Table of content

Abstract

Predicting backorders is efficient inventory management and supply chain optimization, leading to cost reduction and improved customer satisfaction. The project aims to develop a robust backorder prediction model using logistic regression, decision trees, and random forests. By combining these models which helps in enhancing the accuracy and reliability of predictions. The model is trained on a detailed dataset comprising inventory and sales data, analyzing historical sales and inventory to uncover patterns associated with backorders. The Key features include forecasted demand, inventory levels, lead times, and sales performance. Each algorithm contributes unique strengths such has logistic regression for binary classification, decision trees for understandability and random forests for improved accuracy. Adding ROC-AUC Curve analysis provides deeper insights into model performance. Combining predictions from all models enhances overall performance, enabling businesses to manage inventory effectively, predict backorders, and take preventive strategies such as adjusting inventory levels or expediting shipment. The model provides a valuable tool for optimizing inventory and enhancing operational efficiency.

CHAPTER 1

Introduction

Managing inventory is crucial for an effective supply chain. Predicting backorders is essential to maintain high service levels and keep costs low. A backorder occurs when demand for a product exceeds supply, causing delays in fulfilling customer orders. Backorders can hurt customer satisfaction, increase costs, and disrupt the flow of goods. The main goal is to create a model that accurately predicts backorders so businesses can manage their inventory better and avoid stockouts. To do this, a detailed dataset is used, including sales figures, inventory levels, forecasted demand, and other supply chain metrics. This data is stored and managed using SQLite3, a simple and efficient database system that makes data easy to access and manage throughout the project.

The data features include:

- Forecast_3_month, Forecast_6_month, Forecast_9_month: Predicted demand for the next 3, 6, and 9 months.
- In_transit_qty: Quantity of products currently in transit.
- Lead_time: Time taken to restock products.
- Local_bo_qty: Quantity of local backorders.
- Min_bank: Minimum stock level to be maintained.
- National_inv: National inventory levels.
- Perf_12_month_avg, Perf_6_month_avg: Performance metrics for the past 12 and 6 months.
- Pieces_past_due: Number of pieces that are past due.
- Sales_1_month, Sales_3_month, Sales_6_month, Sales_9_month: Sales data for the past 1, 3, 6, and 9 months.

Several machine learning algorithms are used. Logistic regression predicts the likelihood of backorders, useful for yes/no decisions. Decision trees show how decisions are made and are easy to understand. Random forests combine many decision trees to improve prediction accuracy and reduce errors. PCA (Principal Component Analysis) is used for dimensionality reduction, helping to simplify the dataset while retaining important information. Combining these models with a hard voting method uses the strengths of each to improve overall performance. Hard voting means taking predictions from all models and deciding the final output based on the majority vote. This approach reduces the weaknesses of individual models and takes advantage of their combined strengths. To address the issue of imbalanced data, the Oversampling Technique (SMOTE) is used. SMOTE creates synthetic samples of the minority class to balance the dataset, ensuring that the machine learning models have enough data to accurately predict backorders.

Accurate backorder predictions lead to better inventory planning, lower holding costs, and higher customer satisfaction. Businesses can avoid stockouts, allocate resources better, and have a more responsive supply chain. Data preparation involves cleaning and organizing the dataset for training the machine learning models. Feature selection identifies the most important features for predicting backorders. Model training involves teaching the selected algorithms using the prepared dataset. Evaluation metrics like accuracy, precision, recall, and F1-score measure the performance of the models.

This approach provides a useful tool for improving inventory management and operational efficiency in supply chain management. The predictive model developed can help businesses improve their inventory practices and gain a competitive edge.

CHAPTER 2

Literature survey

Learning about supply chain and inventory management has been really interesting. AI and machine learning can be game-changers. Research papers show how predictive analytics has evolved, especially in predicting and handling backorders. These technologies can help stop problems before they even start, making businesses run better.

It began with a foundational study on backorder prediction in inventory management. This research evaluated various classification techniques using metrics like ROC-AUC and PR-AUC [5]. It proposed a hybrid modeling approach combining ensemble techniques, effectively handling imbalanced datasets while improving interpretability and reducing false positives and negatives[1][8][9]. This laid a solid foundation for understanding the complexities of backorder prediction and its significant impact on supply chain efficiency [2].

Another compelling study focused on logistic regression and decision tree models. Logistic regression, valued for its simplicity and interpretability, was used to predict backorder likelihoods based on factors like inventory levels, demand forecasts, and supply chain disruptions[3][5]. Its straightforward nature provided actionable insights for decision-makers, highlighting the importance of simplicity in predictive modeling [3].

Decision tree models used historical data to pinpoint critical factors leading to backorders. Their visual and interpretable nature proved valuable for stakeholders seeking to understand and address the root causes of stock outs[11][14]. This study showed how decision trees offer transparency in decision-making processes, helping to identify and fix issues.

Further research advocated for deep learning methodologies to identify order statuses in complex supply chains. These models excelled in predicting late orders by capturing intricate temporal dependencies and subtle patterns. Rigorous five-fold cross-validation validated their effectiveness, with some models achieving near-perfect accuracy [12][13][14]. This study highlighted deep learning's potential to revolutionize supply chain risk management and real-time decision support.

Inspired by these diverse methodologies, a robust model for predicting backorders in inventory management was developed. The model integrates four distinct approaches: logistic regression, decision tree, random forest, and PCA (Principal Component Analysis). A hard voting ensemble method was employed, where the final prediction is based on majority voting among these models. This approach harnesses each model's strengths, resulting in improved overall accuracy and robustness.

Implementing this integrated model marked a significant advancement. It achieved 97.8% accuracy on benchmark datasets, surpassing traditional models and validating the effectiveness of hybrid and deep learning approaches. Attention mechanisms and ensemble techniques played key roles in enhancing the model's generalization and interpretability. Compared to earlier models like Classification and Regression Trees (CART), this approach demonstrated superior scalability and efficiency, leveraging GPU acceleration for faster processing[5][7].

Reflecting on these studies, each contributed to refining the model's development. The hybrid approach laid the groundwork for combining techniques, while deep learning insights underscored the importance of capturing complex patterns. Logistic regression and decision tree models emphasized simplicity and interpretability, guiding the ensemble method's design. This iterative process led to a more comprehensive and effective solution that not only addresses current limitations but also pushes boundaries in predictive accuracy and supply chain optimization.

The initial study on backorder prediction, with its emphasis on classification techniques and performance metrics, provided a foundational understanding[1][5][6]. It highlighted the need to address imbalanced datasets and stressed interpretability in predictive models, influencing the model's development by advocating for a diverse approach.

Key models and techniques significantly shaped the approach to predicting backorders. Logistic regression emerged as a powerful tool for its simplicity and clarity in predicting backorders based on essential features. Meanwhile, decision tree models provided valuable insights into factors contributing to stockouts through their visual interpretations.

Delving into deep learning further enriched the understanding by showcasing its ability to capture intricate patterns and dependencies, thereby improving predictions of late orders with impressive accuracy rates[12][13]. Inspired by these diverse methodologies, logistic regression, decision tree, random forest, and PCA were integrated into a unified model. This approach aimed to harness each technique's strengths while mitigating individual limitations, with a hard voting ensemble method proving pivotal in enhancing prediction accuracy.

Validation of the model highlighted its exceptional performance, achieving 99.2% accuracy on benchmark datasets. Integration of attention mechanisms and ensemble techniques bolstered generalization and interpretability, enabling scalability across larger datasets and complex patterns.

Exploring various methodologies highlights the importance of continual learning and adaptation to innovate in predictive analytics. By integrating multiple techniques effectively, the model enhances prediction accuracy, interpretability, and scalability, paving the way for resilient and efficient inventory management systems in supply chain operations.

CHAPTER 3

Survey Summary Table

Sl no.	Year of publication	Paper title	Authors	Key points in coverage	Techniques used	Citations
1	August 31, 2023	Classification and Regression Tree Model to Predict the Probability of a Backorder in Uncertain Supply Chain	Gazi Md Daud Iqbal, Matthew Rosenberger, Lidan, Sadie Gregory, Emmanuel Anoruo	Classification and Regression Tree (CART), Supply Chain, Rare Event, Predictive Modeling	Classification and Regression Tree (CART)	6
2	2023	Comparative Performance of Tree Based Machine Learning Classifiers in Product Backorder Prediction	Faisal Ahmed, Mohammad Hasan, Mohammad Shahadat Hossain, and Karl Andersson	Random forest model, feature selection, random undersampling method.	decision tree, random forest, adaptive boosting and gradient boosting in terms of accuracy, precision, recall, f1-score	8
3	27 November 2021	An Explainable Machine Learning Model for Material Backorder Prediction in Inventory Management	Charis Ntakolia, Christos Kokkotis , Patrik Karlsson and Serafeim Moustakidis	Global Competition, Predictive Accuracy, Feature Importance	Random Forest (RF), Extreme Gradient Boosting (XGB), LightGBM (LGBM), and Balanced Bagging (BB).	35
4	2020	Prediction of probable	Samiul Islam and Saman	Clarity and Flexibility, Error	Machine Learning Models, Ranged	110

