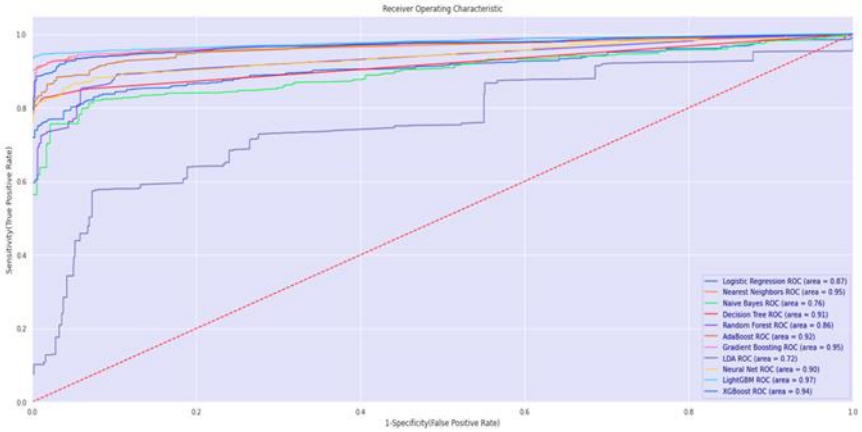


11. Model Training and Evaluation



12. Results of different models applied

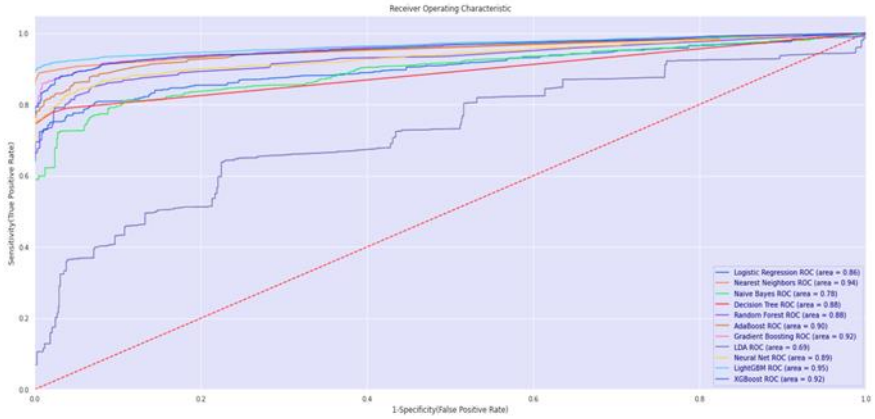
12.1. Applying different classifiers on normal data



Result of models applied on normal data

	Classifiers	Accuracy Scores	F1_Score
9	LightGBM	0.9664	0.9663
6	Gradient Boosting	0.9522	0.9521
1	Nearest Neighbors	0.9500	0.9499
10	XGBoost	0.9374	0.9372
5	AdaBoost	0.9230	0.9228
3	Decision Tree	0.9062	0.9056
8	Neural Net	0.9015	0.9008
0	Logistic Regression	0.8716	0.8706
4	Random Forest	0.8603	0.8594
2	Naive Bayes	0.7598	0.7456
7	LDA	0.7178	0.7173

12.2. Based on EDA with reduced features



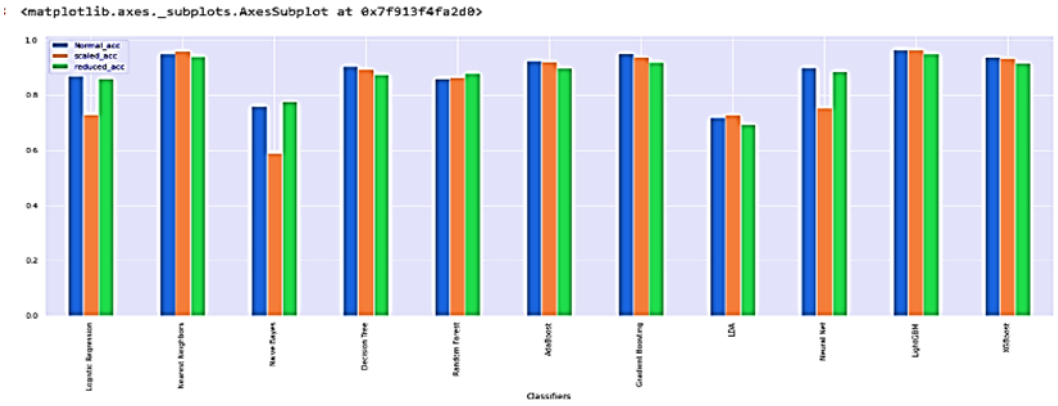
Reduced feature based on EDA of numerical variable

