¡Neuron

High Level Design (HLD)

Contents

Abstract

The efficient management of backorders is crucial for streamlined planning and resource allocation in production, planning, and transportation. By leveraging the wealth of data generated by Enterprise Resource Planning (ERP) systems, including historical and structured data, a predictive model can be developed to forecast backorders and facilitate initiative-taking planning. This project aims to classify products as either going into backorder (Yes) or not (No) based on past data from inventories, supply chain, and sales. The approach involves various stages, including data exploration, data cleaning, feature engineering, model building, and model testing using classical machine learning techniques. Different machine learning algorithms will be evaluated to determine the most suitable approach for this specific case. The goal is to develop a solution that can accurately predict backorder sales for individual products using the provided dataset. The successful implementation of such a model would enable businesses to anticipate and prepare for backorders, optimizing their operations and minimizing disruptions caused by unexpected stockouts.

Introduction

In today's fast-paced business environment, managing backorders efficiently is essential to ensure smooth operations across various aspects of the supply chain, including production, logistics, and transportation. Backorders, which occur when customer demand exceeds available inventory, can lead to delays, customer dissatisfaction, and increased costs. To mitigate these challenges, businesses can leverage the vast amount of data generated by their Enterprise Resource Planning (ERP) systems, which encompass historical and structured data, to develop predictive models that anticipate backorders.

The objective of this project is to build a solution that accurately predicts the likelihood of products going into backorder based on past data related to inventories, supply chain activities, and sales. By employing classical machine learning techniques, including data exploration, data cleaning, feature engineering, model building, and model testing, we aim to develop a robust predictive model that can effectively classify products as either going into backorder (Yes) or not (No).

The process begins with comprehensive data exploration, where we examine and analyze the available dataset to gain insights into the patterns, trends, and characteristics of the data. This step helps us identify any data inconsistencies, missing values, or outliers that need to be addressed during the data cleaning phase. By ensuring data integrity and completeness, we can build a reliable predictive model that produces accurate results.

Next, we focus on feature engineering, which involves transforming the raw data into meaningful features that capture the underlying patterns and relationships within the dataset. This step may include feature selection, extraction, and transformation techniques to enhance the predictive power of the model. By leveraging domain knowledge and understanding the specific requirements of the backorder prediction task, we can engineer features that effectively capture the factors influencing backorders.

With the processed and engineered dataset, we proceed to the model building phase. Here, we explore various machine learning algorithms that are suitable for the backorder prediction task. We consider algorithms such as Random Forest, Gradient Boosting, Decision Trees, K-Nearest Neighbors, and Logistic Regression, among others. By evaluating the performance of these algorithms using appropriate evaluation metrics, such as accuracy, precision, recall, and area under the ROC curve, we can identify the most effective model for our specific case.

Once the model is built, we rigorously test its performance using a separate test dataset. This testing phase enables us to assess the model's ability to generalize and make accurate predictions on unseen data. By comparing the predicted backorder classifications with the actual backorder status, we can measure the model's accuracy and evaluate its performance in terms of true positives, false positives, true negatives, and false negatives. These metrics provide valuable insights into the model's effectiveness and enable us to fine-tune it further if necessary.

The successful implementation of a reliable backorder prediction model offers significant benefits to businesses. It enables proactive planning by anticipating potential backorders, allowing for optimized inventory management, efficient resource allocation, and enhanced customer satisfaction. By streamlining production, logistics, and transportation processes, organizations can reduce costs, minimize disruptions, and maintain a competitive edge in the market.

Why this High-Level Design Document?

The High-Level Design Document serves as a blueprint or roadmap for the development of a predictive model to forecast backorders. It outlines the overall approach, methodology, and steps involved in building the solution. There are several reasons why creating a High-Level Design Document is beneficial:

1. Clarity of Vision

2. Structured Approach

3. Communication and Collaboration

4. Risk Identification and Mitigation

5. Resource Planning and Allocation

6. Scalability and Future Considerations

7. Documentation and Knowledge Sharing

Scope

1. Data Exploration: Analyzing the available data from ERP systems, including structured data related to inventories, supply chain, and sales.
2. Data Cleaning: Preprocessing the data to handle missing values, outliers, and inconsistencies
3. Feature Engineering: Creating relevant features from the available data that can contribute to predicting backorders
4. Model Building: Selecting and implementing appropriate machine learning algorithms that are well-suited for the backorder prediction task. This includes training the models on the historical data, tuning hyperparameters, and evaluating their performance.
5. Model Testing: Assessing the predictive accuracy and performance of the trained models using appropriate evaluation metrics
6. Solution Deployment: Integrating the trained predictive model into a deployable solution that can take input data for a specific product and provide a prediction of whether it is likely to go into backorder or not.