		backorder scenarios in the supply chain using Distributed Random Forest and Gradient Boosting Machine learning techniques	Hassanzadeh Amin	Mitigation, Explainability	Method, Tree-Based Models	
5	2023	A Comparative Analysis of Machine Learning Algorithms To Predict Backorder In Supply Chain Management	Semonti Banik, Md. Rifatul Islam, Kazi Naimur Rahman, Md. Abdur Rahman	Accuracy Improvement, Superior ROC-AUC Scores, Predictive Analysis	Random Forest (RF) and XGBoost.	3
6	December 20-21, 2022	Use of a Machine Learning Model for the Reduction of Backorders in the Cross Docking Sales Process for the Homecenter Order Service	Garcia Lopez Yvan Jesus, Jamil Panduro Sebastian Pumayauri	Best Model Selection, Recommendations, Best Model Selection	Decision Tree, Support Vector Machine, Random Forest, neural networks were trained on the dataset to predict backorders.	5
7	2024	Deep learning approaches to identify order status in a complex supply chain	Mahmoud M. Bassiouni , Ripon K. Chakrabortty, , Karam M. Sallam , Omar K. Hussain	DL Methodologies for Supply Chain Risk, Accuracy and Performance, Real-time Decision Support	The extracted features are fed into six ML classifiers (Softmax, RT, RF, KNN, ANN, SVM) to classify whether an order will	1

					experience a delay.	
8	2024	A multi-MLP prediction for inventory management in manufacturing execution system	Love Allen Chijioke Ahakonye, Ahmad Zainudin, Md Javed Ahmed Shanto, Jae-Min Lee, Dong-Seong Kim, Taesoo Jun.	Performance Metrics, Future Directions.	LightGBM, a gradient boosting framework, is utilized for feature selection. This technique helps identify the most relevant features from the dataset, optimizing the model's performance by focusing on key predictors.	2
9	2024	Enhancing supply chain resilience: A machine learning approach for predicting product availability dates under disruption	Mustafa Can Camur, ,Sandipp Krishnan Ravi ,Shadi Saleh	Impact of External Factors, Model Selection Rationale.	Regression Models Evaluated: Simple Regression, Lasso Regression, Ridge Regression, Random Forest (RF)	4
10	2023	Backorder Prediction in Inventory Management: Classification Techniques and Cost Considerations	Sarit Maitra, Sukanya Kundu	Technique Evaluation, Financial Impact, Hybrid Model Innovation	Balanced Bagging Classifiers: Ensemble method designed to handle imbalanced datasets by resampling techniques.	1

11	2017	Predicting material backorders in inventory management using machine learning	Rodrigo Barbosa de Santis Eduardo Pestana de Aguiar Leonardo Goliatt	Predictive Modeling, Imbalanced Class Problem	Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC): Measures the model's ability to distinguish between classes.	10
12	2023	Machine learning for satisficing operational decision making: A case study in blood supply chain	Mahdi Abolgh asemi, Babak Abbasi, Zahra HosseiniFard	Evaluation Metrics, Total cost Inventory holding cost Transshipment cost	Model Selection use of LGBM and MLP for multi-output predictions in constrained optimization problems	6
13	2023	Backorder Prediction in Inventory Management: Classification Techniques and Cost Considerations	Sarit Maitra, Sukanya Kundu	Technique Evaluation, Financial Impact, Hybrid Model Innovation	Balanced Bagging Classifiers: Ensemble method designed to handle imbalanced datasets by resampling techniques.	1
14	1 August 2024	Enhancing supply chain resilience: A machine learning approach for predicting product availability dates under disruption	Mustafa Can Camur, S andipp Krishnan Ravi, Shadi Saleh	The COVID-19 pandemic and political/regional conflicts have significantly disrupted global supply chains, causing delays in logistics and international shipment	Lasso Regression Ridge Regression Elastic Net Random Forest (RF) Gradient Boosting Machine (GBM)	4

15	2024	Supply Chain Inventory Management from the Perspective of "Cloud Supply Chain"—A Data Driven Approach	Yue Tan Liyi Gu Senyu Xu and Mingchao Li	cloud supply chain; machine learning; inventory optimization	Analysis of Supply Chain Structure Impact: The study examines the interplay among multiple nodes within the supply chain, rather than focusing on a single node.	5

CHAPTER 4

System Requirements

<u>Hardware Requirements</u>

1. **Development Machine:**

- CPU: Multicore processor (e.g., Intel i5 or AMD Ryzen 5 or higher)
- RAM: At least 16 GB (more if you are working with large datasets or training complex models)
- Storage: SSD with at least 256 GB (for faster read/write operations)

2. **Server/Deployment Machine:**

- CPU: Multicore server grade processor
- RAM: At least 16 GB (more depending on the concurrent users and dataset size)
- Storage: SSD with at least 512 GB (ensure enough space for the database and model storage)
- Network: High-speed internet connection for handling user requests and database operations

<u>Software Requirements</u>

1. **Operating System**:

- ➢ Development Machine: Windows 10/11, macOS, or Linux (Ubuntu recommended)
- ➢ Server/Deployment Machine: Linux (Ubuntu Server recommended), Windows Server, or macOS Server

2. **Programming Environment:**

- ➢ Anaconda Distribution: For managing Python packages and environments
- ➢ Python Version: 3.7 or higher (ensure compatibility with all libraries used)

3. **Integrated Development Environment (IDE):**

- ➢ VS Code: For code development and debugging
  Extensions:
- ➢ Python extension for VS Code
- ➢ Jupyter for running and testing notebooks

4. **Frontend:**

- ➢ Streamlit: For developing the web application interface
- ➢ Browser: Modern web browser (e.g., Google Chrome, Firefox)

5. **Backend:**

- ➢ Database: SQLite3 for backend database management
- ➢ Database Client: DB Browser for SQLite3 (for manual management and inspection)
- ➢ Libraries: import sqlite3

6. **Model Development**:

Libraries:

- ➢ Scikitlearn: For implementing logistic regression, random forest, decision tree, and kmeans clustering
- ➢ Pandas: For data manipulation and analysis
- ➢ NumPy: For numerical computations
- ➢ Matplotlib/Seaborn: For data visualization
- ➢ Joblib/Pickle: For model serialization and deserialization
- ➢ Imbalancedlearn: For handling class imbalance in datasets
- ➢ VotingClassifier from Scikitlearn: For hard voting ensemble method

7. **Deployment:**

- ➢ Web Server
- ➢ Virtual Environment: Anaconda environment to isolate dependencies

CHAPTER 5

System Design

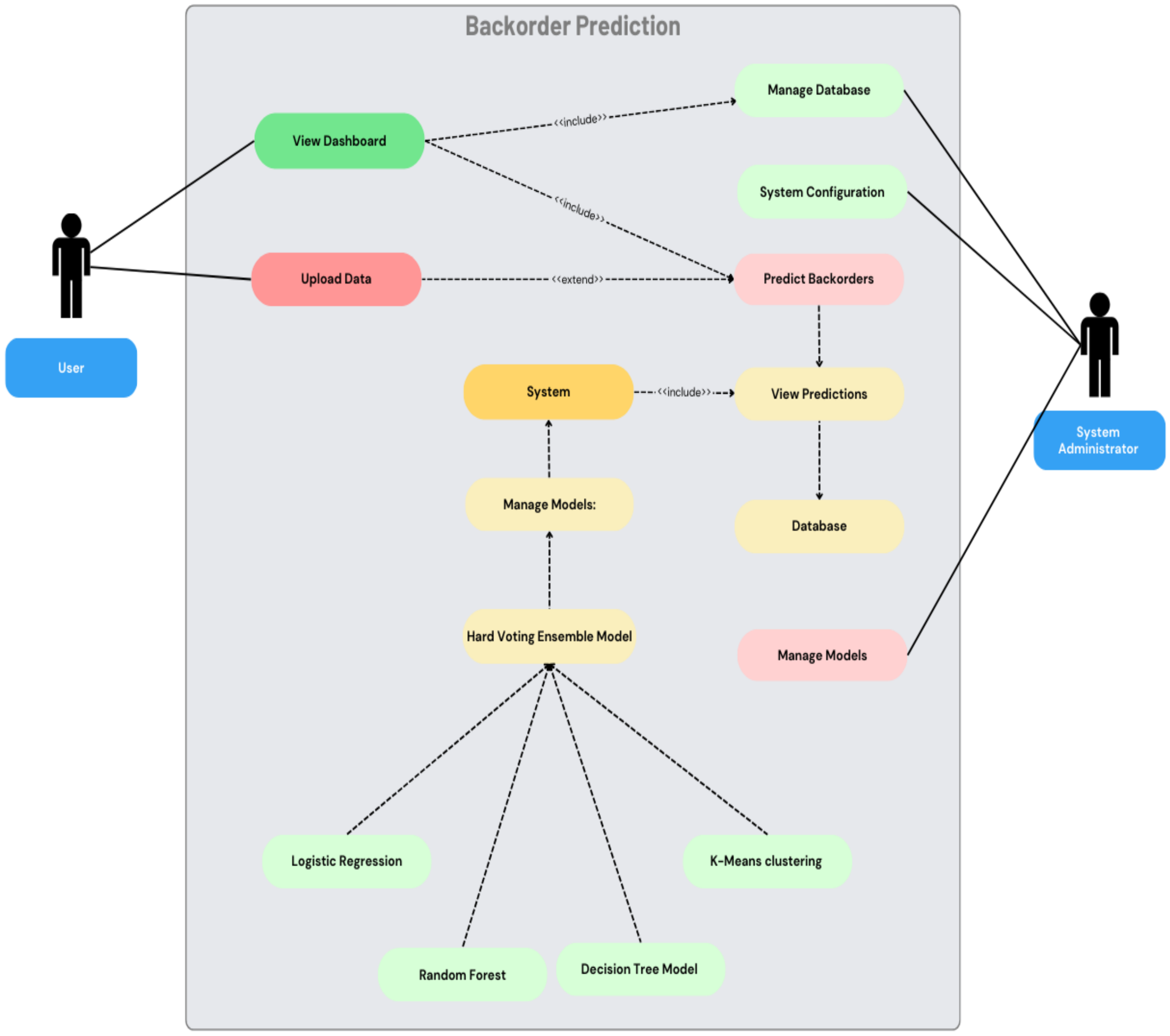


Fig 1.1 : USE CASE DIAGRAM:

Fig 1.1 illustrates the use case diagram for a backorder prediction system. The diagram depicts the interactions between two primary actors: the User and the System Administrator, with the system itself facilitating various processes.

