Micron NUS-ISE Business Analytics Case Competition 2024

Team KJYS

Qn1 Prediction of Usage of Materials

Part(a) Forecast model

Preparation

Import necessary dependencies

```
import pandas as pd
import numpy as np
import random
```

Model Overview

- **Time Series Simulation**:
  To minimize wastage and ensure compatibility for future time series optimization, model training and data fitting, we choose not to split whole duration of 10 weeks into separate weeks(say, counting the usage of material per week), neither do we split the process into single batch renewals, instead we simulate time series per minute.
- **ProductList containing objects 'product'**:
  Ensuring immutability of its intrinsic attributes such as batchLife and loadSize. Also ensuring compatibility for future change of product numbers and properties.
- **Object 'tank'**:
  Ensuring flexibility through avoiding magic numbers such as tank size. Compatibility for future change of tank numbers and structures

Object Classes

```
class Tank:
    def __init__(self, size):
        self.size = size
        self.remainBatchLife = 1
        self.processingStatus = 0

    def __str__(self):
        return f"tank sized {self.size}L"

    def replenish(self, reclaimEfficiency):
        self.remainBatchLife = 1

        return self.size * (1- reclaimEfficiency)

    def processOneBatch(self, product):
        return 0

    def getRemainBatchLife(self):
        return self.remainBatchLife

    def isProcessing(self):
        return self.processingStatus

    def shiftProcessing(self):
        self.processingStatus = not self.processingStatus

    def cutBatchLife(self, product):
        self.remainBatchLife -= product.getCostPerBatch()
```

```
class Product:
    def __init__(self, id, batchLife, loadSize):
        self.id = id
        self.remain = 0
        self.costPerBatch = 1 / batchLife
        self.loadSize = loadSize

    def getId(self):
        return self.id

    def getLoadSize(self):
        return self.loadSize

    def __str__(self):
        return f"Product {self.id}"

    def getCostPerBatch(self):
        return self.costPerBatch
```

Constants and ReadData

At this stage weekly demand of wafers is given and chemicalLife is fixed constant.

```
# Constants
WEEKMINS = 7 * 24 * 60
chemicalLife = 500
# Read Data
demand = pd.read_csv("Datasets/Weekly_Projections.csv")
```

Functions reserved for future optimization

At this stage, we assume that within each week only after meeting the demand of the previous products will we start producing batches of the next product in the productList. Actually this reduces wastage in some way since one replenishment can be therefore equally divided into batches(according to batchLife respectively).

However, there exists possibilities that given the automatic replenishment every 500 minutes, we may further optimize the wastage by changing the order of production of batches(eg. change from default productA 1 batch, productA 1 batch, ..., A demand reached, productB 1 batch, ... to productA 1 batch, productB 1 batch,....) thus making the replenishment time due to batchLife close enough to replenishment due to chemicalLife.

In later parts of model fitting and optimization, we are to address this problem by machine learning the time series for productionSequence.

```
def chooseProduct(tank, product, weekProducts):
    if weekProducts:
        #print(weekProducts[0])
        return weekProducts[0]
    else:
        print("demand met alr")
        return []
```

For reclaim efficiency, since unlike batchLife and chemicalLife which are fixed during the whole simulation process it may change(or more precisely, fluctuate) with time, we reserve a function for it. For sustainability purposes, we will target at increasing this value with time.

```
def chooseProduct(tank, product, weekProducts):
    if weekProducts:
        #print(weekProducts[0])
        return weekProducts[0]
    else:
        print("demand met alr")
        return []
```

Part(b) Model training and fitting

Objective:

In this phase of our analysis, we will employ the predictive model constructed in Part A to accurately pinpoint the optimal values for Load Size and Batch Lives that achieve the lowest Mean Average Error (MAE). This determination will be made with a precision of one decimal place, ensuring our findings are both rigorous and actionable within the specified range. Our objective is to enhance the model's predictive accuracy and operational efficiency by systematically evaluating the impact of these key variables on performance outcomes.