Definitions

1. Backorder: Backorder refers to a situation where a requested product is temporarily out of stock or unavailable and is ordered to be fulfilled at a later time when it becomes available again.
2. Predictive Model: A predictive model is a statistical or machine learning model that is trained on historical data to make predictions or forecasts about future events or outcomes. In the context of backorder prediction, a predictive model uses historical data from inventories, supply chain, and sales to classify products as either going into backorder or not.
3. Data Exploration: Data exploration is the process of analyzing and understanding the available data to gain insights and identify patterns.
4. Data Cleaning: Data cleaning, also known as data preprocessing, is the process of preparing and transforming the raw data to address any quality issues or inconsistencies.
5. Feature Engineering: Feature engineering involves creating new features or transforming existing features in the dataset to improve the predictive power of the model.
6. Model Building: Model building refers to the process of selecting and implementing machine learning algorithms to develop a predictive model.
7. Model Testing: Model testing involves assessing the performance and accuracy of the trained predictive model using appropriate evaluation metrics.
8. Solution Deployment: Solution deployment involves integrating the trained predictive model into a deployable system or application.

General Description

The backorder prediction solution aims to address the challenges associated with managing inventory, supply chain, and sales by leveraging data from ERP systems. The solution utilizes historical data to develop a predictive model that can classify products as either going into backorder or not. By accurately forecasting backorders, businesses can streamline their planning processes, mitigate unexpected strains on production, logistics, and transportation, and ensure efficient operations.

The solution follows a classical machine learning approach, involving various stages such as data exploration, data cleaning, feature engineering, model building, and model testing. The first step is to explore the available data, analyze its characteristics, and gain insights into the relationships between different variables. Data cleaning techniques are then applied to address any data quality issues, such as missing values, outliers, or inconsistencies.

Feature engineering techniques are employed to create new features or transform existing ones that capture relevant information for predicting backorders. These features can include inventory levels, historical sales data, supplier information, lead time, and other relevant factors that may impact product availability.

The next stage involves selecting and implementing machine learning algorithms suitable for the backorder prediction task. Different algorithms, such as decision trees, random forests, logistic regression, or neural networks, are experimented with to identify the best-performing model. The model is trained on the historical data, and its parameters are tuned to optimize its predictive capabilities.

Once the model is trained, it undergoes rigorous testing to evaluate its performance and accuracy. Testing involves splitting the data into training and testing sets to assess how well the model generalizes to unseen data. Various evaluation metrics, such as accuracy, precision, recall, and F1 score, are used to measure the model's performance.

The final deliverable of the solution is a deployable system or application that allows users to input relevant data for a specific product and obtain predictions about its likelihood of going into backorder. The solution can be integrated with existing ERP systems or used as a standalone tool to assist in decision-making related to inventory management, supply chain optimization, and sales forecasting.

Technical Requirements

Data Requirements

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

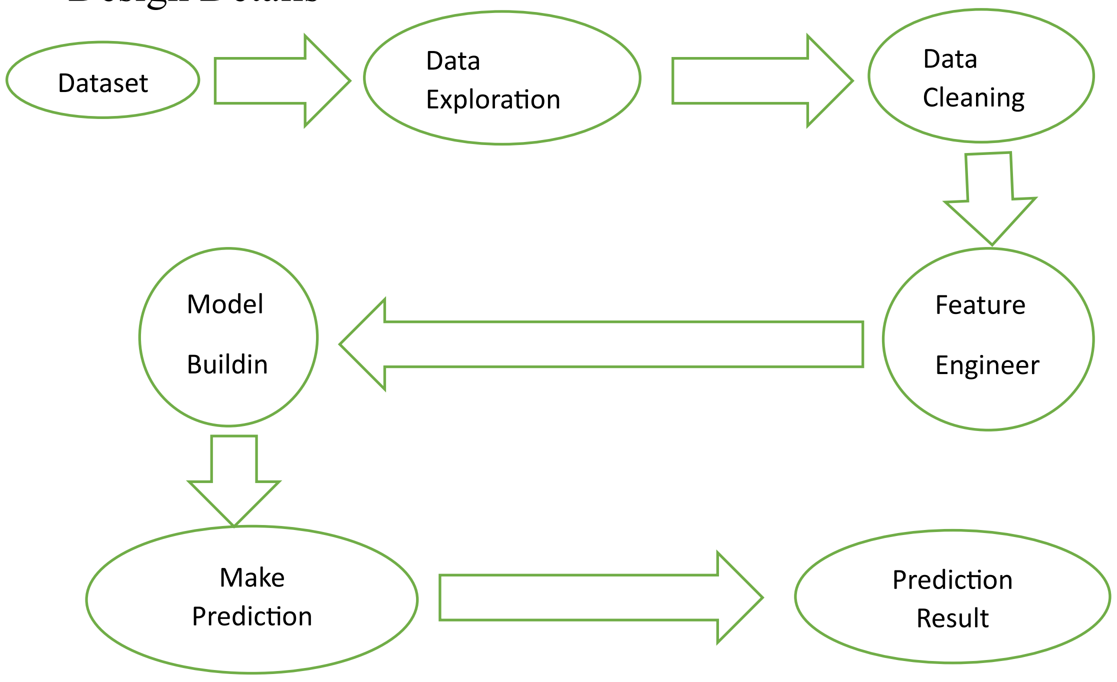
Tools used

- Programming Languages: Python
- Data Analysis and Machine Learning Libraries: Pandas, scikit-learn, TensorFlow
- Data Visualization: Matplotlib, Seaborn
- Integrated Development Environments (IDEs): Jupyter Notebook

Hardware Requirements

- RAM: 8 GB or more is recommended
- Graphics Processing Unit (GPU)

Design Details



Process Flow

1. Data Collection
2. Data Exploration
3. Data Cleaning and Preprocessing
4. Feature Engineering
5. Model Development
6. Model Evaluation
7. Model Deployment
8. Model Maintenance and Iteration

Model Training and Evaluation

Data Split:

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,train_size=0.7)

Model Selection:

RandomForestClassifier classifier

AUC Score is : 0.5420778807268269

# Use SMOTE to oversample the minority class

oversample = SMOTE()

X_train, y_train = oversample.fit_resample(X_train, y_train)

[[285366 2021]

[1283 814]]

0.9885865885506625

precision recall f1-score support

0 1.00 0.99 0.99 287387

1 0.29 0.39 0.33 2097

accuracy 0.99 289484

macro avg 0.64 0.69 0.66 289484

weighted avg 0.99 0.99 0.99 289484

AUC Score is : 0.6905706260390485

oversample = SMOTE()

X_train, y_train = oversample.fit_resample(X_train, y_train)

model1 = DecisionTreeClassifier()

model2 = KNeighborsClassifier()

model3 = LogisticRegression()

Accuracy score: 0.9603259592930871

AdaBoostClassifier(algorithm = "SAMME.R", learning_rate = 0.1)

clf.fit(X_train,y_train)

AUC Score is : 0.8249221469625202

XGBClassifier

Gradient Boosting algorithm at lr 0.05 AUC Score is 0.7596349490695469

Gradient Boosting algorithm at lr 0.075 AUC Score is 0.762278458693953

Gradient Boosting algorithm at lr 0.1 AUC Score is 0.7661929105152597

```
Gradient Boosting algorithm at lr 0.25 AUC Score is 0.790908793163794
Gradient Boosting algorithm at lr 0.5 AUC Score is 0.80424728202226
Gradient Boosting algorithm at lr 0.75 AUC Score is 0.7722288123598609
Gradient Boosting algorithm at lr 1 AUC Score is 0.7801860399564
Gradient Bosssting algorithm at septh of 10 AUC Score is 0.806347139930128
Gradient Bosssting algorithm at septh of 15 AUC Score is 0.7849763335231996
Gradient Bosssting algorithm at septh of 20 AUC Score is 0.7571819702628692
Classifier KNN
AUC Score is : 0.5554565885819278
Classifier Random Forest
AUC Score is : 0.9002789185259485
Classifier Naive Bayes
AUC Score is : 0.5840135289416875
Classifier StackingClassifier
AUC Score is : 0.5554565885819278
Classifier KNN
AUC Score is : 0.5544302118345903
Classifier Random Forest
AUC Score is : 0.8991110966217853
Classifier Naive Bayes
AUC Score is : 0.5811899240664249
Classifier StackingClassifier
AUC Score is : 0.5544302118345903
Classifier KNN
AUC Score is : 0.5609373776731992
Classifier Random Forest
AUC Score is : 0.8994388321620667
Classifier Naive Bayes
AUC Score is : 0.5609373776731992
Classifier KNN
AUC Score is : 0.5540610136250124
Classifier Random Forest
```

```
AUC Score is : 0.9018367027462328

Classifier Naive Bayes

AUC Score is : 0.5819578326138359

Classifier StackingClassifier

AUC Score is : 0.5540610136250124

Classifier KNN

AUC Score is : 0.5579763274715996

Classifier Random Forest

AUC Score is : 0.9004846231457531

Classifier Naive Bayes

AUC Score is : 0.5820604816550243

Classifier StackingClassifier

AUC Score is : 0.5579763274715996


Model: "sequential_1"

_________________________________________________________________

Layer (type) Output Shape Param #

=================================================================

dense_3 (Dense) (None, 128) 2944

activation_2 (Activation) (None, 128) 0

dropout_2 (Dropout) (None, 128) 0

dense_4 (Dense) (None, 64) 8256

activation_3 (Activation) (None, 64) 0

dropout_3 (Dropout) (None, 64) 0

dense_5 (Dense) (None, 1) 65

=================================================================

Total params: 11,265

Trainable params: 11,265

Non-trainable params: 0

_________________________________________________________________

Epoch 1/10

25446/25446 [==============================] - 79s 3ms/step - loss: 46.4291 -
```

```
accuracy: 0.5004
Epoch 2/10
25446/25446 [==============================] - 81s 3ms/step - loss: 0.6981 -
accuracy: 0.4999
Epoch 3/10
25446/25446 [==============================] - 81s 3ms/step - loss: 0.6978 -
accuracy: 0.4997
Epoch 4/10
25446/25446 [==============================] - 82s 3ms/step - loss: 0.6961 -
accuracy: 0.5003
Epoch 5/10
25446/25446 [==============================] - 84s 3ms/step - loss: 0.6958 -
accuracy: 0.4996
61
Epoch 6/10
25446/25446 [==============================] - 86s 3ms/step - loss: 0.6942 -
accuracy: 0.4995
Epoch 7/10
25446/25446 [==============================] - 80s 3ms/step - loss: 0.6947 -
accuracy: 0.5001
Epoch 8/10
25446/25446 [==============================] - 88s 3ms/step - loss: 0.6967 -
accuracy: 0.4997
Epoch 9/10
25446/25446 [==============================] - 83s 3ms/step - loss: 0.7017 -
accuracy: 0.5002
Epoch 10/10
25446/25446 [==============================] - 83s 3ms/step - loss: 0.7037 -
accuracy: 0.5005
9047/9047 [==============================] - 14s 1ms/step
Test set performance:
[[ 21 287366]
[ 0 2097]]

0.007316466540465103

precision recall f1-score support

0 1.00 0.00 0.00 287387

1 0.01 1.00 0.01 2097

accuracy 0.01 289484

macro avg 0.50 0.50 0.01 289484

weighted avg 0.99 0.01 0.00 289484

[116]: ff = model.predict(X_test)

9047/9047 [==============================] - 13s 1ms/step

[117]: roc = roc_auc_score(y_test,ff)

print("AUC Score is :", roc)

AUC Score is : 0.500036536099406
```

conclusion

RandomForestClassifier with SMOTE oversampling approach yielded the best results, achieving an AUC score of 0.6905706260390485. The ensemble approach and Gradient Boosting also showed decent performance with AUC scores of 0.6635556190882292 and 0.806347139930128, respectively. However, the neural network model performed poorly in this case.

Reference

[1] M. Shajalal, A. Boden, and G. Stevens, “Explainable product backorder prediction exploiting CNN: Introducing explainable models in businesses,” *Electron. Mark.*, vol. 32, no. 4, pp. 2107–2122, 2022, doi: 10.1007/s12525-022-00599-z.

[2] R. B. De Santis, E. P. De Aguiar, and L. Goliatt, “Predicting material backorders in inventory management using machine learning,” *2017 IEEE Lat. Am. Conf. Comput. Intell. LA-CCI 2017 - Proc.*, vol. 2017-November, no. November, pp. 1–6, 2018, doi: 10.1109/LA-CCI.2017.8285684.

[3] C. Ntakolia, C. Kokkotis, P. Karlsson, and S. Moustakidis, “Scopus - Document details - An explainable machine learning model for material backorder prediction in inventory management,” 2021, [Online]. Available: https://scopus.upc.elogim.com/record/display.uri?eid=2-s2.0-85119934595&origin=resultslist&sort=plf-f&src=s&st1=machine+learning+predictive+model&nlo=&nlr=&nls=&sid=dfcf96eeb76e43ebe428a3faeb38d5b7&sot=b&sdt=cl&cluster=scopubyr%2C%222021%22%2Ct%2C%222020%22%2Ct%2C%222019%22%2Ct%2Bscosubtype%2C%22ar%22%2Ct%2Bscosubjabbr%2C%22COMP%22%2Ct%2C%22ENGI%22%2Ct%2C%22SOCI%22%2Ct&sl=48&s=TITLE-ABS-KEY%28machine+learning+predictive+model%29&relpos=76&citeCnt=0&searchTerm=&featureToggles=FEATURE_NEW_DOC_DETAILS_EXPORT:1

[4] G. Lopez, Y. Jesus, and J. Panduro, “Use of a Machine Learning Model for the Reduction of Backorders in the Cross Docking Sales Process for the Homecenter Order Service,” 2022.