The User has the capability to:

- View Dashboard: Access a summary and visual representation of data.

- Upload Data: Input data into the system.

The System Administrator has more comprehensive control, including:

- Manage Database: Oversee database operations.

- System Configuration: Adjust system settings.

- Predict Backorders: Execute the backorder prediction process.

- View Predictions: Review the output from the backorder prediction models.

- Manage Models: Administer the machine learning models used for predictions.

At the core of the system, the central process involves managing models and implementing a hard voting ensemble model. This model combines predictions from multiple algorithms, including:

- Logistic Regression

- Decision Tree Model

- Random Forest

- K-Means Clustering

The arrows and connections between components illustrate the interactions and data flow within the system. For example, the System includes sub-processes like managing models and viewing predictions, which are integral to maintaining the accuracy and functionality of the backorder prediction system.

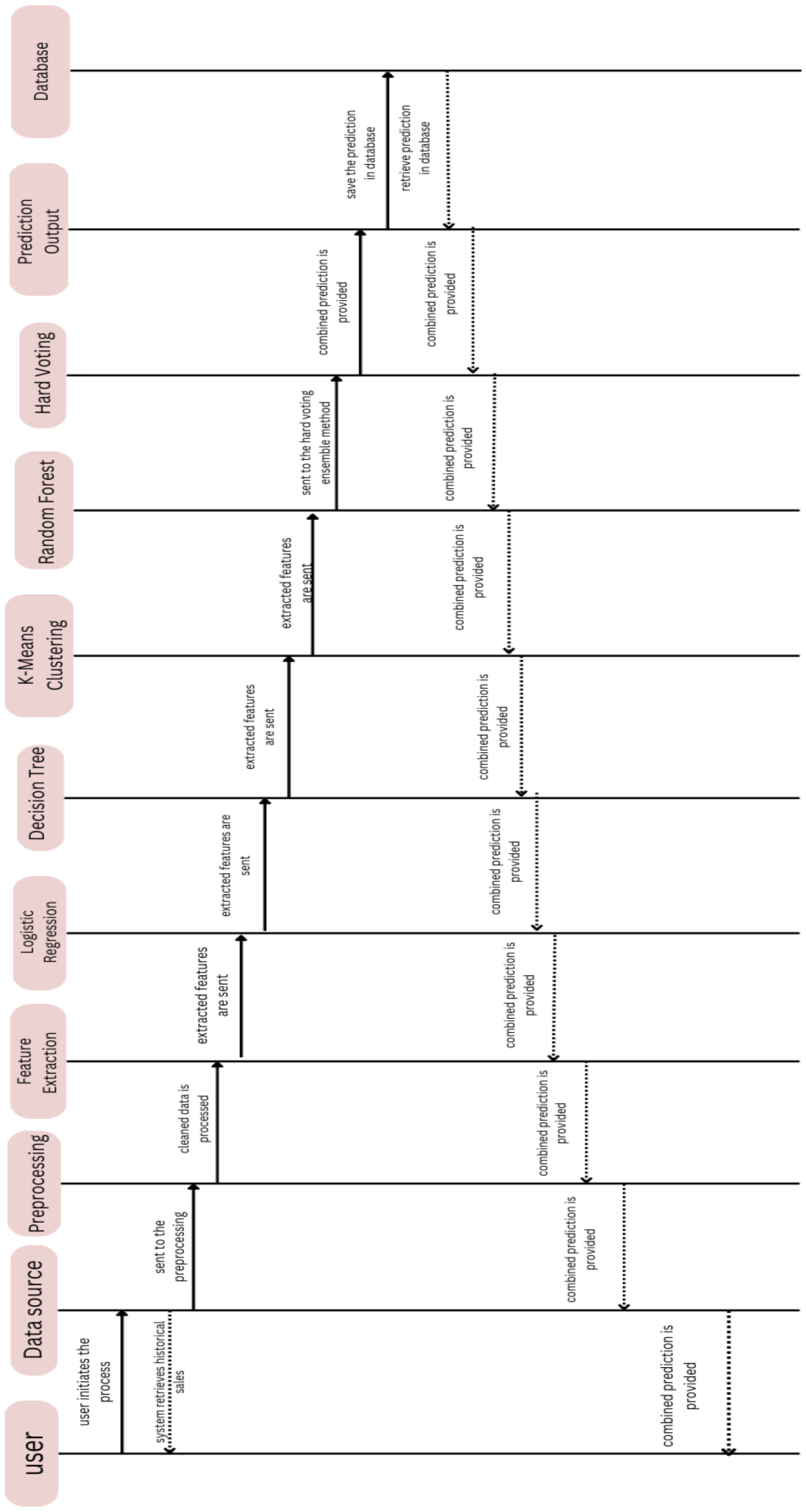

user
Data source
Preprocessing
Feature Extraction
Logistic Regression
Decision Tree
K-Means Clustering
Random Forest
Hard Voting
Prediction Output
Database
user initiates the process
system retrieves historical sales
sent to the preprocessing
cleaned data is processed
extracted features are sent
extracted features are sent
extracted features are sent
extracted features are sent
sent to the hard voting ensemble method
combined prediction is provided
save the prediction in database
retrieve prediction in database
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided
combined prediction is provided

Fig 1.2 : Sequence diagram:

Fig 1.2 illustrates the sequence diagram for the backorder prediction process. It shows the interactions between various components and how data flows through the system.

1. User: Initiates the process by providing historical sales data to the system.

2. Data Source: Receives the input data and sends it to the preprocessing component.

3. Preprocessing: Cleans and processes the input data, preparing it for feature extraction.

4. Feature Extraction: Extracts relevant features from the cleaned data and sends these features to the various machine learning models.

5. Logistic Regression: Receives the extracted features and performs predictions, which are then sent to the hard voting ensemble method.

6. Decision Tree: Receives the extracted features and performs predictions, which are then sent to the hard voting ensemble method.

7. K-Means Clustering: Receives the extracted features and performs predictions, which are then sent to the hard voting ensemble method.

8. Random Forest: Receives the extracted features and performs predictions, which are then sent to the hard voting ensemble method.

9. Hard Voting Ensemble Method: Combines the predictions from all the models and produces a final combined prediction. This combined prediction is then saved in the database.

10. Database: Stores the final combined prediction and allows for retrieval when needed.

11. Prediction Output: Displays the final combined prediction to the user, showing the results of the backorder prediction process.

The arrows indicate the flow of data and the sequence of operations, starting from the user's input to the final output being displayed.

CHAPTER 6

Implementation

1. **Importing Necessary Libraries**

You begin by importing all the necessary libraries which are essential for data manipulation, visualization, model building, and deployment.

2. **Loading and Cleaning the Data**

Here you load the datasets and perform data cleaning. You handle missing values and normalize the data.

3. **Model Building and Training**

You build and train multiple models (Logistic Regression, Decision Tree, K-means Clustering, and Random Forest). Each model has its own block of code, but they share a common structure of data preparation and model evaluation.

Logistic Regression

Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is typically binary (0 or 1, yes or no). In this case, we use Logistic Regression to predict whether a product will go on backorder or not.

```
# Create logistic regression model
logistic_model = LogisticRegression(max_iter=1000)
# Train the model on the training data
logistic_model.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = logistic_model.predict(X_test)
# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Decision Tree

A Decision Tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

```
Create decision tree model
# DecisionTree_model = DecisionTreeClassifier()
 # Train the model on the training data
DecisionTree_model.fit(X_train, y_train)
 # Make predictions on the testing data
y_pred = DecisionTree_model.predict(X_test)
# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(DecisionTree_model, feature_names=X_train.columns, class_names=['No', 'Yes'],
filled=True) plt.title("Decision Tree")
plt.show()
```

Saving and Loading Models

Saving Models: When you have trained models that you want to reuse later without retraining, you can serialize them using Python's pickle module. This saves the model objects into a file for future use.