	Classifiers	Accuracy Scores	F1_Score
9	LightGBM	0.9491	0.9490
1	Nearest Neighbors	0.9409	0.9408
6	Gradient Boosting	0.9203	0.9202
10	XGBoost	0.9166	0.9163
5	AdaBoost	0.8991	0.8988
8	Neural Net	0.8858	0.8851
4	Random Forest	0.8809	0.8800
3	Decision Tree	0.8754	0.8744
0	Logistic Regression	0.8617	0.8606
2	Naive Bayes	0.7764	0.7653
7	LDA	0.6928	0.6908

Metric(s):

- ROC AUC Score: AUC score signifies a model's ability to differentiate between positive and negative classes. If for a model 1 AUC > model 2 AUC then at most of the threshold values model 1 is able to identify the positive class better than negative class. So it makes a good metric for the problem.
- Area under Precision - recall curve: AUC of Pr-Re curve is very important as the business needs to select threshold based on trade off b/w precision and recall
- Macro F1-Score: Macro F1 score is average of F1 scores of both positive and negative classes. As class imbalance is present to evaluate the model F1 scores of both classes needs to be considered.

12.3. Combined Performance Plot



13.Document Tracking

The following chart is used to log of all changes made to this document.

Version	Date of edit/change	Who made the edit/change	Description of edit/change
HLD Ver-1.0	20th Aug 2021	Vincent Thomas	Initial Version
HLD Ver- 1.1	25th Sep 2021	Vincent &Kashit	Modeling, Model prediction
HLD Ver- 1.2	1st Nov 2021	Vincent & Kiran	Model Performance& deployment

13.Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, Seaborn, Malplotlib are used to build the whole model.

- PyCharm is used as IDE.
- Juypter notebook is used as IDE.
- Matplotlib, Seaborn, Plotly are used for visualization of plots.
- Flask used as API.
- Cassandra is used to retrieve, insert, delete and update the customer input database.
- Front end development is done using HTML.
- GitHub used as version control.

14.Deployment: - The Model Deployed on to HEROKU.

15.Data Base: - Cassandra DB is used to retrieve, insert, delete and update the customer input database. **Flask** is a micro web framework written in python. Here we use the flask to create an API that allows sending data, and receive a prediction as a response. Flask API's act as an interface between the user, ML- Model and DB. Users interact with the deployed model on HEROKU and the result will be returned to the user by API's. The Flask API will collect all the customer input data store in Cassandra DB.

16.Reusability: - The written code can be reused and has the possibility to be altered with the help of business experts with more effective features**.**

17.Conclusion: - This project has been shown to identify those products that will be backordered based on certain features from the known data. The results show that a predictive machine learning classification can control the inventory system, which helps to reduce the pressure of the supply chain. It results in greater flexibility in inventory control and better customer satisfaction at a very low cost. Models like Light GBM show the highest accuracy score. (KNN) K Nearest Neighbor model also has a good accuracy score**.**

18.Reference: -

https://www.wikipedia.org,ResearchGateConferencePaper,https://www.journals.elsevier.com/information-sciences,google.com for images

Project partners backorder Prediction.

1. *Mr. Anurag Bisen. (Low level document (LLD), Model Exploration, Testing and evaluation)*
2. *Mr. Kashit Duhan. (Data collection and labeling, Model Exploration, Model refinement, HLD)*
3. *Mr. Kiran G. (Model Exploration, Testing and evaluation, Model deployment, HLD)*
4. *Mr. Sachin Kunchanur. (Detail Project Report(DPR), Model deployment, Testing and evaluation)*
5. *Mr. Shashank Bhat. (Low level document, planning and project setup, Model exploration)*
6. *Mr. Suraj B R. (Detail Project report, Testing and evaluation)*
7. *Ms. Tejaswini Ahirkar. (wireframe Document, Model Exploration, Testing and Evaluation)*
8. *Mr. Vincent Thomas Varghese. (Planning and project setup, Data collection labelling, High Level Document (HLD),*