Preparation:

Import and define necessary properties

```
import numpy as np
from sklearn.metrics import mean_absolute_error
from Forecast import *
import pandas as pd

actualUsage = pd.read_csv("Datasets/Actual_Usage.csv")

actual_usage = actualUsage.values.flatten()


# Define the ranges for batch life and load size for both products
batch_life_range_A = np.arange(5, 8, 0.1)  # Product A: 5 to 7
load_size_range_A = np.arange(35, 39, 0.1)  # Product A: 35 to 38
batch_life_range_B = np.arange(7, 9, 0.1)  # Product B: 7 to 8
load_size_range_B = np.arange(45, 53, 0.1)  # Product B: 45 to 52
# Placeholder for the best MAE and corresponding parameters
best_mae = float('inf')
best_params = {}
```

Finding smallest MAE:

Running all combination possibilities and returning the smallest MAE and its respective parameter values

```
for batch_life_A in batch_life_range_A:
    for load_size_A in load_size_range_A:
        print("current batch_life_A", batch_life_A)
        for load_size_A in load_size_range_A:
            print("load_size__A", load_size_A)
            for batch_life_B in batch_life_range_B:
                print("batch_life_B", batch_life_B)
                for load_size_B in load_size_range_B:
                    print(load_size_B)
                    # Run the forecast with the current set of parameters
                    forecasted_usage_flat = forecast(tankSize, numOfProducts, numOfTanks,[batch_life_A, batch_life_B],
                                                     [load_size_A, load_size_B],getReclaimEfficiency, demand, weeks)
                    # print(forecasted_usage_flat)

                    # Calculate MAE for this combination
                    # Replace with your actual usage values
                    mae = mean_absolute_error(actual_usage, forecasted_usage_flat)

                    # Update the best MAE and parameters if the current MAE is lower
                    if mae < best_mae:
                        best_mae = mae
                        best_params = {
                            'batch_life_A': batch_life_A,
                            'load_size_A': load_size_A,
                            'batch_life_B': batch_life_B,
                            'load_size_B': load_size_B
                        }
```

Result:

```
print(f"Best MAE: {best_mae}")
print(f"Best parameters: {best_params}")
```

	Best Load Size	Best Batch Lives
Product A	5.1	35.0
Product B	7.0	45.0
*** Lowest MAE:** 93.840		

From the results we can conclude when Load Size and Batch Lives of Product A is 5.1 and 35.0, and of Product B is 7.0 and 45.0, will generate the lowest MAE value of 93.840, rounded to three decimal places.

Part(c) Long-term Prediction

Preparation for machine learning

Why time series forecasting model?

To predict a sequence of demand for two products in the future on a weekly basis, one commonly used approach is to utilize Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, which are well-suited for sequential data.

At this stage, we implement LSTM for prediction of basic data such as demand and reclaim efficiency.

Key Factors to Consider:

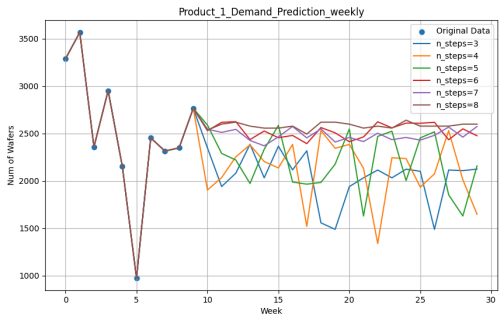
1.

