Low Level Design

**Food Recommendation System**

|  |  |
| --- | --- |
| Attributes | Values |
| Written By | Thomas Antony |
| Document Version 0.3 | 0.3 |
| Last Revised Date | 23 – May -2021 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 |  | Thomas | First Draft |
|  |  |  |  |
|  |  |  |  |

**Document Control**

# 1. Components and Modules

1.1 Model Training Module

* **Purpose:**
  + Train machine learning models for backorder prediction using historical data.
* **Sub-Components:**

### Data Preprocessing

* + Feature Selection

Only Features most helpful have been used other redundant columns has been removed which is also our feature scaling step

* Data Preprocessing

We have scaled the data for a better performance by the model and we have used Synthetic oversampling to rectify the class imbalance present in the data

* + Model Training

We have used RandomForestClassifier as the model we have selected we came to selecting this model by conducting experiments using jupyter notebooks

from sklearn.ensemble import RandomForestClassifier

* + Model Evaluation

We have evaluated the model using accuracy, precision, recall and F1 score

## 1.2 Data and Model Versioning Module

## DVC (Data Version Control):

* + Track data versions to identify data drifts.
  + **Process:**
    - Hash key tracking of data versions.

## MLflow:

* + Log models, evaluation metrics, and parameters for versioning.
  + **Process:**
    - Log model details and metrics.
    - Compare different model versions.

## 1.3 Event Logging Module

* **Purpose:**
  + Log every event within the system for debugging and monitoring.
* **Sub-Components:**
  + Custom Exception Handling
  + Logging Methods (Database/File)
  + Non-blocking Logging

# 2. Process Flow

## Model Training and Evaluation:

* + Utilize supervised machine learning models for binary classification.
  + Follow a step-by-step process involving data preprocessing, feature engineering, model training, and model evaluation.

## Model Prediction:

* + Utilize html webpage to create an API for predicting new data by the user

# 3. Error Handling

## Purpose:

* + Display explanations for encountered errors.

### Sub-Components:

* + - Custom Exception Function
    - Error Logging

# 4. Performance

## Evaluation Metrics:

* + Assess accuracy, precision, and recall.
  + Addressed class imbalance for better precision and recall.

# 5. Version Control

## DVC (Data Version Control):

* + Track data versions to identify data drifts.
  + **Process:**
    - Hash key tracking of data versions.

## MLflow(Model Version Control):

* + Log models, evaluation metrics, and parameters for versioning.
  + **Process:**
    - Log model details and metrics.
    - Compare different model versions.

## GitHub:

* + Source code management

# 8. Conclusion

## Application Performance

The Back Order Prediction application has undergone thorough evaluation, and the performance metrics provide insights into its effectiveness in predicting backorders. Here is a detailed analysis of the key metrics:

1. **Accuracy:**
   * The overall accuracy of the Back Order Prediction model is commendable, reaching approximately 91.25%. This metric reflects the ability of the model to correctly classify both positive and negative instances.
2. **F1 Score:**
   * The F1 score, a harmonic mean of precision and recall, is calculated at approximately 0.10. While the F1 score is relatively lower, it accounts for the trade-off between precision and recall, especially in scenarios with class imbalance.
3. **Precision:**
   * Precision, representing the ratio of true positive predictions to the total predicted positives, stands at around 5.41%. This indicates that a low percentage of predicted backorders are actual backorders. The model may have a tendency to generate false positives.
4. **Recall:**
   * Recall, also known as sensitivity or true positive rate, is approximately 72.47%. This metric signifies that the model is effective in capturing a significant portion of actual backorders. However, it is important to note that there might be instances where true backorders are not identified (false negatives).

## Interpretation:

* **Class Imbalance Impact:**
  + The class imbalance in the dataset is evident from the metrics. While accuracy appears high, precision is disproportionately affected, leading to a lower F1 score. Class imbalance often results in models biased towards the majority class.
* **Trade-off Between Precision and Recall:**
  + The trade-off between precision and recall is a common challenge in binary classification. A higher recall indicates a model that captures more positive instances, but this comes at the cost of precision.
* **Use Case Considerations:**
  + The application's utility depends on the specific requirements of the business use case. If the emphasis is on capturing as many backorders as possible, the model's recall is a crucial factor. However, if minimizing false positives is a priority, precision becomes more critical.
* **Areas for Improvement:**
  + To enhance the model's performance, addressing the class imbalance issue through techniques like resampling or adjusting class weights may be explored. Additionally, fine-tuning the model parameters or trying different algorithms could contribute to better precision-recall balance.
* **Application Recommendations:**
  + The Back Order Prediction application, despite its limitations, can still serve as a valuable tool for inventory management, product merchandising, and customer satisfaction. It provides insights that can be strategically used in decision-making processes.

In conclusion, while the model demonstrates strong accuracy and recall, further refinement and optimization are advisable to achieve a more balanced and effective Back Order Prediction system. The choice between precision and recall should be aligned with the specific goals and priorities of the business or operational use case.