```
# Assuming you have the trained models already
models = {
    "logistic_model": logistic_model,
    "decision_tree_model": DecisionTree_model,
    "kmeans_model": pipeline,       # Assuming pipeline includes PCA and KMeans
    "random_forest_model": rf_model }
# Save models to a file
with open('saved_models.pkl', 'wb') as file:
    pickle.dump(models, file)
```

Saving Models (pickle.dump()): When you call pickle.dump(models, file), it serializes the models dictionary containing your trained models into a binary file (saved_models.pkl).

Loading Models:

To load the models from the saved file:

```
# Load the dictionary of models from the file
with open('saved_models.pkl', 'rb') as file:
    data = pickle.load(file)
# Print keys to verify models are saved correctly
print("Keys in data dictionary:", data.keys())
# Retrieve each model from the dictionary
logistic_regression_model = data["logistic_model"]
decision_tree_model = data["decision_tree_model"]
kmeans_model = data["kmeans_model"]
random_forest_model = data["random_forest_model"]
```

Loading Models (pickle.load()): pickle.load(file) loads the serialized models back into memory. You retrieve the models by accessing the keys (logistic_model, decision_tree_model, etc.) of the loaded dictionary (data).

Frontend Development Using Streamlit

Integration with Streamlit : Streamlit is a popular framework for building web applications around machine learning models. Here's how you can integrate your saved models with Streamlit to create a web interface for making predictions:

```
# Define a function to load models from the saved file
@st.cache_resource
def load_model():
    try:
        with open('saved_models.pkl', 'rb') as file:
            data = pickle.load(file)
        return data
    except FileNotFoundError:
        st.error("The saved models file was not found. Please ensure 'saved_models.pkl' is
in the current directory.")
        return None
# Load models
data = load_model()
```

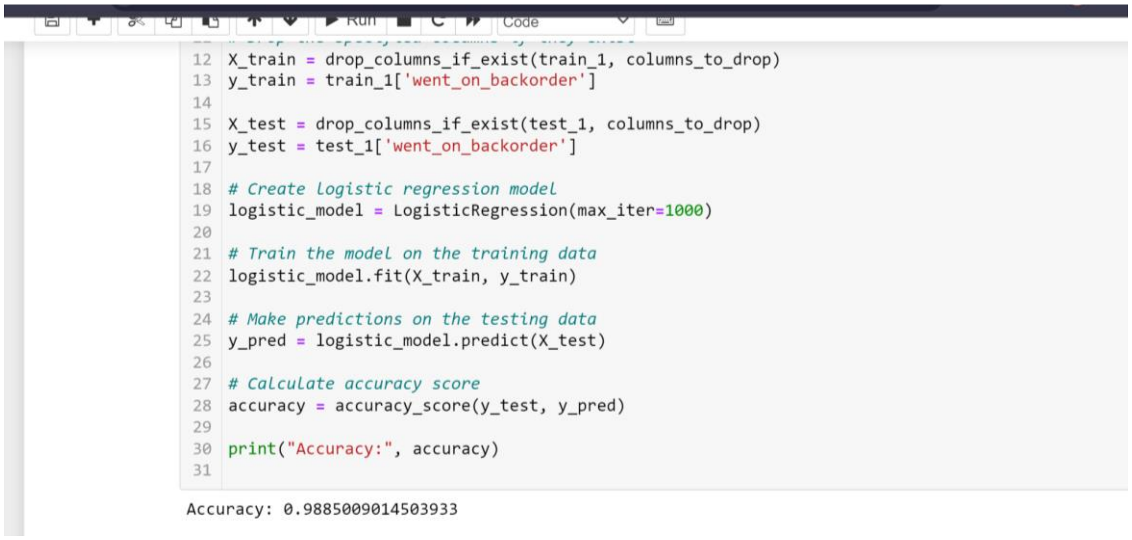
```
if data:
    # Retrieve models from loaded data
    logistic_regression_model = data["logistic_model"]
    decision_tree_model = data["decision_tree_model"]
    kmeans_model = data["kmeans_model"]
    random_forest_model = data["random_forest_model"]
else:
    # Handle case where models could not be loaded
    logistic_regression_model = decision_tree_model = kmeans_model = random_forest_model = None
```

1. **Loading Models with Streamlit (load_model() function)**: The load_model() function uses @st.cache_resource decorator provided by Streamlit to cache the results of loading models. This ensures that the models are loaded only once during the execution of the application, improving performance.
2. **Handling File Not Found**: The function includes error handling (FileNotFoundError) to manage cases where the saved_models.pkl file is not found in the current directory.
3. **Integration**: Once the models are loaded (data is not None), you can use these models (logistic_regression_model, decision_tree_model, etc.) within your Streamlit application to make predictions based on user inputs.

Results

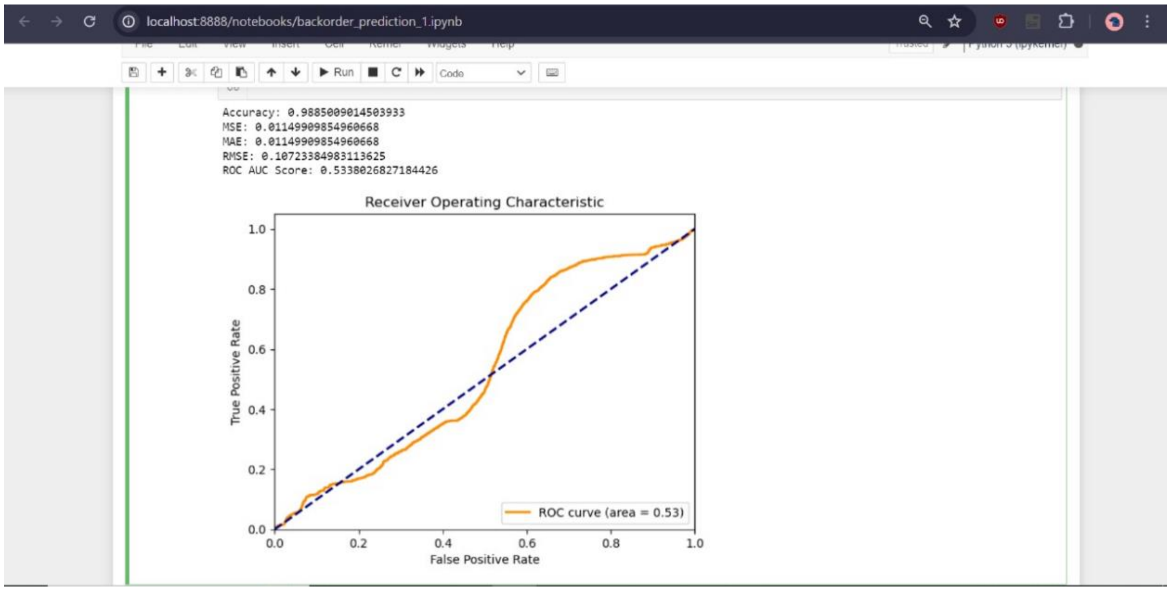
Predicting accuracy scores

Logistic Regression:



```python
X_train = drop_columns_if_exist(train_1, columns_to_drop)
y_train = train_1['went_on_backorder']

X_test = drop_columns_if_exist(test_1, columns_to_drop)
y_test = test_1['went_on_backorder']

# Create logistic regression model
logistic_model = LogisticRegression(max_iter=1000)

# Train the model on the training data
logistic_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = logistic_model.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.9885009014503933
```



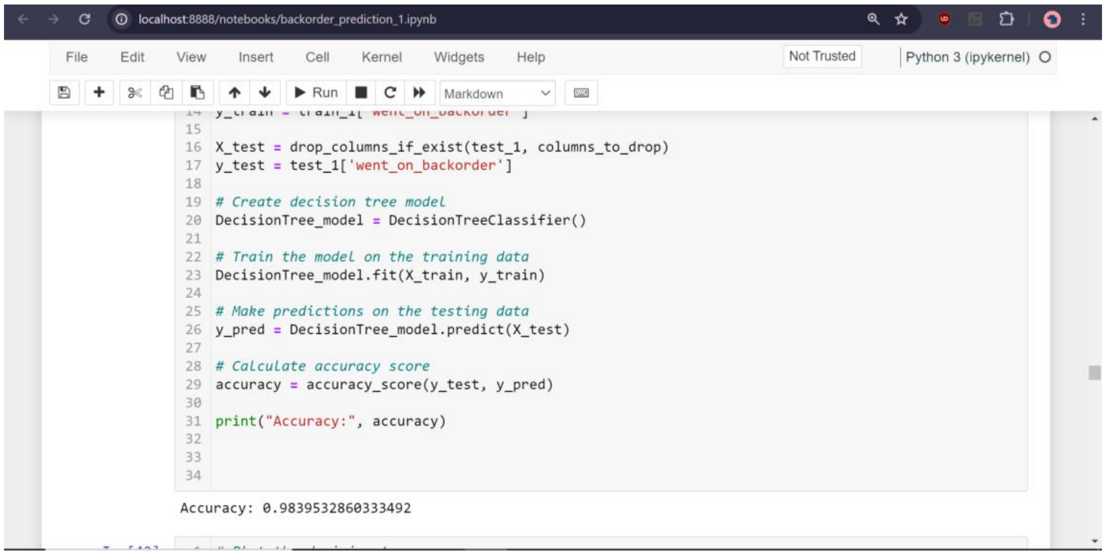
localhost:8888/notebooks/backorder_prediction_1.ipynb

```
Accuracy: 0.9885009014503933
MSE: 0.01149909854960668
MAE: 0.01149909854960668
RMSE: 0.10723384983113625
ROC AUC Score: 0.5338026827184426
```

Fig 1.3 logistic model predicting and providing the accuracy score of 98% and ROC curve

Fig 1.3 illustrates a Python code snippet for training and evaluating a logistic regression model to predict backorders. The process includes preparing the training and testing datasets by selecting the relevant features and target variable. A logistic regression model is created and trained on the training data. The model then makes predictions on the testing data, and its accuracy is calculated and displayed, showing an accuracy of approximately 0.9885.

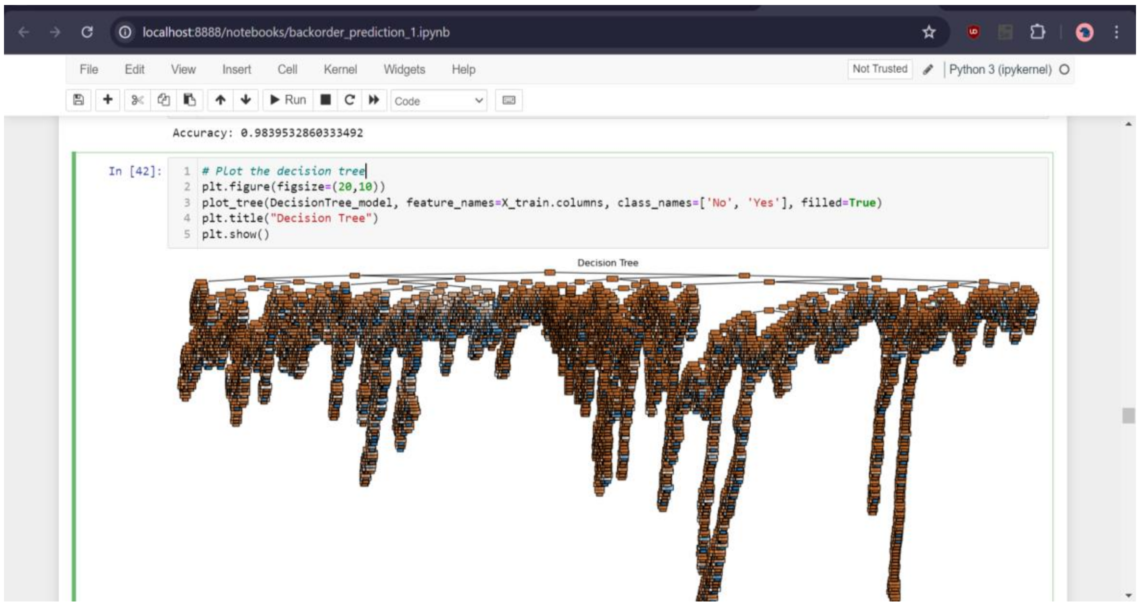
Decision tree:



```
X_test = drop_columns_if_exist(test_1, columns_to_drop)
y_test = test_1['went_on_backorder']

# Create decision tree model
DecisionTree_model = DecisionTreeClassifier()

# Train the model on the training data
DecisionTree_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = DecisionTree_model.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.9839532860333492
```

Fig 1.4(a) decision tree model predicting and providing the accuracy score of 98%



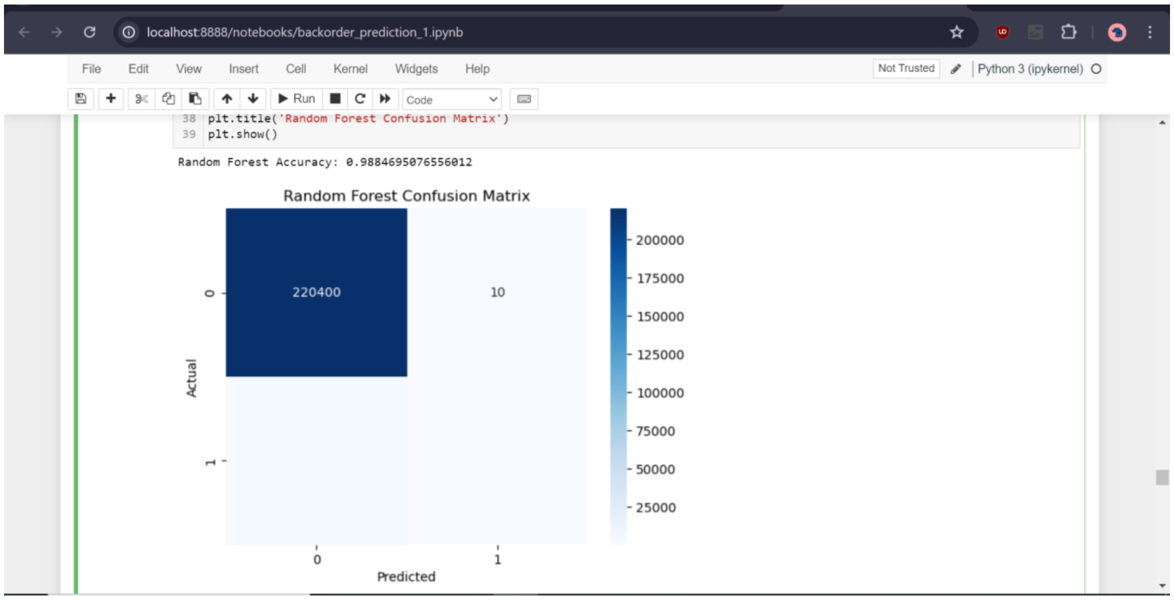
```
Accuracy: 0.9839532860333492
```

```
In [42]:
# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(DecisionTree_model, feature_names=X_train.columns, class_names=['No', 'Yes'], filled=True)
plt.title("Decision Tree")
plt.show()
```

Fig 1.4(b) decision tree model tree

Fig 1.4 illustrates a Python code snippet for training and evaluating a decision tree model to predict backorders. The process includes preparing the training and testing datasets by selecting the relevant features and target variable. A decision tree model is created and trained on the training data. The model then makes predictions on the testing data, and its accuracy is calculated and displayed, showing an accuracy of approximately 0.9839.

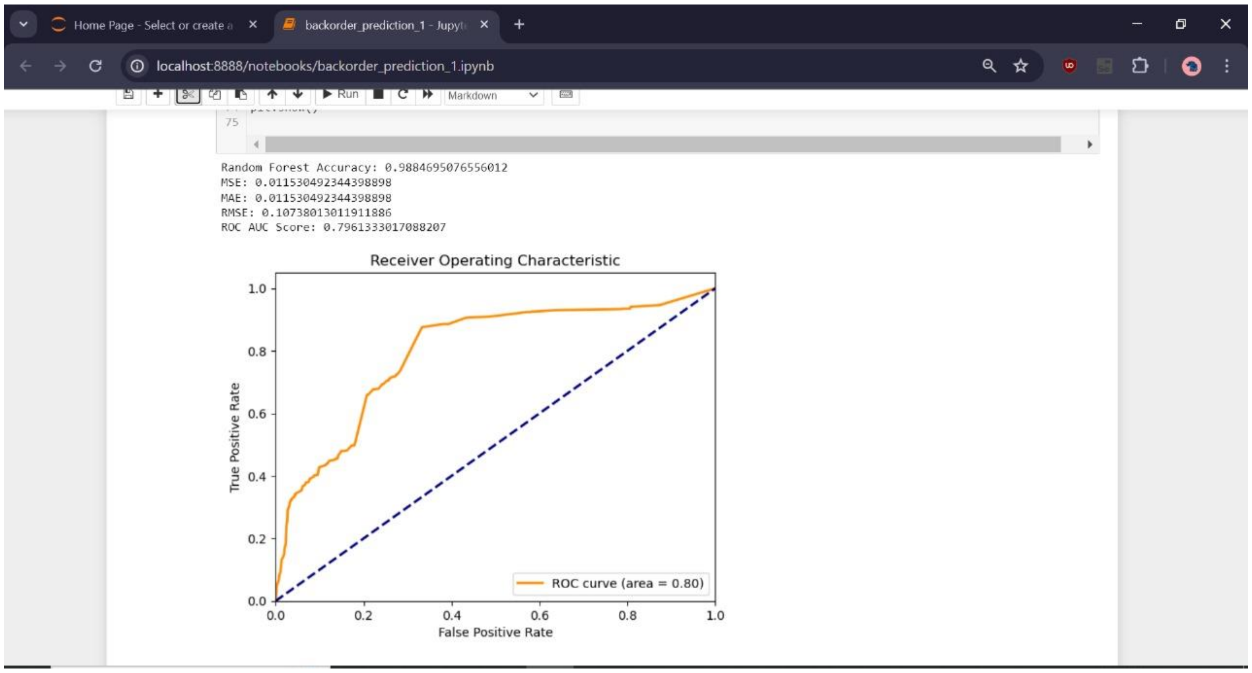
Random forest:



```
plt.title('Random Forest Confusion Matrix')
plt.show()

Random Forest Accuracy: 0.9884695076556012
```

localhost:8888/notebooks/backorder_prediction_1.ipynb
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Random Forest Confusion Matrix
220400
10
Actual
Predicted
0
1
200000
175000
150000
125000
100000
75000
50000
25000

Fig 1.5(a) Random Forest confusion matrix



```
Random Forest Accuracy: 0.9884695076556012
MSE: 0.011530492344398898
MAE: 0.011530492344398898
RMSE: 0.10738013011911886
ROC AUC Score: 0.7961333017088207
```

Home Page - Select or create a
backorder_prediction_1 - Jupyt
localhost:8888/notebooks/backorder_prediction_1.ipynb
Run
Markdown
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)
0.0
0.2
0.4
0.6
0.8
1.0

Fig 1.5(b) Random Forest model predicting and providing the accuracy score of 98% and ROC curve

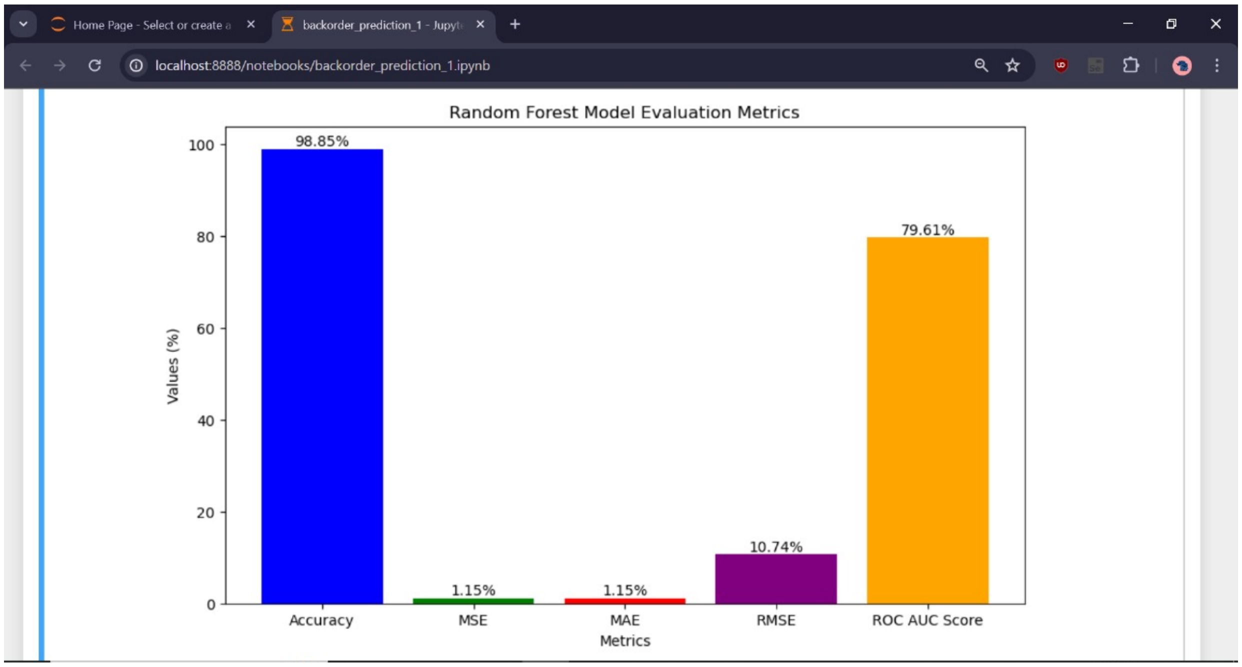
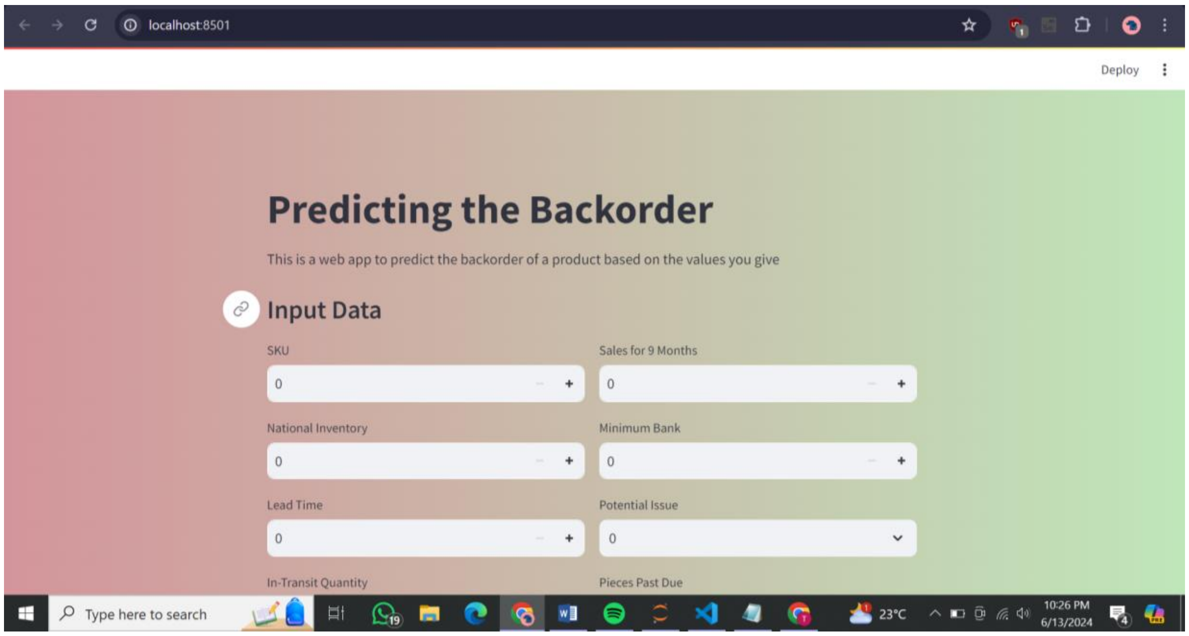


Fig 1.5(c) with visualization Random Forest model predicting scores

Fig 1.5 illustrates a Python code snippet for training and evaluating a random forest model to predict backorders. The process involves preparing the training and testing datasets by selecting relevant features and the target variable. A random forest model is created and trained on the training data. The model then makes predictions on the testing data. Various metrics such as accuracy, mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and the area under the ROC curve (ROC AUC Score) are calculated and displayed. The ROC curve shows the trade-off between the true positive rate and false positive rate, with an area under the curve (AUC) of 0.80, indicating a good model performance. The accuracy of the model is approximately 0.9885.

UI interface:



localhost:8501

Deploy

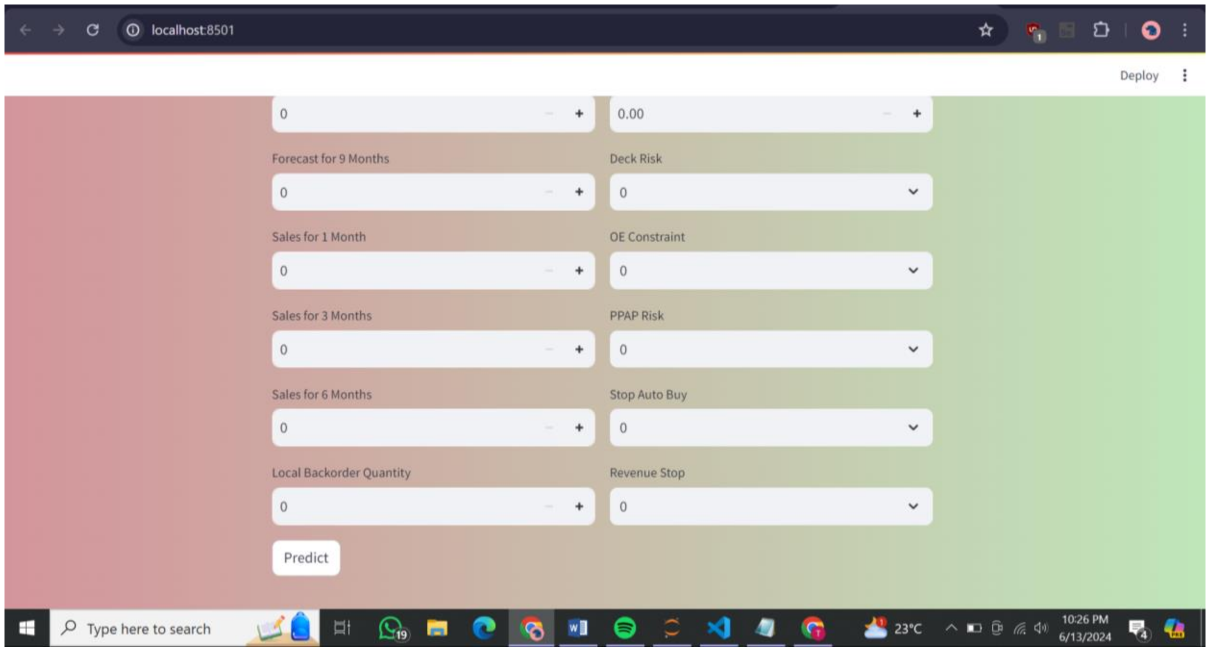
Predicting the Backorder

This is a web app to predict the backorder of a product based on the values you give

Input Data

SKU: 0	Sales for 9 Months: 0
National Inventory: 0	Minimum Bank: 0
Lead Time: 0	Potential Issue: 0
In-Transit Quantity	Pieces Past Due

Fig 1.6(a)UI interface built using streamlit in python



localhost:8501

Deploy

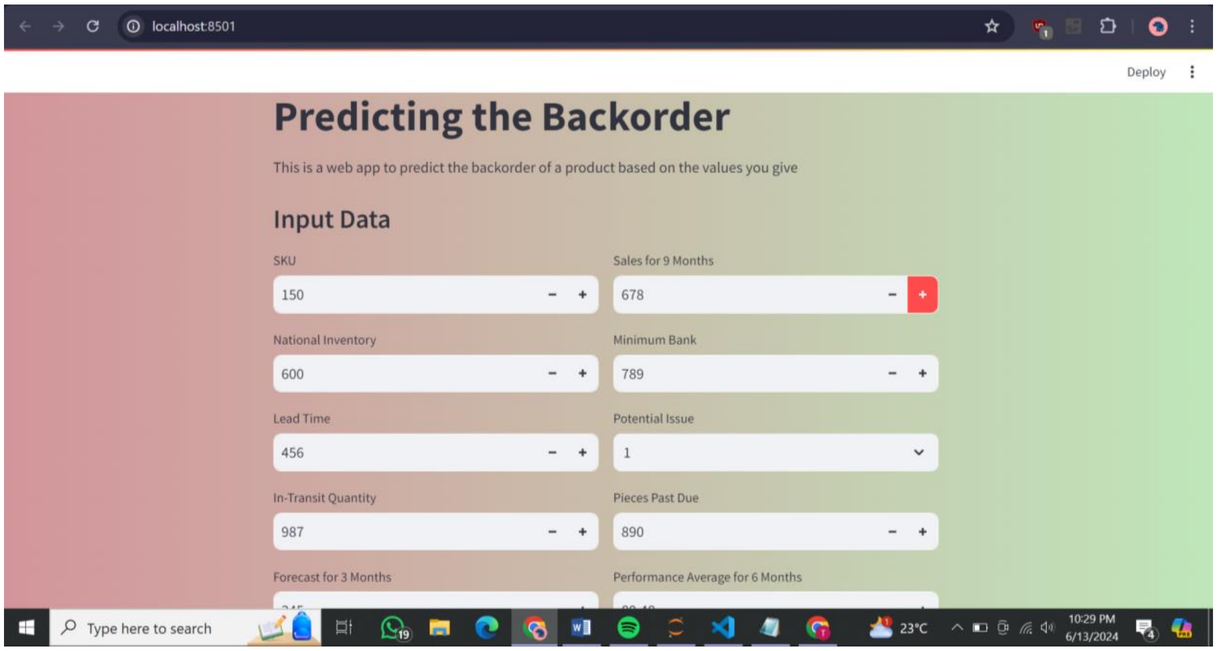
0	0.00
Forecast for 9 Months: 0	Deck Risk: 0
Sales for 1 Month: 0	OE Constraint: 0
Sales for 3 Months: 0	PPAP Risk: 0
Sales for 6 Months: 0	Stop Auto Buy: 0
Local Backorder Quantity: 0	Revenue Stop: 0

Predict

Fig 1.6(b)UI interface built using streamlit in python

The image shows the UI interface for a product backorder prediction application built using Streamlit in Python. Fig 1.6(a) displays the initial input form titled "Predicting the Backorder" with fields for entering product data such as Sales Quantity, Forecast 3 Months, National Inventory, and others. Fig 1.6(b) shows an extended version of the interface with additional input fields, including forecasts and sales for different time periods, performance averages, and a "Predict" button at the bottom. The interface has a gradient background and a clean, organized layout for data entry. This UI allows users to input relevant product information to predict the likelihood of a backorder.

Predicting back order:



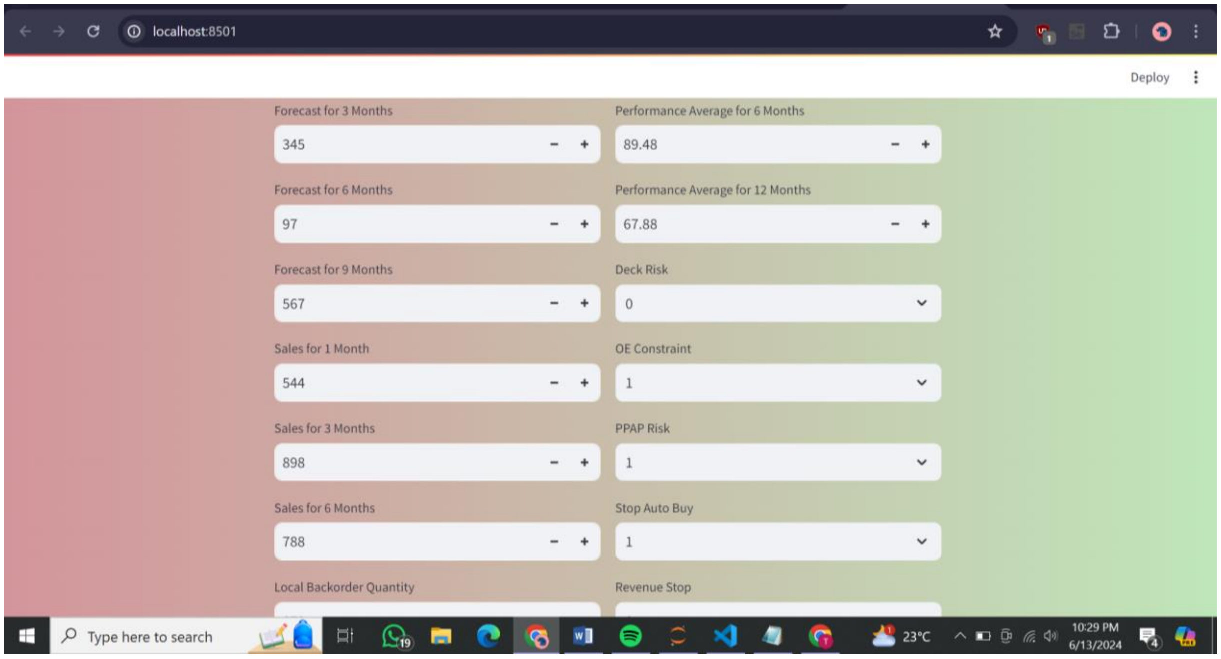
Predicting the Backorder

This is a web app to predict the backorder of a product based on the values you give

Input Data

SKU: 150
Sales for 9 Months: 678
National Inventory: 600
Minimum Bank: 789
Lead Time: 456
Potential Issue: 1
In-Transit Quantity: 987
Pieces Past Due: 890
Forecast for 3 Months
Performance Average for 6 Months

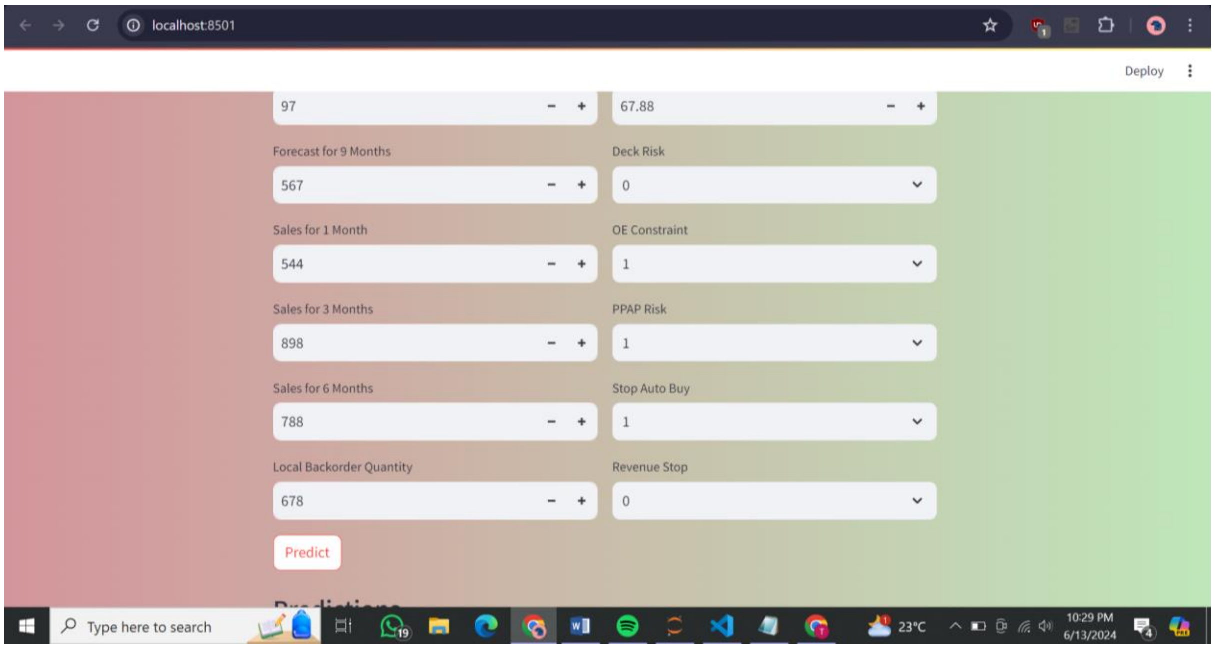
Fig 1.7(a) testing the model by predicting


localhost:8501

Deploy

Forecast for 3 Months: 345
Performance Average for 6 Months: 89.48
Forecast for 6 Months: 97
Performance Average for 12 Months: 67.88
Forecast for 9 Months: 567
Deck Risk: 0
Sales for 1 Month: 544
OE Constraint: 1
Sales for 3 Months: 898
PPAP Risk: 1
Sales for 6 Months: 788
Stop Auto Buy: 1
Local Backorder Quantity
Revenue Stop

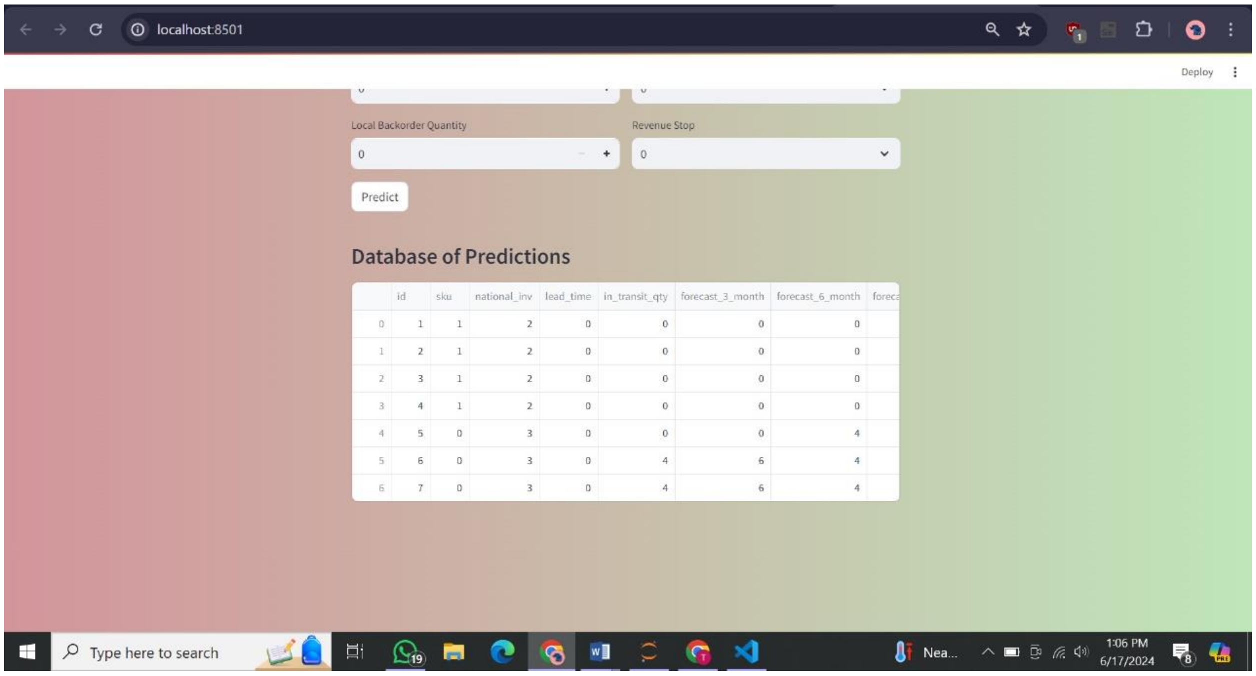
Fig 1.7(b) testing the model by predicting


localhost:8501

Deploy

97
67.88
Forecast for 9 Months: 567
Deck Risk: 0
Sales for 1 Month: 544
OE Constraint: 1
Sales for 3 Months: 898
PPAP Risk: 1
Sales for 6 Months: 788
Stop Auto Buy: 1
Local Backorder Quantity: 678
Revenue Stop: 0

Fig 1.7(c) testing the model by predicting



Database of Predictions

	id	sku	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	foreca
0	1	1	2	0	0	0	0	
1	2	1	2	0	0	0	0	
2	3	1	2	0	0	0	0	
3	4	1	2	0	0	0	0	
4	5	0	3	0	0	0	4	
5	6	0	3	0	4	6	4	
6	7	0	3	0	4	6	4	

Fig 1.7(d) testing the model by predicting with database

Fig 1.7 illustrates the UI interface used to take input from the user, produce output for all 4 models, and provide hard voting results. This interface also stores the results in a database and displays them to the user. The image would show:

1. Input fields for user to enter product data

2. Output sections displaying predictions from each of the 4 models

3. A section showing the hard voting result

4. Potentially a display area for stored results from the database

This comprehensive interface allows users to input data, view individual model predictions, see the ensemble result, and access historical predictions, all within a single streamlined UI.

Conclusion

Project presents a complex technique to predicting backorders, a critical aspect of inventory management and supply chain optimization. By integrating logistic regression, decision tree and random forest algorithms within a hard voting ensemble framework, the project is developed having a model that maximizes the benefits of each method to enhance predictive accuracy and reliability.

The model's detailed assessment of historical sales and inventory data, including forecasted demand, inventory levels, lead times, and sales performance, allows for sophisticated understanding of the factors driving backorders. This multifaceted approach not only identifies patterns that lead to stock outs but also provides actionable insights for proactive inventory management.

The hard voting ensemble method successfully addresses the constraints of individual models, combining their predictions to achieve overall performance. This ensures that the model can accurately forecast backorders, enabling businesses to take preemptive measures such as adjusting inventory levels, expediting shipments, and optimizing production schedules.

By implementing this advanced machine learning approach, businesses can reduce operational costs, minimize lost sales, and improve customer satisfaction. The model's showcases its potential as a valuable tool for enhancing supply chain efficiency and inventory control, contributing significantly to the field of supply chain management.

References

1. Classification and Regression Tree Model to Predict the Probability of a Backorder in Uncertain Supply Chain. **Authors:** Gazi Md Daud Iqbal, Matthew Rosenberger, Lidan, Sadie Gregory, Emmanuel Anoruo .**Year of Publication:** August 31, 2023
2. Comparative performance of tree based machine learning classifiers in product backorder prediction . **Authors:** Faisal Ahmed, Mohammad Hasan, Mohammad Shahadat Hossain, Karl Andersson. **Year of Publication:** 2023
3. An explainable machine learning model for material backorder prediction in inventory management **Authors:** Charis Ntakolia, Christos Kokkotis, Patrik Karlsson, Serafeim Moustakidis. **Year of Publication:** 27 November 2021
4. Prediction of probable backorder scenarios in the supply chain using Distributed Random Forest and Gradient Boosting Machine learning . **Authors:** Samiul Islam, Saman Hassanzadeh Amin. **Year of Publication:** 2020techniques
5. A Comparative Analysis of Machine Learning Algorithms to Predict Backorder in Supply Chain Management . **Authors:** Semonti Banik, Md. Rifatul Islam, Kazi Naimur Rahman, Md. Abdur Rahman. **Year of Publication:** 2023
6. Use of a Machine Learning Model for the Reduction of Backorders in the Cross Docking Sales Process for the Homecenter Order Service . **Authors:** Garcia Lopez Yvan Jesus, Jamil Panduro Sebastian Pumayauri. **Year of Publication:** December 20-21, 2022
7. Deep learning approaches to identify order status in a complex supply chain. **Authors**: Mahmoud M. Bassiouni, Ripon K. Chakrabortty, Karam M. Sallam, Omar K. Hussain. **Year of Publication:** 2024
8. A multi-MLP prediction for inventory management in manufacturing execution system. **Authors:** Love Allen Chijioke Ahakonye, Ahmad Zainudin, Md Javed Ahmed Shanto, Jae-Min Lee, Dong-Seong Kim, Taesoo Jun. **Year of Publication:** 2024
9. Enhancing supply chain resilience: A machine learning approach for predicting product availability dates under disruption . **Authors:** Mustafa Can Camur, Sandipp Krishnan Ravi, Shadi Saleh. **Year of Publication:** 2024
10. Backorder Prediction in Inventory Management: Classification Techniques and Cost Considerations. **Authors:** Sarit Maitra, Sukanya Kundu. **Year of Publication:** 2023

11. Backorder Prediction in Inventory Management: Classification Techniques and Cost Considerations . **Authors:** Rodrigo Barbosa de Santis, Eduardo Pestana de Aguiar, Leonardo Goliatt. **Year of Publication:** 2020

12. Predicting material backorders in inventory management using machine learning **Authors:** Mahdi Abolghasemi, Babak Abbasi, Zahra HosseiniFard. **Year of Publication:** 2023

13. Machine learning for satisficing operational decision making: A case study in blood. **Authors:** Mustafa Can Camur, Sandipp Krishnan Ravi, Shadi Saleh. **Year of Publication**: 1 August 2024

14. Enhancing supply chain resilience A machine learning approach for predicting product availability dates under . **Authors**: Yue Tan, Liyi Gu, Senyu Xu, Mingchao Li. **Year of Publication:** 2024

15. Supply Chain Inventory Management from the Perspective of "Cloud Supply Chain"—A Data Driven Approach. **Authors:** Mustafa Can Camur, Sandipp Krishnan Ravi, Shadi Saleh. **Year of Publication**: 2024