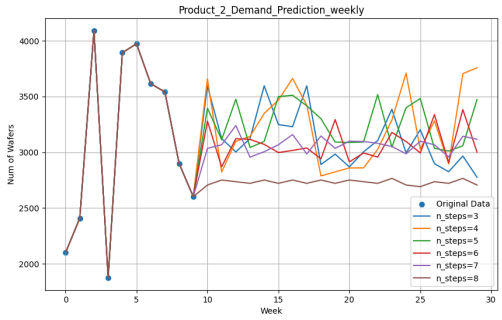
Wafer demand change over time

Hypothesis: Demand will increase over time as product variation increases because:

- product variation increases (more products in the future)
- company develops

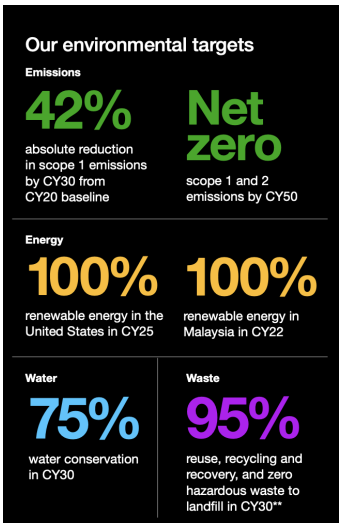
Time Series Prediction Model Result:





Utilizing the Random Forest Regressor, we've conducted a rigorous analysis on the initial 10-week wafer demand to project future requirements. The chosen n_steps of 6 emerges as the optimal parameter, adeptly balancing forecast stability with the inherent demand fluctuations. This strategic selection equips us with a model that's finely tuned to market trends, ensuring dependable supply chain management.

	1 3.2890000000000000e+03 2 3.5660000000000000e+03 3 2.3580000000000000e+03 4 2.9510000000000000e+03 5 2.1520000000000000e+03 6 9.7300000000000000e+02 7 2.4530000000000000e+03 8 2.3140000000000000e+03 9 2.3470000000000000e+03 10 2.7650000000000000e+03 11 2.5288499999999909e+03 12 2.6170500000000182e+03 13 2.6243499999999909e+03 14 2.4348499999999909e+03 15 2.5255500000000182e+03 16 2.4545000000000000e+03 17 2.4787500000000000e+03 18 2.3921500000000091e+03 19 2.5616500000000091e+03 20 2.5065000000000000e+03 21 2.4126500000000091e+03 22 2.4627500000000000e+03 23 2.6226999999999818e+03 24 2.5583499999999909e+03 25 2.6087500000000000e+03 26 2.6070999999999909e+03 27 2.6170500000000182e+03 28 2.4331999999999818e+03 29 2.5480999999999909e+03 30 2.4750500000000182e+03 1 2.1000000000000000e+03 2 2.4090000000000000e+03 3 4.0900000000000000e+03 4 1.8740000000000000e+03 5 3.8930000000000000e+03 6 3.9730000000000000e+03 7 3.6150000000000000e+03 8 3.5410000000000000e+03 9 2.9000000000000000e+03 10 2.6020000000000000e+03 11 3.2792500000000000e+03 12 2.8694499999999818e+03 13 3.1221500000000091e+03 14 3.1154000000000091e+03 15 3.0705999999999909e+03 16 2.9976500000000091e+03 17 3.0162500000000000e+03 18 3.0386500000000091e+03 19 2.9409499999999818e+03 20 3.2928000000000182e+03 21 2.9134000000000091e+03 22 2.9938499999999909e+03 23 2.9566500000000091e+03 24 3.1778499999999909e+03 25 3.0966999999999818e+03 26 2.9938499999999909e+03 27 3.3374000000000091e+03 28 2.9060000000000000e+03 29 3.3829000000000091e+03 30 3.0013499999999909e+03 The raw predictions data for Product A's demand (left), and Product's B demand (right).
2.	**Purchasing Order Consideration** ● Looking into Micron Singapore's branch, and the three biggest Micron chemical suppliers in Asia are from China, Japan, and Singapore. The average delivery transportation time is approximately 12 days (reference from www.fluentcargo) from China and Japan to Singapore. ● 12 days means an average of 3190.63 liters of acid will be used according to the actual usage of data in the first five weeks given ● Meaning we have to pre-purchase 12 days worth of products, which is 3190.63 liters, to still process during the shipping period
3.	**Wastage Limitations** ● According to Micron's 2023 sustainability report, one of the goals is to "Reuse, Recycle, and Recover **95%** of the waste", aiming for "zero hazardous waste to landfill in CY30". Through this motive from Micron and our expense considerations, we will choose to prioritize waste limitation over inventory space.

	• **Prioritize Waste Limitation over inventory space:**
	  ○ When analyzing demand trends and considering processing time adjustments in response to fluctuating demand, it's essential to factor in the constraints of chemical and batch life.
	  ○ Reducing processing time during periods of lower demand might seem efficient, but it can lead to increased waste and additional costs for waste management and raw material expenditure.
	  ○ On balance, maintaining consistent processing times, despite the higher inventory storage needs and associated costs, is the more prudent choice.
	  ○ This approach aligns with a commitment to environmental responsibility, avoiding the excessive generation of waste, and ensuring resource allocation remains aligned with production sustainability.
	 

4.	**Inventory Space**
	• **Product Space**
	  ○ We currently have product A and B, in the long term we will assume the possibility of more product variations
	  ○ More product variations bring in higher demand, but generates expenses on inventory on inventory space, potential wastage and manpower
	• **Prioritize Waste Limitation over inventory space:**
	  ○ When analyzing demand trends and considering processing time adjustments in response to fluctuating demand, it's essential to factor in the constraints of chemical and batch life.
	  ○ Reducing processing time during periods of lower demand might seem efficient, but it can lead to increased waste and additional costs for waste management and raw material expenditure.
	  ○ On balance, maintaining consistent processing times, despite the higher inventory storage needs and associated costs, is the more prudent choice.
	  ○ This approach aligns with a commitment to environmental responsibility, avoiding the excessive generation of waste, and ensuring resource allocation remains aligned with production sustainability.

5.	**Quartz tank replacements for Sulphuric Acid Storage** Exploring the lifespan of a quartz tank, we would consider factors such as acid concentration, tank material, maintenance, supplier quality, and operational conditions. Despite the lack of precise lifespan data for an 80-liter tank, anecdotal evidence suggests that with proper maintenance, tanks can last over 10 years, minimizing the need for frequent replacements. This approach

	underscores the economic and operational viability of the investment, with the tank's initial cost being a minor consideration in the overall efficiency strategy.

Qn2 Optimization of Production Plan

Part(a) Constraints and Restrictions

Before constructing the optimal production plan, it is crucial to consider a list of factors influencing production, such as the cost of ingredients and the time required to complete a lot. Given that each completed lot sells for $40,000, and with three pieces of equipment, there are a total of 108 combinations. Among these, the most time-effective combination is 'D', 'C', 'D', 'B', 'D', requiring only 14 hours to complete a lot. However, this combination necessitates 66, 76, and 18 liters of ingredients X, Y, and Z, respectively. On the other hand, the 'B', 'E', 'E', 'C', 'C' combination requires only 12, 56, and 8 liters of X, Y, and Z, despite taking 25 hours to complete a lot. To maximize profit, a cost-efficient combination is preferred over a time-efficient one.

Several factors must be weighed in decision-making:

- Step Sequence: The requirement for lots to proceed through steps sequentially dictates the order of operations, influencing scheduling and throughput.
- Recipe Completion: It is essential that each recipe is completed before a lot progresses to the next step to ensure proper equipment allocation and timing.
- Final Product: Completing a lot in full sequence is crucial for it to be sellable.
- Recipe Selection: With multiple recipes available for each step, selecting the optimal one is necessary to enhance processing times and equipment usage, considering the different materials and costs involved.
- Processing Time: Varied processing times for each recipe require careful planning to avoid bottlenecks and maximize throughput.
- Equipment Capability: The equipment's limitations on running certain recipes introduce a constraint that must be balanced with lot scheduling and recipe
- choice.
- Material Usage: Accurate tracking and ordering of material consumption by each recipe are vital to ensure availability without waste.
- Material Supply and Costs: The fluctuating costs of different material volumes necessitate strategic ordering to minimize expenses without risking shortages.
- Equipment Availability: Consideration of maintenance schedules, potential failures, and availability issues is essential for a realistic and robust production plan.
- Changeover Times: Optimizing the time needed to switch recipes is critical to reducing idle equipment time.
- Capacity Constraints: Avoiding equipment overload requires a balance between capabilities and production demands.
- Buffer Stock: Maintaining appropriate buffer stock levels is necessary to manage production variability, balancing inventory costs and the risk of material expiration.

Part(b) Simulation Algorithm

Given enormous possibilities for a time span of 168, it's never possible to iterate through all possibilities during simulation . However, since the possibility matrix for state changes at every time step is simply dependent on the previous step (thus irrelevant to past or future time steps), we adopted Markov Chain Monte Carlo algorithm for the whole runthrough process.

As for determination of the new state, we use weightage calculation to assign different priorities to different choices of new states and different recipes. Therefore, as long as we run the programs for enough times, the optimized sample choice list is approximately generated in limited results. This approach minimizes redundant computational expense without compromising overall optimization.
As for the weighting function, we first assign priority values to different parameters:

```
# priority
stateIdPrio = [MINIMUM_POSSIBILITY,0.7,0.2]
recipeTimePrio = [4, 3, 5, 2, 6] ## negatively
recipeCostPrio = [3, 2, 5, 1,4]
stepPrio = [1, 2, 4, 6, 8]
```

And then as for the influence of time, we suggest setting time priority higher (in our case by dividing the priority values again as it's negatively associated with weight calculation) when simulation approach simulation range (>= 168 - 14), where 14 is the minimum number of time steps needed for a whole lot.

```
def weightage(state, time):
    if state.getId() == 1:
        recipe = state.getProcessedLot().getRecipe()
        step = state.getProcessedLot(). getStep()
        if time <= SIMULATION_RANGE - 14:
            return stateIdPrio[1] * recipeCostPrio[recipe]/recipeTimePrio[recipe] * stepPrio[step]
        else:
            return stateIdPrio[1] * recipeCostPrio[recipe]/recipeTimePrio[recipe] * stepPrio[step]

    elif state.getId() == 2:
        recipe = state.getEndLot().getRecipe()
        step = state.getEndLot(). getStep()
        if time <= SIMULATION_RANGE - 14:
            return stateIdPrio[1] * recipeCostPrio[recipe]/recipeTimePrio[recipe] * stepPrio[step]
        else:
            return stateIdPrio[1] * recipeCostPrio[recipe] /recipeTimePrio[recipe]/recipeTimePrio[recipe] * stepPrio[step]
    else:
        if time <= SIMULATION_RANGE - 14:
            return MINIMUM_POSSIBILITY
        else:
            return NORMAL_POSSIBILITY
```

- Simulation Logic: The simulation progresses through a predetermined range of time steps(in our simulation 168 as specified in the problem statement), updating the status of each piece of equipment based on their actions (processing or switching between lots) and the availability of lots for processing. Equipment decisions are informed by the current situation, available lots, and set priorities (such as recipe priority), incorporating

probabilities to reflect realistic decision-making dynamics. It assigns lots to equipment by evaluating overlaps in capabilities and lot requirements, with an emphasis on efficiency and profit maximization.

- Decision-Making Process: Functions like overlap, convert, and choose_with_probability facilitate the decision-making, aiding in identifying viable actions and selecting based on probabilities. An events dictionary tracks potential future states for each equipment and their probabilities, guiding the selection of the next state to minimize disruptions and streamline operations.
- Optimization and Profit Calculation: While the primary objective is to simulate the manufacturing workflow and decision-making, it also lays the foundation for process optimization and profit calculation, taking into account variables such as recipe costs, switching times, and recipe priorities.

Our OOP hierarchy is as follows: