
UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

# *Inventory Management using Artificial Intelligence*

***Arnav Garg***

(Student number: 2091895)

School of Mechanical, Industrial and Aeronautical Engineering

University of the Witwatersrand

Johannesburg, South Africa.

**Supervisors:**

Dr Bevan Smith

Mr William Rich

A Research Project submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, in fulfilment of the requirements for the degree of Masters in Engineering in Engineering.

*27 August 2024*

## Acknowledgements

I had the pleasure to conduct my research with Deloitte and would like to thank them for this wonderful research topic and constant supervision of my project. I would like to thank Mr William Rich and Dr Bevan Smith for assisting and guiding me throughout my research project to ensure it was successful and of the highest quality. Your insights and knowledge have allowed me to learn and grow as an individual.

## Abstract

Poor inventory management negatively affects a company's profits. Too little stock limits potential sales and customer satisfaction while too much stock increases storage costs and potential damage. Company X distributes butter in a multi-echelon supply chain consisting of multiple entities such as a manufacturing plant, distribution centre and retailers before reaching the customer. For every 1% of the demand that is not met, revenue in excess of R18 million is lost per year. This study aims to use machine learning methods (supervised and reinforcement learning) for optimal decision making that maximizes profits. Supervised learning methods (random forest, recurrent neural network and support vector machines) were used to forecast the demand based on historical data. Thereafter, deep reinforcement learning (DRL) was used to train an agent to decide when and how much to order over a period of a year. Various algorithms (PPO and DDPG) and unique reward functions were tested and the performance was compared to a benchmark heuristic that stocks inventory based on a sum of the forecasted demand. The random forest algorithm performed the best at predicting the forecasted demand. The DRL model using a continuous action and state space together with the DDPG algorithm and a reward function based on a combination of the current profit, order fulfilment rate, units available and units unsatisfied performed the best. The DDPG algorithm outperformed the PPO algorithm with the DDPG model being able to provide a 21% increase in net profit over the benchmark heuristic when the production and warehouse facilities of the supply chain were merged. The DRL models were not able to provide a higher order fulfilment rate compared to the benchmark heuristic but they were able to provide better asset utilization by sending full trucks and minimizing the inventory held to maximize the profitability in the supply chain. The results suggest that RL has potential of better handling the stochastic constraints (demand and lead times) in real supply chains to automate the ordering process. It was found that increasing the order fulfilment rate does not necessarily lead to higher profits and the reward function has a significant effect on the net profit which can be further optimized in the future.

# Table of Contents

# List of Figures

## List of Tables

## Nomenclature

| Symbol/Acronym | Description |
|---|---|
| AI | Artificial Intelligence |
| ML | Machine Learning |
| RL | Reinforcement Learning |
| LR | Linear Regression |
| KNN | K-Nearest Neighbors |
| RF | Random Forest |
| SVR | Support Vector Regression |
| MSE | Mean Squared Error |
| RMSE | Root Mean Squared Error |
| MAE | Mean Absolute Error |
| $r^2$ | Coefficient of determination |
| NN | Neural Network |
| ANN | Artificial Neural Network |
| DQN | Deep Q-Network |
| PPO | Proximal Policy Optimization |
| DDPG | Deep Deterministic Policy Gradient |
| SB3 | Stable-Baselines3 |
| MLP | Multi-Layer Perceptron |
| TD | Temporal Difference |
| KL | Kullback-Leibler |
| JIT | Just-in-time |
| ROP | Reorder point |
| DSI | Days Sales of Inventory |
| EOQ | Economic Order Quantity |
| DRL | Deep Reinforcement Learning |
| A2C | Advantage Actor Critic |
| A3C | Asynchronous Advantage Actor Critic |
| DFO | Derivative Free Optimization |
| ERP | Enterprise Resource Planning |
| CBC | CardBoard Company |
| HP | Hewlett-Packard |
| RAM | Random Access Memory |

| | |
|---|---|
| ARIMA | Auto Regression Integrated Moving Average |
| FIFO | First in, First Out |
| FMCG | Fast-Moving Consumer Goods |
| RNN | Recurrent Neural Network |

# 1 Introduction

This research aims to explore the possibility of developing an automated model using machine learning techniques to determine an optimal ordering strategy that maximizes the profitability and order fulfilment rate in a particular supply chain.

## 1.1 Research background

A supply chain is a sequential network of processes involved in the production and sale of a product. Inventory management is a strategy used to manage the planning, manufacture and distribution of goods in a supply chain to ensure optimal amounts of goods are stored to meet customer demand while maximizing profits [1]. Inventory is stored to respond to fluctuations in demand and supply, accounting for seasonal trends as well as taking advantage of economies of scale related to purchasing, transportation and manufacturing. An inventory policy is a guideline regarding what to manufacture or purchase, when to order, in what quantity and where the products should be located geographically [1]. The order fulfilment rate or fill rate is the percentage of customer demand that is satisfied from inventory on hand. By holding more inventory, a higher fill rate can be attained but the inventory costs also increase hence, the optimal balance between inventory costs and the fill rate is required.

Many inventory control problems limit the efficiency of supply chains. A bullwhip effect is a common phenomenon caused by the fluctuating demand at the customer end of a supply chain. Figure 1 shows that when the customer demand increases, the retailer overestimates the customer demand causing the retailer to increase their orders. This results in the supplier overestimating the retailer demand causing the supplier to overproduce their products. Similarly, when the customer demand decreases, it leads to the retailer decreasing their orders and the supplier underproducing their products. Customer demand uncertainty increases upstream of the supply chain resulting in a larger shortage or excess of goods. Poor communication and incorrect demand forecasts create an imbalance between supply and demand which is amplified at each step moving upstream of the supply chain. Shorter product cycles in recent times due to technological advancements, changing trends and competition cause larger fluctuations in shorter periods of time making it more difficult to predict the future demand. Manual stock-taking and the use of outdated software is time-consuming and can lead to inaccurate records of stock being available at various positions of the supply chain resulting in larger lead times and storage costs [3]. Stock such as food items requires careful and frequent control to ensure good quality and safety [3]. Unclear communication and a lack of expertise can harm employee relationships affecting their productivity with larger probabilities of placing incorrect orders, late deliveries, and damage to goods [3].

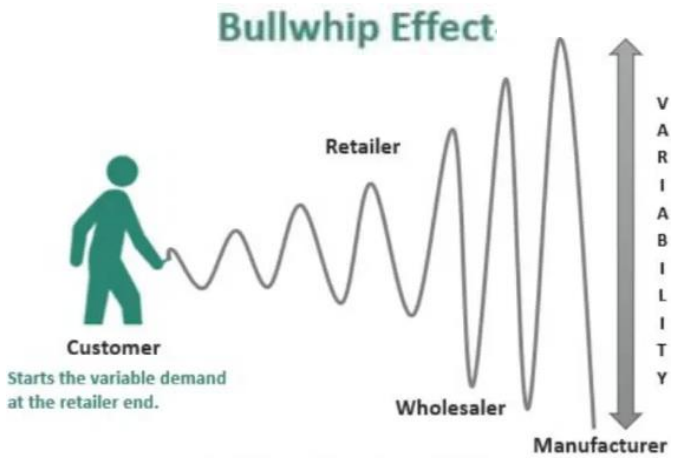


Figure 1: Diagram showing the bullwhip effect in supply chains [2].

Research shows that manual inventory tracking is still conducted by 43% of small businesses as of the year 2017 which can be time-consuming and susceptible to human errors [4]. Most companies have 20-40% of their working capital attached to inventory which lead to large revenue losses if inventory is poorly managed [5]. 34% of businesses miss a shipment deadline because they sell a product that was out of stock limiting their profitability [6]. Future supply chains need to be more agile, adaptable and aligned (Triple-A supply chain framework) through clear visibility and constant supervision to minimize wastage and increase revenue of businesses.

### 1.2 Case description

This research considers inventory management for company X that produces and distributes butter. Due to confidentiality, the real name is not mentioned but real data was used from the company. The supply chain structure of this company is shown in Figure 2 which consists of a single production unit (butter blocks are produced in the factory), a single warehouse (mass storage location for the butter blocks) and five outlet stores (different locations from which customer demand is satisfied) with delivery trucks in operation between the various locations in the supply chain. The warehouse facility receives finished goods from the production facility whereas the five outlet stores can be supplied from either the production or warehouse facility. Customers can place orders at the warehouse or any of the five outlet stores. To perform inventory management, relevant supply chain data was obtained including historical customer demand, product price and lifespan, costs (manufacture, storage, and transport), limits on production sizes, capacity (storage and truck) constraints, and lead times (processing and transport). Company X wants to find the optimal way to produce, distribute, and store butter to maximize profits. Relevant supply chain parameters and their corresponding values from Company X that are used in this research are included in Appendix A.

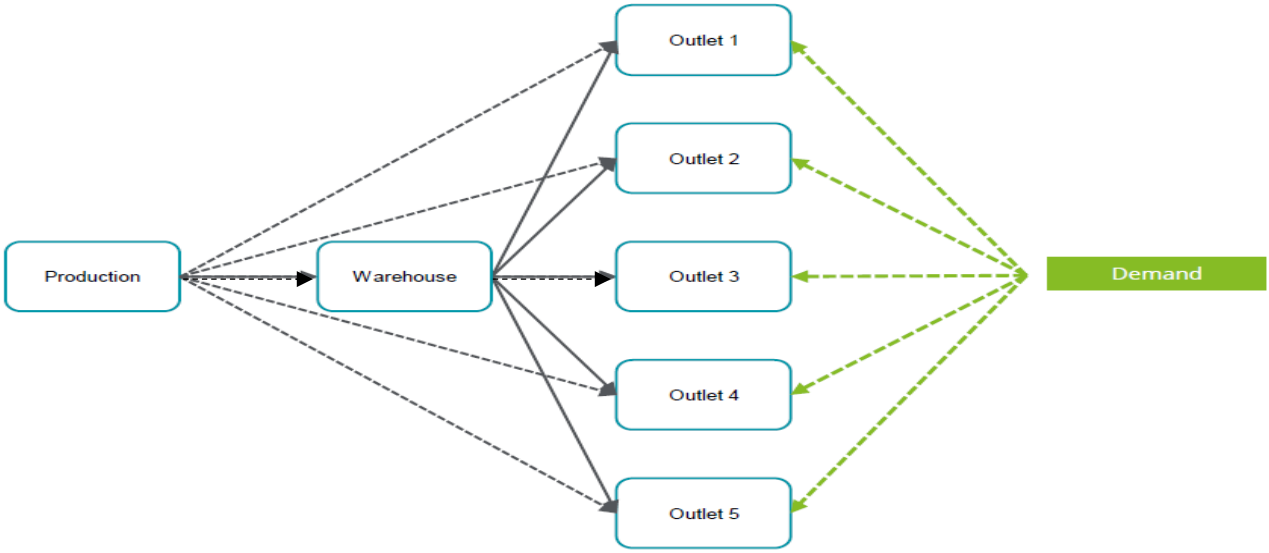



Figure 2: Diagram showing the supply chain of Company X.

A heuristic is a practical strategy used to solve problems and make observations quickly and efficiently [7]. Company X uses a heuristic that involves storing inventory at each stock point based on the sum of forecasted demand for the next few days. For each of the outlets, seven days of their respective forecasted demand was used while the warehouse was based on its twelve-day forecasted demand. This guides stock replenishment processes at each timestep in an attempt to maximize profit. However, this method can potentially be improved upon by using machine learning approaches that incorporate the stochastic nature of components in the supply chain to determine a variable inventory policy (optimal order quantities at each timestep) that will maximize profits [8]. A modification of the supply chain structure such as merging the production and warehouse facility also has the potential to reduce logistics costs and improve customer accommodation and will be tested in silico to determine if it can further improve the net profit.

Some interesting statistics regarding the supply chain of Company X include:

- The storage cost of each butter block in the production facility of the company is R0.11 per week and the storage capacity is 3, 175, 200 units, hence if the storage cost can be reduced by even 1%, it provides a significant saving of money of approximately R3, 500 each week which is approximately R182, 000 for a year. Using a similar approach, the storage cost for the entire supply chain reduces by approximately R500, 000 for every 1% reduction in storage costs.
- Historic demand data (sales) for the past year (361 days) is available with the total units sold amounting to 73, 266, 690 units providing a total revenue of R1, 831, 667, 250 using a selling price of R25 per unit. For every 1% of the demand that is not met, revenue in excess of R18 million is lost for a year, hence accurate forecasting and inventory

management have the potential for achieving significant increases in profits. The current fill rate in the company is approximately 95%, based on the current heuristic.

- The trend of the historical demand for a single outlet (Pretoria) can be observed in Figure 3. Promotions are run at various periods leading to large fluctuations which can be taken advantage of to maximize the service level and profitability of the company. For example, a large spike in the graph can be observed in the month of November which could potentially be linked to *Black Friday* deals.

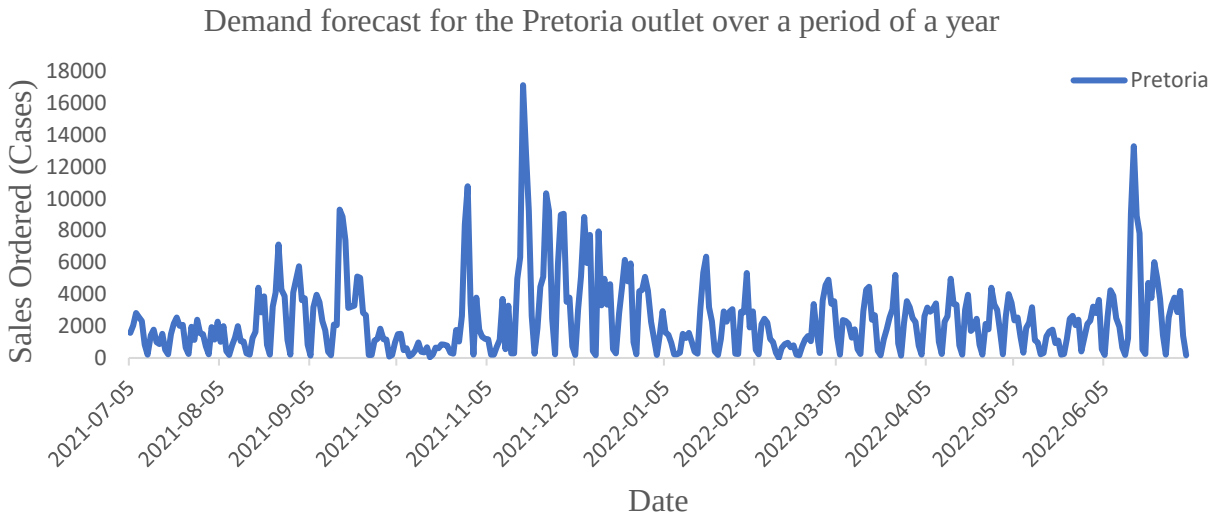



Figure 3: Actual historic demand for the Pretoria outlet.

### 1.3 Problem description

The problem statement can be summarized as follows: poor inventory management can negatively affect a company's profits. Poor demand forecasting and inefficient inventory policies (when and how much to order) cause a bullwhip effect reducing the stability of supply chains. Having too little stock on hand can lead to a loss of potential sales and customer satisfaction while too much stock can lead to increased storage costs, wastage and potential damage of goods. Traditional inventory control methods, mathematical models and heuristics have provided effective methods to improve inventory management in the past. However, it was difficult to incorporate stochastic supply chain constraints (demand and lead times) causing the methods to become too complex and time-consuming relying heavily on assumptions/simplifications making them less realistic.

### 1.4 Artificial Intelligence

The current trends in supply chain management include going green, circular supply chains, increased supply chain integrations, shorter product lifecycles, elastic logistics, better transparency, blockchain tools, robotic automation and artificial intelligence [9]. Artificial Intelligence (AI) is the ability of digital system to mimic human-like behaviour by performing

tasks such as learning, reasoning and acting in an environment to accomplish a particular goal. AI technology aims to enhance productivity and performance by automating tasks that previously required a human element. AI has shown to outperform human abilities at natural language and image processing making it the future of all complex decision making as it has the potential to achieve goals more easily, quickly and effectively [10]. AI is currently the main trend in supply chain management as it has the potential of managing large amounts of stochastic supply chain data in real time [11]. It also allows for testing of different scenarios to determine an optimal ordering strategy for a particular supply chain. Machine Learning (ML) is a subset of AI that uses data and algorithms to create models which allow a machine to conduct human-like tasks such as identifying patterns, analysing/categorizing data and making predictions about new data. An increase in data, computational power and development of new techniques enhances the power of solving complex problems and discovering new insights. ML can be used in supply chains to assist with inventory management by accounting for various constraints (capacity, cost, lead times, product lifespan) and forecasting the demand to learn how operations should be planned ahead of time to maximize profits.

ML is divided into supervised, unsupervised and reinforcement learning depending on the type of feedback and goal of the learning process. Figure 4 illustrates that supervised learning is like learning from a teacher who guides your learning, unsupervised groups the observed data according to similarities and differences and reinforcement learning involves an autonomous agent learning to perform a task through trial and error in an interactive environment. Supervised learning utilizes matched pairs of input and output data (labelled data) to predict and forecast the trends of a specific topic whereas unsupervised learning attempts to find patterns in uncategorized data (unlabelled data) [12]. Supervised learning is most commonly used in classification and regression analyses whereas unsupervised learning is typically used in association and clustering applications. Reinforcement learning (RL) uses a continuous feedback loop incorporating an agent (such as the supply chain manager), actions (such as the amount of stock replenishment), states (such as the position within the supply chain including the availability and position of stock, etc) and rewards (such as the extent of profit or loss achieved) indicating how good/bad the action was in an environment (such as the representation of the inventory system in a supply chain) [12]. It is able to handle large and stochastic input datasets in complex environments in which the model learns [13]. RL can learn to take sequential actions to maximize returns (which are accumulated rewards) [13] providing recommendations on when and how much stock to replenish at each timestep thereby finding the balance between inventory costs and the fill rate to maximize profit.

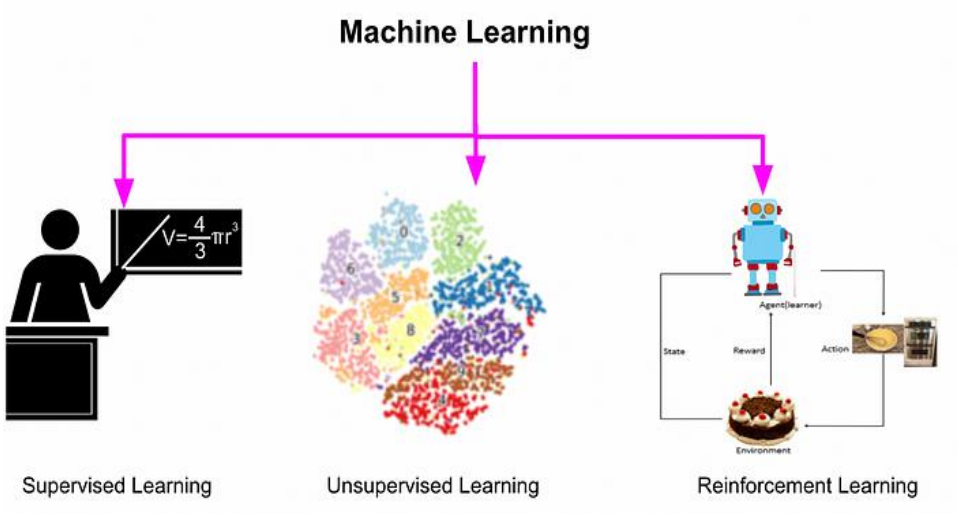



Figure 4: Different ML techniques [14].

RL uses a similar method in which humans learn and have shown promising results in other fields such as computer chess players and autonomous vehicles [15]. RL and humans both use feedback, learn from trial and error and adapt their strategies based on changing environments. However, RL may not be able to capture all the complexities of reality, can struggle to generalize information requiring large amounts of data and time to learn effectively. Additionally, computers act logically without the effect of negative factors such as hype, greed, indecisiveness and other human emotions. In 2016, Google's AlphaGo RL model was able to defeat the world's best human player at the game of go [16]. This shows RL's capabilities of determining brand new solutions that is outside of the input dataset (which is used to train the RL model) making it a promising technique for this research. Forecasted demand data can assist with storing optimal units, cost savings, promotional planning and overall supply chain efficiency. Supervised learning techniques is used to forecast the demand which then forms the input of the baseline heuristic model (to calculate the inventory target) and a RL model (part of the state space) to learn an optimal ordering strategy. Further investigation is required to test whether RL can improve inventory management in the supply chain of Company X throughout a year to maximize profits.

### 1.5 Research Question

How does the net profit of a reinforcement learning approach compare to the heuristics technique when applied to the supply chain of company X and to a modification of the supply chain structure (merging the production and warehouse facility) to improve inventory management over a period of a year?

### 1.6 Research Objectives

The overall objective of this research is to apply reinforcement learning for inventory management (when and how much to order) in the supply chain of company X to maximize profits and compare its net profit achieved over a period of a year with the heuristic technique. The sub-objectives of this research include:

- To predict the forecasted demand using a supervised learning model based on actual historic sales data.
- To compare the net profit of the reinforcement learning model with the heuristics technique when applied to the supply chain of Company X in its current state.
- To merge the production and warehouse facility in the supply chain of Company X and compare the net profit achieved using the reinforcement learning model with the heuristic technique.

### 1.7 Contribution to literature

The following contributions are made in this research in the fields of inventory management and deep reinforcement learning:

- A combination of supervised learning and reinforcement learning was applied to an inventory management system.
- Reinforcement learning is applied to a case that has not been considered before using a unique combination of real supply chain parameters (Section 1) and unique action, state and reward variables (Section 3). This adds value to the inventory management community as it allows for a more complex supply chain to be modelled using real supply chain data from a company. This also adds value to the reinforcement learning community indicating useful supply chain parameters to include in the action, state and reward function of an inventory management RL environment.
- To the best of our knowledge, this is the first research that applied the Deep Deterministic Policy Gradient algorithm to a multi-echelon inventory system with a general network structure (stock points having many successors and many predecessors) using real supply chain data (Section 2). The DDPG model achieved a marginally lower net profit but utilized approximately half of the supply chain compared to the benchmark heuristic.
- RL helps determine what aspects of a supply chain is working well and what is not with the potential of advising certain business decisions in the future relating to supply chain structure, pricing and distribution of goods based on current trends (Section 4). For example, the production storage costs were too high resulting in very little units being

available at the warehouse. Thus, a RL model can be simulated with a reduction in production storage costs and optimized in the future to determine whether the profitability increases.

- Increasing the fill rate in the supply chain does not necessarily lead to higher profits which was shown in Section 4.

# 2 Literature review

This section provides detailed information about the machine learning techniques and the Python software that will be used in this research. Prior works are also presented that shows related work performed on the subject in the past.

## 2.1 Supervised learning

Supervised learning was briefly discussed in Section 1.4 and was used to forecast the demand in this research. The supervised learning workflow is shown in Figure 5 which usually involves obtaining raw data, pre-processing the data and splitting it into a training and validation set [17]. Different algorithms are used to build models on the training set with their performance compared on the validation set. The model with the best performing algorithm is used to make predictions on new data to achieve the goal providing a benefit (profit). Algorithms attempt to imitate intelligent human behaviour of decision making with hyperparameter tuning being performed to optimize their performance. Linear Regression, K-Nearest Neighbors Regression, Random Forest and Support Vector Regression was considered in this study as they are the most common regression algorithms [18].

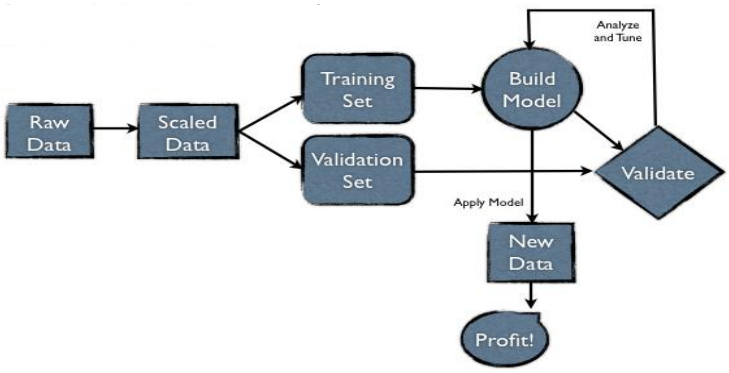



Figure 5: Supervised Learning Workflow [17].

## 2.2 Supervised learning algorithms

### 2.2.1 Linear Regression (LR) model

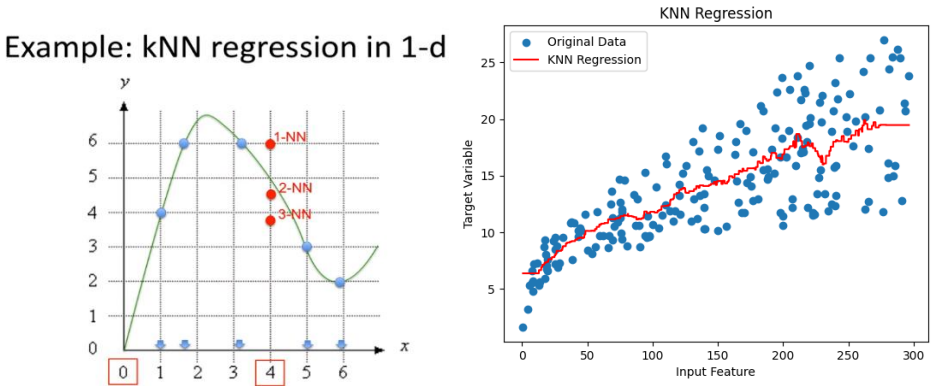
LR is a parametric model since it assumes the functional form to be linear and consists of a finite number of parameters. A straight-line function, $h_\theta$, is fit with the aim of estimating the weight parameters, theta, $\theta$, for each training feature as shown in Equation 1 that best represents the general trend of the dataset. During training, the theta coefficients are continually iterated such that the mean squared-error (MSE) between the actual y-values of the data points and the predicted y-values on the straight line is minimized [19]. This is a simple ML model with little complexity making it the

baseline model for this application even though the dataset is nonlinear as it consists of seasonal demand with fluctuations based on many external factors.

$$h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \quad (1)$$

### 2.2.2 K-Nearest Neighbors (KNN) Regression model

KNN is a non-parametric model since it does not assume a fixed functional form and the number of parameters increases as the training dataset increases. Predictions are made based on taking the average of the output values from the nearest data points with 'k' referring to the number of neighbors considered [20]. An example is shown in Figure 6a where the y-value for x = 4 is predicted using the blue dots which represent the input training data [21]. Using k = 1, the nearest neighbour on the x-axis is x = 3.2 which has a corresponding y-value of 6. Thus, the predicted y-value for x = 4 is 6. Using k = 2, the nearest neighbours on the x-axis is x = 3.2 and x = 5 with corresponding y-values of 6 and 3 respectively. Thus, the predicted y-value is calculated by averaging the two y-values amounting to 4.5. Similarly, using k = 3, the predicted y value for x = 4 is 3.8. It is noticed that increasing the k-value does not necessarily lead to improved performance but rather there exists an optimal 'k' value that provides the most accurate prediction. During training, a range of k-values is defined (for example 1 to 10), cross validation is used to test each k-value by splitting the training data into multiple folds and training the model on each fold while validating on the remaining data using a performance metric such as MSE. The optimal k value is the one that has the minimum error on the validation set. This model is expected to provide improved performance over linear regression as it should be able to account for some nonlinear effects in the dataset as shown by the non-linear red line in Figure 6b.




(a) (b)

Figure 6: K-Nearest Neighbours regression models [21].

### 2.2.3 Random Forest (RF) Regression model

A decision tree model's decision and their potential consequences by splitting the data into subsets based on a feature to maximize the separation of the target variable. The goal is to select an attribute that reduces uncertainty (entropy) the most after the split. The attribute with the highest information gain (largest reduction in entropy) is selected as the splitting criterion. It can provide better interpretability of data, capture non-linear effects and provide insights into feature importance. RF is an ensemble of decision trees that uses the bootstrapping technique whereby each decision tree has a bootstrap sample that is randomly selected with replacement from the training dataset. Each decision tree is trained independently and the out-of-bag sample (data not included in the bootstrap sample) is used to evaluate the performance of the tree. Each tree consists of nodes representing a decision, branches representing the outcomes of the decision and leaf nodes indicating the final prediction. The final results from each tree are averaged to provide the overall prediction. Figure 7 illustrates multiple decision trees, each consisting of a random subset from the input dataset with orange dots representing the selected decision at each node with the prediction from each tree being averaged to determine the overall prediction [22]. An ensemble of decision trees using random bootstrap samples provides the opportunity to consider diverse trees which can allow for complex relationships to be captured more effectively compared to a single decision tree as well as reduce the impact of noise, variance and overfitting of the data [22].

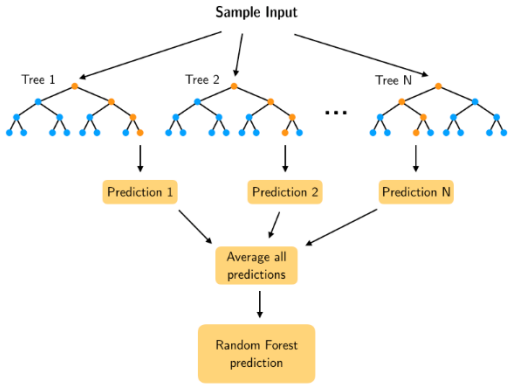



Figure 7: Random Forest regression model [22].

### 2.2.4 Support Vector Regression (SVR) model

SVR attempts to find a hyperplane that is closest to most of the data points within a particular tolerance level (decision boundary) as shown in Figure 8 [23]. This model is useful for handling complex nonlinear relationships but is vulnerable to noise and very sensitive to its hyperparameters. SVR uses the $\epsilon$-insensitive loss function where the loss is zero if the predicted value lies within a margin, $\epsilon$, while deviations from the margin contribute to the loss. This leads to a "tube" around the

regression line with support vectors being the data points that lie on the boundary of the ϵ-tube or outside it which influence the final regression function.

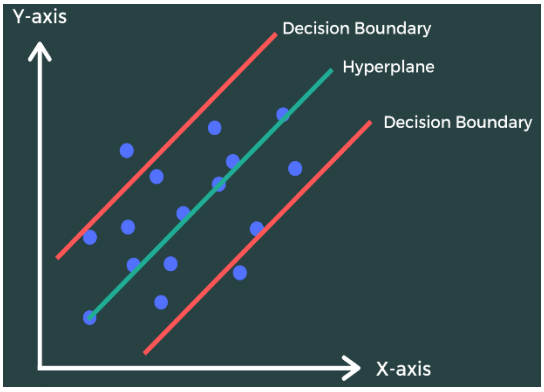



Figure 8: Support Vector Regression model [23].

### 2.3 Supervised learning evaluation

Evaluation metrics are required to evaluate different models, compare their performance and determine the practicality of the ML solution. The mean absolute error (MAE) calculates the average magnitude of the errors between the actual and predicted values without considering their direction as shown in Equation 2 where N is the number of data points, $y_i$ is the actual value and $\hat{y}_i$ is the predicted value [24]. The absolute value treats all errors equally without significantly penalizing larger errors. The mean squared error (MSE) calculates the average of the squared errors between the actual and predicted values as shown in Equation 3 [24]. Squaring the differences penalizes larger errors more but also provides a smooth gradient for better optimization in training models. The root mean squared error (RMSE) as shown in Equation 4 calculates the square root of the MSE providing an absolute error with the same units as the original data making it easier to interpret [24]. The coefficient of determination ($r^2$) indicates the proportion of the variance in the dependent variable as predicted by the independent variables as shown in Equation 5 where $\bar{y}$ is the mean of the actual values [24]. It shows how well the predictions approximate the actual values but does not provide information about the size of the errors. These 4 metrics were used in this research to quantify the demand forecast accuracy.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \quad (2)$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (3)$$

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad (4)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \quad (5)$$

For each of the supervised learning models, k-fold cross validation and hyperparameter tuning was performed to balance the bias-variance trade-off to prevent the model from underfitting (high bias, low variance) or overfitting (low bias, high variance) the training data [25]. K-fold cross validation involves splitting the training dataset into 'k' folds with training being performed on the k-1 set and testing on the remaining set as shown in Figure 9. The model is run through all k-1 sets until it has trained and tested on all the data with the average performance being recorded. K-fold cross validation is more useful to evaluate the training of different models as it provides a more robust and reliable estimate of model performance, assists in parameter tuning, and allows for improved utilization of data. Hyperparameter tuning involves testing different hyperparameter combinations to improve the learning process of the models by minimizing the loss function to maximize the accuracy. The randomized search cross validation method was used to select the best hyperparameters as literature shows that it is usually able to provide models that are as good or even better than the grid search method over the same domain with a reduced computational time [26].

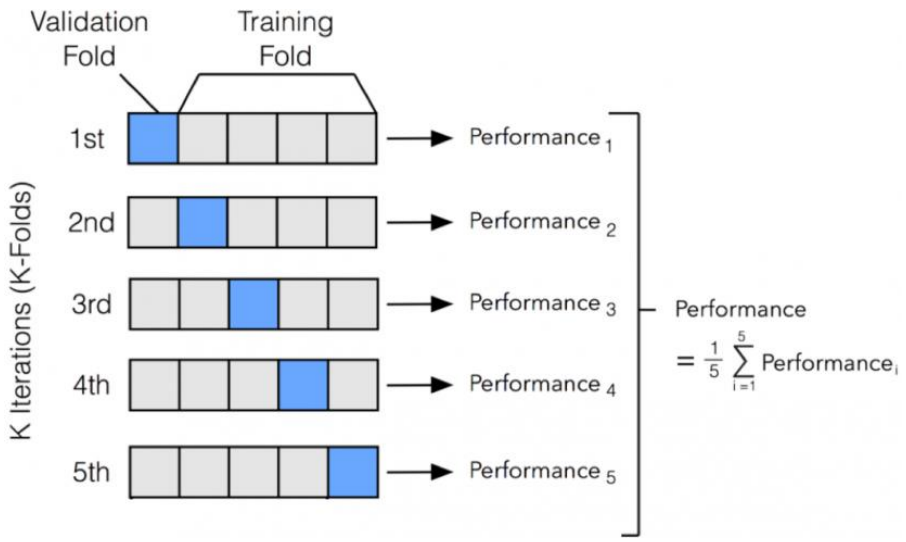



Figure 9: Process of K-fold cross validation [25].

### 2.4 Reinforcement learning

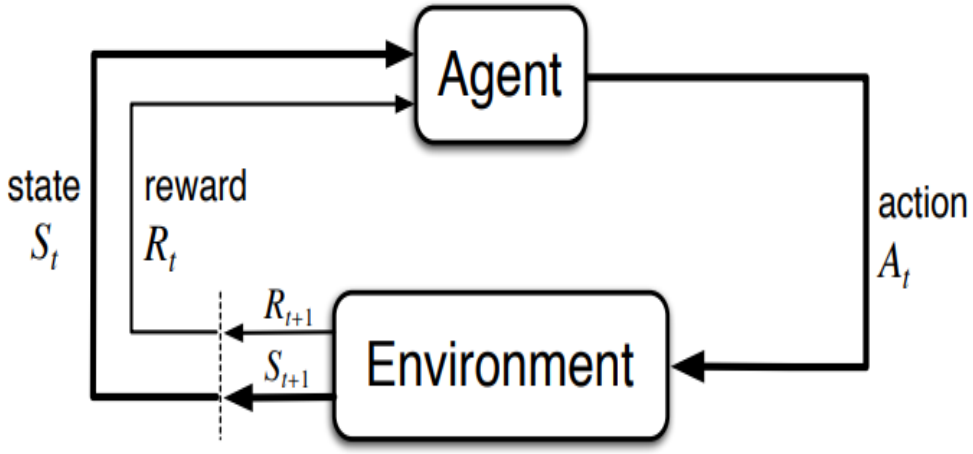



Figure 10: Reinforcement learning feedback loop [27].

Supervised learning was briefly discussed in section 1.3 and is used to determine the forecasted demand. The forecasted demand then forms an input to determine an optimal ordering strategy that maximizes profits in the supply chain by incorporating the RL feedback loop as shown in Figure 10. RL involves an agent making decisions in an environment to maximize a goal. Depending on the current state of the environment, $S_t$, the agent selects a particular action, $A_t$, receives feedback in the form of a reward, $R_{t+1}$, and transitions to a new state, $S_{t+1}$ within the environment. The agent trains through many iterations using rewards which indicate how good or bad the actions are in each state. The goal is for the agent to learn a policy that indicates the best actions to take in a given state of the environment. The Markov property is an important concept in RL stating that the future state of a model only depends on the current state and action taken with no consideration to the preceding events. Thus, the state includes all relevant information required to make a future prediction. A Markov Decision Process (MDP) is a mathematical framework used to model decision-making in scenarios where the outcomes are partly under the control of the decision-maker and partly random. It includes components such as states, actions, rewards, probabilities, policies, etc which are explained in this section. Relevant terminology when discussing RL is presented in this section and the RL algorithms are presented in the next section.

#### 2.4.1 Agent

The agent is the decision-making entity that operates in an environment to learn a strategy and achieve a specific goal. A single agent analysis involves taking one action whereas a multi-agent analysis involves multiple actions being taken within an environment simultaneously at each timestep [28]. A single agent analysis, such as a supply chain manager overlooking the entire supply chain or a multi-agent analysis, such as having a supply chain manager for each facility in the supply

chain can be used. Multi agent analysis reduces the size of the action space for each agent but increases the overall state-action space as the interactions between the agents increase making it usually more difficult to train and tune as compared to single-agent RL models [73], since the success of the entire model depends on good training and tuning of every agent, and also on the cooperation between agents. Since the overall profitability and fill rate needs to be maximized in this research, a single agent analysis that focuses on achieving a global optimum is preferred over a multi-agent analysis that focuses on a local optimum of each facility in the supply chain.

### 2.4.2 Environment

The environment is the external structure with which an RL agent interacts [28]. It contains the agent, action space, state space and rewards to achieve a particular goal. For this research, a custom inventory management environment was developed incorporating the stochastic demand, parameters and operation of this unique supply chain to replicate a realistic scenario.

### 2.4.3 Action space

The action space is a set of all actions that are allowed in the environment such as the ordering quantity from an upstream facility in a supply chain. These can either be discrete action spaces, where agents could decide to order in fixed quantities or they can be continuous action spaces where agents could decide on the rate of ordering or production with any value being selected within a certain range [28]. A continuous action space is preferred for this application as it has the potential to allow the agent to adapt its actions to the specific demands and conditions of the environment to maximize profitability. Additionally, a continuous action space can be used on larger action spaces making it more scalable.

### 2.4.4 State/Observation space

A state is a partial or full description of the environment in which the agent is operating in at a particular timestep [28]. The state of the current environment is dependent on the previous state and the previous action. The state space needs to be carefully defined to provide an efficient decision-making process for the agent. It can include any combination of the supply chain variables and parameters used in the environment such as the forecasted demand, current profit, etc.

### 2.4.5 Rewards

Rewards are used to assist the RL model with achieving the overall objective such as maximizing profits in this study. Rewards can be positive (such as an increase in current profit) or negative (such as a decrease in current profit) depending on the consequence of taking a particular action at a given

state within the environment [28]. Reward shaping can be applied which tests reward functions using a rating based on a sliding scale indicating that some scenarios are more beneficial than others [29]. In the field of inventory management, reward functions can be based on current profit per timestep, fill rate, units available, etc. Different combinations of these can be used to test the RL model with their effectiveness compared to one another [28]. The reward function has a significant effect on the learned policy as the agent aims to maximize cumulative rewards during training. The Return is the cumulative reward received by the agent over a single episode [28]. Equation 6 summarizes this where *R*, is the return; *t*, is the specific time step in the episode; *N*, is the termination timestep; $r_t$ is the reward at timestep *t*, and $\gamma$, is the discount factor which determines the importance of future rewards.

$$R = \sum_{t=0}^{N} \gamma^t r_t \tag{6}$$

### 2.4.6 Timesteps and Episodes

RL models learn using a trial and error process with each timestep representing one increment as the agent takes an action in a given state and receives a reward to reach a new state within the environment [28]. More training timesteps usually produce an improved learning process but at the expense of higher computing power and longer simulation times especially with larger datasets in complex environments. An episode is one iteration of the agent starting from an initial state in the environment, such as starting at day 0, until it reaches a terminal state in the environment, such as reaching a period of a year.

### 2.4.7 Exploration and Exploitation

During each timestep within an episode, the agent either explores or exploits the environment. Exploring involves selecting an action randomly to try new decisions that may lead to better outcomes in the future while exploitation involves selecting the best action based on current information and past experiences [30]. An epsilon greedy approach is commonly used to balance the trade-off between exploration and exploitation using an exploration rate, $\epsilon$ [30]. This parameter has a value between 0 and 1 with a higher value encouraging exploration. An agent selects a random action with probability $\epsilon$ (exploration) and the best action according to the current learned policy with a probability of $1 - \epsilon$ (exploitation).

### 2.4.8 Policy

A policy is a rule that an agent uses to determine what action to select at a given state and is based on the weights and biases of a NN in Deep Reinforcement Learning (DRL) which is updated during

the training process [28]. RL algorithms are used to perform this process which is explained in the next section. RL agents can learn by observing the rewards and punishments received for taking a set of actions to determine an optimal policy which is a unique sequence of actions that maximizes long-term rewards [28]. This is the main goal of a RL model which determines how successful it is for a specific task.

### 2.5 Reinforcement learning algorithms

A neural network (NN) is computational model that is inspired by the human brain as signals are sent from one neuron to another to process and analyze complex data. An artificial neuron is shown in Figure 11 which is the building block of a NN [31]. A NN consists of weights which represent the importance of data transfer from one neuron to the next and biases which provide a trainable constant to allow the model to better fit the data [28]. Figure 12 shows the structure of an artificial neural network (ANN) where input values are propagated from the input layer through the hidden layers to the output layer using the current weights and biases (Forward pass) [32]. During the training process, the model alters its weights and biases (backpropagation) using optimization techniques (Gradient ascent/descent) to minimize the difference between the actual and predicted output (loss/error) [28]. This involves determining the derivative (gradient) with respect to the loss function and then using the derivate to maximize (gradient ascent) or minimize (gradient descent) the function.

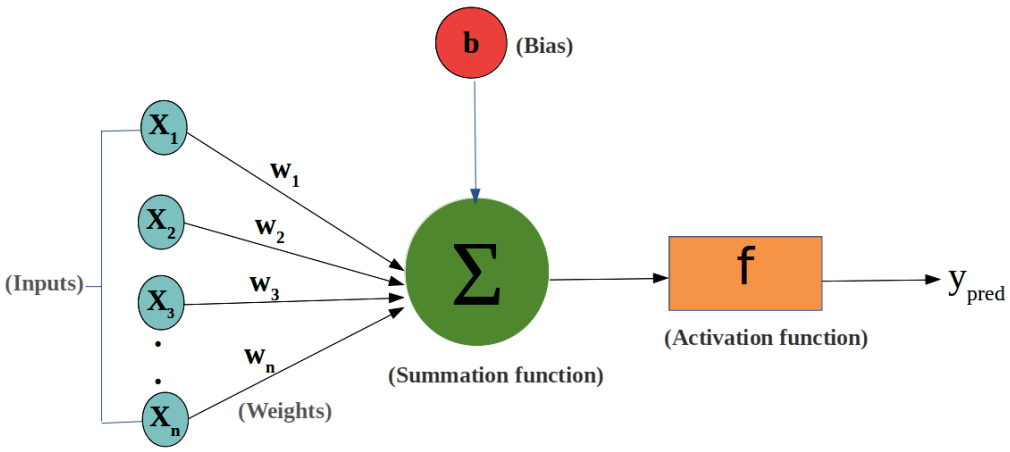



Figure 11: Components of the basic Artificial Neuron [31].

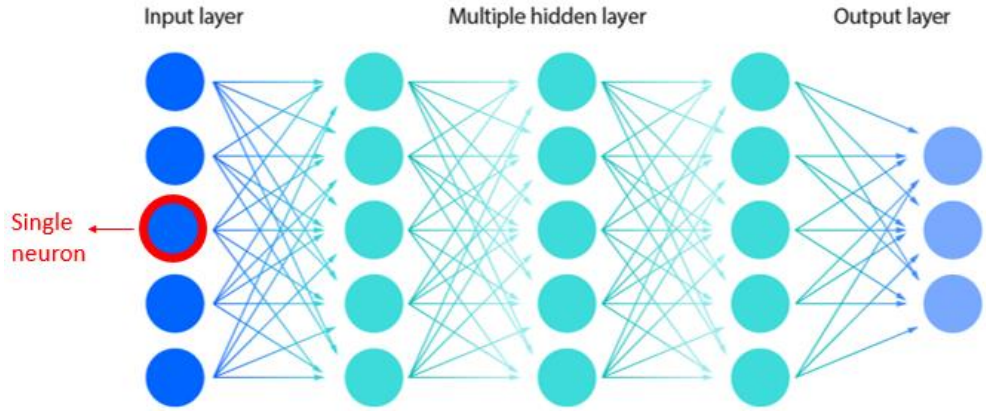


Figure 12: Structure of an ANN model [32].

RL algorithms aim to maximize the return over an episode by adjusting the weights and biases of its NN in a different way during the training process depending on how the optimal policy is approximated, the exploration strategy, and the method used to update the network parameters. A discount factor, $\gamma$, has a value between 0 and 1 with a higher value placing more value on future rewards compared to current ones [28]. A learning rate parameter, $\alpha$, is also used which has a value between 0 and 1 with a higher value representing an increased rate at which the weights and biases of a NN are updated [28]. The two main approaches used to train RL algorithms are value learning such as Q-learning and policy learning such as policy optimization.

### 2.5.1 Value learning

In value learning, the value function indicates how well the action taken by the agent in a given state is and an estimate of the expected return from a given state. The Bellman equation is used to iteratively compute the state-value function, $V^{\pi}(s)$, under the policy, $\pi$, until convergence is reached as shown in Equation 7 where $\mathbb{E}$ is the expectation, $\gamma$, is the discount factor, $R_{t+1}$, is the reward for taking an action in the current state, s, leading to the next state, $S_{t+1}$. The optimal value of any state is calculated using the Bellman optimality equation as shown in Equation 8 where $V^{*}(s)$, is the optimal value of the state; $r(s,a)$, is the reward obtained for taking action, $a$, in a state, $s$, according to the optimal policy $\pi^{*}$; $V^{*}(s')$ is the optimal value of the next state $s'$ and $\gamma$, is the discount factor [28]. Figure 13 shows an example from literature indicating the general value of the states before (left) and after (right) the Bellman equations are used. The state-value, *V*, represents the quality of the state in terms of how close it is to reaching the goal (trophy) and avoiding failure (fire) within an environment. This helps the agent choose an optimal path that maximize cumulative rewards.

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma V^{\pi}(S_{t+1}) | S_t = s] \quad (7)$$

$$V^{*}(s) = \max\{r(s,a) + \gamma\{V^{*}(s')\}\} \quad (8)$$

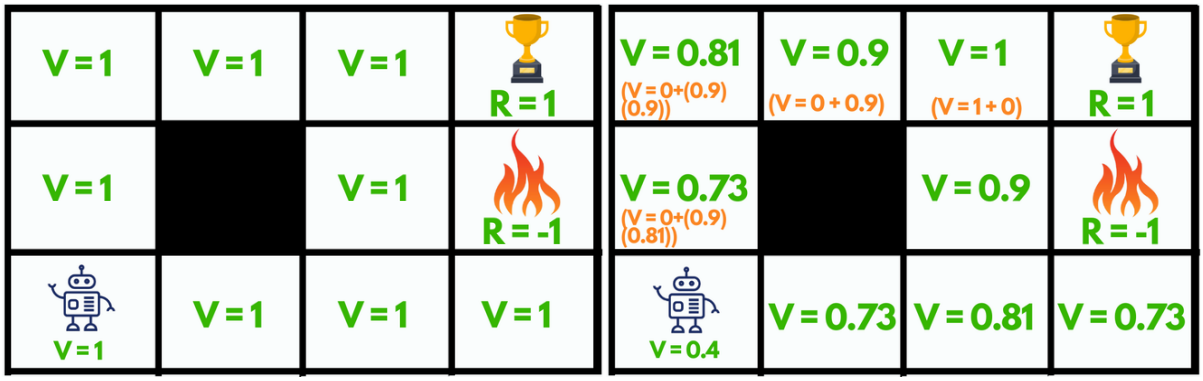



Figure 13: State values before (left) and after (right) using the Bellman equation [33].

Q-learning aims to approximate the optimal action-value function represented by *Q(s,a)* where Q-values represent the quality of a specific action, *a*, in a given state, *s*. [34]. A Q-value exists for each action in a given state showing the importance of different actions. The Q-values are updated using the update equation as shown in Equation 9 where $Q(s_t, a_t)$ and $r_t$ are the Q-value of the state action pair and the reward at a particular timestep, *t*, respectively. The discount factor, $\gamma$, and the learning rate parameter, $\alpha$, are also included. In tabular methods such as Q-learning, $\gamma$, is used to update the Q-values based on the discounted value of the highest future Q-value. The maximum expected future reward, $max_b Q(s, b)$, is the largest Q-value for all possible actions, *b*, in the next state, $s'$ [35]. The Q-values are stored in a Q-table and are optimized to extract the optimal policy by selecting the action with the highest Q-value in each state. The learning takes place off-policy as the NN learns using data obtained in the current or previous policy. Q-learning is usually more sample efficient (faster learning and improved convergence) but tend to be less stable compared to policy optimization methods (discussed in Section 2.5.2) due to the variance in data between past and present policies [34]. Deep Q-Network (DQN) is the most common Q-learning algorithm which combines Q-learning with neural networks to handle complex relationships in the data [34]. The inclusion of the neural network allows for continuous and larger state spaces to be modelled more effectively. In neural network methods such as DQN, the target Q-value function includes $\gamma$, to discount the future predicted rewards which guides the process of minimizing the loss function to optimize the RL agent to learn to take the best action in any possible state and follow the optimal policy.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot max_b Q(s', b) - Q(s_t, a_t)] \quad (9)$$

### 2.5.2 Policy learning

Policy learning directly learns a policy by mapping states to actions to maximize a long-term reward [34]. In policy methods, a state is used as an input to directly predict the best action whereas in Q-learning, the input state predicts a distribution of Q-values and the action with the largest Q-value is

selected. A stochastic policy maps states to probability distributions over actions whereas a deterministic policy maps states directly to particular actions [28]. Policy optimization is a subset of policy learning that uses an iterative process by altering the parameters within a policy to maximize the expected reward thereby improving an existing policy [34]. Policy optimization methods directly optimize an objective function by using gradient ascent/descent to update the weights and biases of the NN to learn an optimal policy without first learning the optimal action-value function [34]. Gradient ascent is used to maximize the expected return whereas gradient descent minimizes the loss function. The learning takes place on-policy as the NN learns using data only from the current policy. Policy optimization provides better stability and convergence as they directly optimize a policy but are less sample efficient as less exploration of the environment occurs compared to Q-learning [36] [37]. Proximal Policy Optimization (PPO) is the most common policy optimization algorithm. Deep Deterministic Policy Optimization (DDPG) is a popular algorithm that combines both Q-learning and policy optimization.

Policy optimization methods usually perform better with continuous action spaces, highly stochastic environments and situations where fine control of the policy is required making it a better option for this application [38]. However, it is also useful to test DDPG to determine whether elements of Q-learning provide good performance in this application. Thus, PPO and DDPG algorithms will be researched further and tested in this application.

### 2.5.3 Proximal Policy Optimization (PPO)

PPO is designed for optimizing policy functions and was introduced by Schulman et al. (2017), from OpenAI [39]. PPO has been shown to outperform the DQN algorithm of Mnih et al. (2015) in numerous cases by making more efficient samples and being less computationally intensive [40]. The PPO algorithm consists of an Actor NN which is a policy method and a Critic NN which is a value method making it a combination of policy and value learning [39]. The Actor NN determines what action to take in a particular state. For a continuous action space, it uses an input state and outputs a standard deviation, $\sigma$, and a mean, $\mu$, for every potential action. A normal distribution is then used to select an action. The Actor NN begins with higher exploration when the mean is unknown and the standard deviation is large to understand different actions and find which ones provide higher rewards in particular states. Exploration gradually reduces as the NN better predicts the standard deviation and the mean. The Critic NN takes an input state and outputs a state value. Each NN is updated during training using an objective function. The Log-Likelihood function is used for PPO which determines the probability of taking a specific action, $a$, in a given state, $s$. This

is represented in Equation 10 where $ln$, is the natural logarithm; $ln(a|s)$, is the Log-Likelihood function; $\mu$, is the mean and $\sigma$, is the standard deviation of the Actor NN [39] [41].

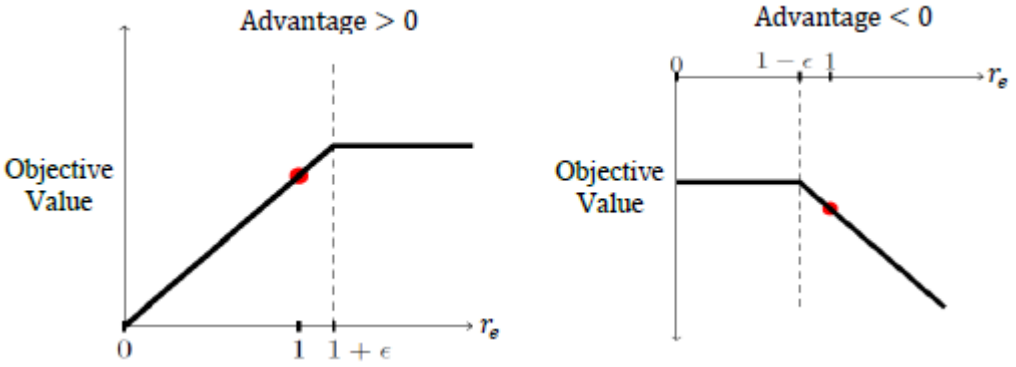
$$\ln(a|s) = -\frac{1}{2}\left(\frac{(a-\mu)^2}{\sigma^2}\right) - \frac{1}{2}\ln(2\pi\sigma^2) \quad (10)$$

Since PPO is updated on-policy, the NN only trains on the current episode's data which can make it challenging for the NN to generalize to all states. A very large learning rate can cause the NN to overfit while a very small learning rate can cause the NN to not capture relevant trends. The motivation for PPO is the desire to prevent large policy updates (weights and biases of NN) during training that can lead to instability by introducing a "proximal" term that limits the size of policy updates while allowing an episode's experiences to be trained upon multiple times [39]. Equation 11 shows the adapted objective function where $ln_{new}(a|s)$ and $ln_{old}(a|s)$ is the Log-Likelihood of selecting action, *a*, in state, *s*, in the current and previous policy respectively [39]. The exponential of the ratio of the new to the old Log-Likelihood is represented by $r_e$. The advantage, *A*, is the difference between the actual and predicted value of the state-action pair $(s, a)$ as shown in Equation 12 where $Q(s, a)$ is the estimated expected return for following the policy after selecting action, a, in state, s, and $V(s)$ is the estimated expected return from state, s, while following the current policy [28]. The advantage, A, indicates how much better it is to take a specific action, a, in a state, s, over randomly selecting an action according to the policy, $\pi(\cdot\,|s)$. The expectation, *E*, over the trajectories of state-action pairs is used as the objective function is determined after multiple iterations [41].

$$Objective\ function = E[\exp(\frac{ln_{new}(a|s)}{ln_{old}(a|s)})A(s,a)] = E[r_e A(s,a)] \quad (11)$$

$$A(s,a) = Q(s,a) - V(s) \quad (12)$$

The *A* and $r_e$ values determine the allowable change of the weights and biases of the Actor NN during training as shown in Figure 14 [39]. A positive advantage (actual action value is better than predicted for the state-action pair) in Equation 12 causes the value of the objective function of the Actor NN to be clipped to a maximum when the value of $r_e$ is above $1 + \varepsilon$, where $\varepsilon$ represents the small allowable change of the Actor NN during each update. However, a negative advantage causes the objective value to be clipped to a maximum when the value of $r_e$ is less than $1 - \varepsilon$. The final objective function is summarized in Equation 13 [41].




(a) (b)

Figure 14: Clipping of the objective function for a positive (a) and negative (b) advantage [39].

$$Objective\ function = E[\min\{r_e A(s,a), clip(r_e, 1-\ \varepsilon, 1+\ \varepsilon)A(s,a)\}] \quad (13)$$

Gradient ascent is used to maximize the objective function and update the weights and biases of the Actor NN appropriately [39]. The Actor NN is updated using the Critic NN hence, the Critic NN also has to be updated. The mean-squared error (MSE) is the objective function that is used to update the Critic NN. This is shown in Equation 14 where $V'(s)$, is the actual state value calculated using rewards from the current episode's experiences and $V(s)$, is the predicted state value from the Critic NN [39]. Gradient descent is used to minimize this error with the weights and biases of the Critic NN being updated appropriately [41].

$$error = [V'(s) - V(s)]^2 \quad (14)$$

The most important hyperparameters of the PPO algorithm are shown in Table 1:

Table 1: Hyperparameters of the PPO algorithm [42].

| **Hyperparameter** | **Description** |
|---|---|
| Entropy coefficient | This balances the trade-off between exploration and exploitation with a higher value encouraging exploration. If the value is too high, the policy can become random and if the value is too low, the agent can miss out on finding better policies. |
| Learning rate ($\alpha$) | This determines the step size at which the model parameters are updated with each iteration to minimize the loss function. If the value is too high, the algorithm can diverge and if the value is too low, little to no learning can take place. |

| | |
|---|---|
| Clip range | This determines the maximum allowable change during each policy update. A larger value can speed up learning but introduce instability while a lower value can improve stability but slow down learning. |
| Number of steps per update (Horizon) | This determines the number of steps taken in the environment before a policy update occurs. A larger value can provide more accurate estimates but slow down learning while a smaller number can provide more frequent updates but lead to a larger variance. |
| Batch size | This determines the number of samples used in the optimization process with each iteration with a larger value providing more stability but requires more computation. |
| Number of epochs | This indicates the number of times the policy will be optimized over the collected data. A high value can result in overfitting of the training data. |
| Discount factor ($\gamma$) | This indicates the importance of future rewards with a higher value encouraging long-term rewards. |

### 2.5.4 Deep Deterministic Policy Gradient (DDPG)

DDPG is designed to handle continuous action spaces by combining Q-learning and policy optimization methods to learn deterministic policies. It also uses an Actor and Critic NN but returns different outputs compared to the PPO algorithm. The Actor NN uses an input state and returns a continuous value for each possible action [43]. This differs from the Actor NN in PPO where the mean and standard deviation is output allowing for exploration in the initial training stages. The Ornstein-Uhlenbeck process generates temporally correlated exploration noise that is less volatile by using a continuous-time process that tends to return towards a mean over time making it useful for environments where consistency of actions is required. Noise either through the Ornstein-Uhlenbeck process or a sample from a normal distribution is added to the DDPG algorithm to account for exploration in the initial training stages which gradually decreases as the agent learns the optimal policy [44]. It is preferred to use noise that is sampled from a normal distribution since it is simpler to execute while obtaining similar results [45]. A state-action pair is used as input to the Critic NN with a state-action value (Q-value) being returned [43]. PPO is an on-policy algorithm whereas DDPG is an off-policy algorithm that utilizes a replay buffer to store past experiences so that past and present policies can be used during training. This stabilizes model training allowing the NN to learn from the same experience multiple times [41] [43].

The objective function represented by the mean-squared error between the predicted and the target state-action value is used to update the Critic NN. The target state-action value is predicted using a target NN with the weights and biases of this NN staying relatively constant during training for better stability. The same NN cannot be utilized to estimate the target state-action value as it would result in the NN attempting to chase a moving target since the target state-action value keeps varying causing training to be unstable. A target NN is also used for the Actor NN hence, the DDPG algorithm involves training four NN's. The target Actor NN uses the next state, $s'$, as input to return the next action, $a'$ during training. The next state and action from the target Actor NN are used as input to the target Critic NN which returns the target state-action value for the next state and action, $\hat{Q}(s', a')$. This process is shown in Figure 15. The target state-action value is discounted (using the discount factor, $\gamma$) to the current state and included in the reward, $r(s, a)$, obtained from selecting action, $a$, in state, $s$. The predicted state-action value, $Q(s, a)$, is obtained from the trained Critic NN. The objective function (error) is shown in Equation 15 [41] [46].

$$error = [r(s, a) + \gamma \hat{Q}(s', a') - Q(s, a)] \tag{15}$$

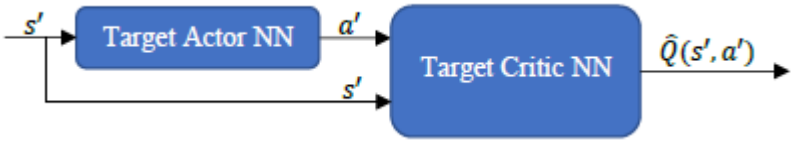



Figure 15: Process of determining the target state-action value for the next state-action pair.

The error is minimized using gradient descent with the weights and biases of the Critic NN updated appropriately. The Actor NN is updated using an objective function represented by the state-action function. The states from a batch are passed through the training Actor NN to calculate the output of the objective function. The actions that would have been taken by the Actor NN in those states are then given to the Critic NN along with the states. For each state-action pair, $(s, a)$, in a batch, the Critic NN returns a state-action value, $Q(s, a)$, as shown in Equation 16 and in Figure 16. The expectation, E, is used as the average of the state-action values is calculated after multiple iterations of a batch [41] [46].

$$objective\ function = E[Q(s, a)] \tag{16}$$

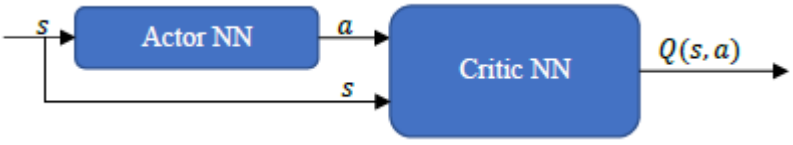



Figure 16: Process of determining the objective function of the Actor NN.

Gradient ascent is used to maximize the objective function with the weights and biases of the Actor NN updated appropriately. After the Actor and Critic NN's are updated, the weights and biases of the target Actor and Critic NN's are updated gradually using Equation 17 where $\theta^*$ and $\theta$ are the weights and biases of the old and new target NN's respectively. The update constant is represented by $\tau$ and $\theta'$ represents the weights and biases of the training NN [41] [46].

$$\theta = \tau(\theta') + (1 - \tau)\theta^* \tag{17}$$

The most important hyperparameters of the DDPG algorithm are shown in Table 2.

Table 2: Hyperparameter of the DDPG algorithm [47].

| **Hyperparameter** | **Description** |
|---|---|
| Learning rate | This determines the step size at which the model parameters are updated with each iteration to minimize the loss function. If the value is too high, the algorithm can diverge and if the value is too low, little to no learning can take place. |
| Buffer size | This indicates the size of the experience replay buffer which stores previous training experiences of the networks. |
| Batch size | This indicated the amount of experiences extracted from the replay buffer for every training iteration. |
| Soft update rate ($\tau$) | This indicates the rate at which the target networks are updated which helps stabilize the training process. |
| Discount factor ($\gamma$) | This indicates the importance of future rewards with a higher value encouraging long-term rewards. |

A summary of the differences between PPO and DDPG are shown in Table 3.

Table 3: Differences between the PPO and DDPG algorithms.

| **Criteria** | **PPO** | **DDPG** |
|---|---|---|
| Stability | Usually more stable due to the constraint placed on the policy update preventing large policy changes [48]. | Can have instability and convergence problems due to hyperparameter sensitivity and correlation in experience replay [48]. |
| Sample efficiency | Usually sample efficient since it uses the current policy's data more effectively [49]. | It has the advantage of reusing past experiences (off-policy) but with the potential for overestimating the Q- |

| | | |
|---|---|---|
| | | values. This bias may result in more samples being required to learn an optimal policy [49]. |
| Continuous action spaces | Less suitable for high-dimensional action spaces due to discretization issues. A surrogate objective encouraging small policy updates is used making it challenging to handle the infinite nature of real numbers. It uses a stochastic policy with inherent exploration capabilities [49]. | Specifically designed for continuous action spaces using a deterministic policy which allows the agent to select precise actions. However, this may not be suitable in certain stochastic environments [49]. |
| Exploration-exploitation balance | Limited exploration in complex environments since the current policy distribution is only used [49]. | The use of noise addition has the potential to explore the action space more effectively [49]. |
| Model complexity | Usually more difficult to implement with more complex hyperparameters [50]. | Simpler architecture with less tuning of hyperparameters required [50]. |

### 2.6 Reinforcement learning evaluation

Unlike supervised learning, RL typically does not use a separate test set to evaluate its performance but rather it is tested by how well the model is able to generalize to new states within an environment [51]. RL uses sequential actions where each action influences the next state of the environment using rewards as feedback rather than labelled data hence, a static test set may not capture changing trends. It is rather evaluated based on its ability to learn good policies, convergent behaviour and sample efficiency. Metrics relating to entropy, episode reward, policy standard deviation, learning curves, and the exploration-exploitation trade-off helps quantify the training performance using multiple episodes. An effective model will lead towards a stable learning curve, increasing episode rewards (improving net profit and fill rate), gradually decreasing entropy and standard deviation and transitioning from exploration to exploitation. This is further explained in the next section. Hyperparameters also control the learning process of RL models and are tuned to optimize the performance of the model. RL models are compared to a baseline approach to determine its relative

performance. In the field of inventory management, net profit and fill rate are primarily used to quantify the performance of the RL model.

### 2.7 Python Software

Python is an open source programming language that consists of a simple structure and syntax making it easy to use. It has an interpreter system providing quick prototyping and assists with debugging operations [52]. Python is the most popular programming language used for web/software development, data science and ML. It consists of many libraries which are collections of pre-written code that can be used to programme more efficiently in specific applications as required. These libraries are able to provide good functionality and flexibility to analyse various forms of data and process complex operations [53]. The following libraries are useful to introduce for this research:

- The NumPy library consists of in-built mathematical functions that can support multi-dimensional data and eases mathematical computations [54].
- The Pandas library consists of flexible data structures that can make data analysis and data conversion easier [54] which is helpful when reading large and complex supply chain data.
- The Matplotlib and Tensorboard libraries are useful for plotting numerical data in different forms [54] allowing for supply chain trends to be visualized.
- The Scikit-learn library consists of a wide range of algorithms and is useful for data processing, model selection and evaluation. It can be used to import the supervised learning algorithms and evaluation metrics to compare the performance of different regression models.
- PyTorch is a deep learning framework that is used for developing and training neural networks with better flexibility, graph analysis and ease of use.

An agent and an environment need to be created to perform the RL process. OpenAI Gym and Stable Baselines are useful Python libraries used to create the RL environment and obtain the agent (algorithm) respectively.

#### 2.7.1 OpenAI Gym

OpenAI Gym provides the framework that allows RL environments to be developed, altered and compared to one another. The library provides a simulation environment in which an agent can take relevant actions and receive observations and rewards as a result of taking a specific action [55]. This library also provides a structural layout framework that can be used to create and test a unique custom environment as shown in Figure 17. For every step (action) taken by the agent, the environment returns the following [55]:

- *observation*, representing the state of the environment.
- *reward*, representing the score received from taking the previous action.
- *done*, representing whether a terminal state has been reached in which case the environment is reset.
- *info*, representing useful information for debugging purposes.

```
import gym
from gym import spaces

class CustomEnv(gym.Env):
  """Custom Environment that follows gym interface"""
  metadata = {'render.modes': ['human']}

  def __init__(self, arg1, arg2, ...):
    super(CustomEnv, self).__init__()
    # Define action and observation space
    # They must be gym.spaces objects
    # Example when using discrete actions:
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)
    # Example for using image as input:
    self.observation_space = spaces.Box(low=0, high=255, shape=
                    (HEIGHT, WIDTH, N_CHANNELS), dtype=np.uint8)

  def step(self, action):
    # Execute one time step within the environment
    ...
  def reset(self):
    # Reset the state of the environment to an initial state
    ...
  def render(self, mode='human', close=False):
    # Render the environment to the screen
    ...
```

Figure 17: Layout of a RL environment using the Open AI gym package [56].

A custom environment can be developed by implementing unique code within each function to simulate an inventory management environment and test various actions, state and reward functions.

### 2.7.2 Stable-Baselines3 (SB3)

SB3 consists of a set of reliable and improved implementations of RL algorithms in *Pytorch* such as more efficient computation, consistent results, ability to monitor the training process, integration with other libraries, etc [55]. It can be used as a framework for which new ideas and refinements can easily be integrated. SB3 is used by importing different algorithms (such as PPO and DQN) and applying them to a RL environment to create a model which can learn an optimal policy. An example of this general process in python code is shown in Figure 18. SB3 utilizes the Multi-Layer Perceptron (MLP) policy which is a feedforward NN that handles all the data useful for training [57] and is

based on the Actor-Critic method where the Actor decides what action to take and the Critic informs how good the action was with temporal difference (TD) being used to provide feedback from the environment to improve the learning process [58].

```
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3 import A2C,PPO,DQN

env = gym.make('CustomEnv', ...)
env_maker = lambda: gym.make('CustomEnv', ...)
env = make_vec_env(lambda: env_maker())

model = PPO('MlpPolicy',env,verbose=1)
model.learn(total_timesteps=100000)
```

Figure 18: Testing of an algorithm in a RL environment using the SB3 package [56].

The SB3 library produces useful information in each training cycle to update the performance of the model including [57]:

- The *approx_kl* parameter indicates the approximate *Kullback-Leibler* (KL) divergence between old and new policy distributions. It is used to prevent large policy updates between old and new polices to maintain stability during training.
- The *clip fraction* parameter indicates the fraction of policy updates that are clipped (when the policy update is restricted within a certain range) during the training process.
- The *clip range* parameter indicates the maximum allowable change during each policy update.
- The *entropy loss* parameter indicates the negative entropy of the policy distribution and is used to balance exploration and exploitation with a higher value encouraging exploration which should gradually decrease as the model trains and learns a deterministic policy.
- The *explained variance* parameter indicates how well the value function (Critic NN) predicts the actual rewards in the environment with a higher value indicating a better fit of the value function to the data allowing for a better prediction.
- The *learning rate* parameter indicates the amount and frequency at which the policy and value functions are updated and controls the rate of learning and convergence.
- The *loss* parameter indicates the overall loss obtained during the policy (policy gradient loss) and value (value loss) function updates.
- The *n_updates* parameter indicates the number of updates in the policy and value functions during the training process.
- The *std* (standard deviation) parameter indicates the spread/variability of the returns obtained in different episodes during training. A lower value indicates better stability but a higher value can also be linked to the agent exploring different strategies.

- The *policy_gradient_loss* parameter indicates the difference between the policy followed by the agent with a more optimal policy that leads to higher expected returns with a lower value driving the policy to improved performance.
- The *value_loss* parameter indicates the difference between the actual and predicted value of the state with a lower value representing a better value function approximation.
- The *actor loss* parameter indicates the negative expected return for the current state-action pair with the goal of maximizing the expected return so that actions are selected that provide higher rewards.
- The *critic loss* parameter indicates the MSE between the predicted and target Q-values with a lower value representing more accurate estimates of the Q-values.

## 2.8 Prior works

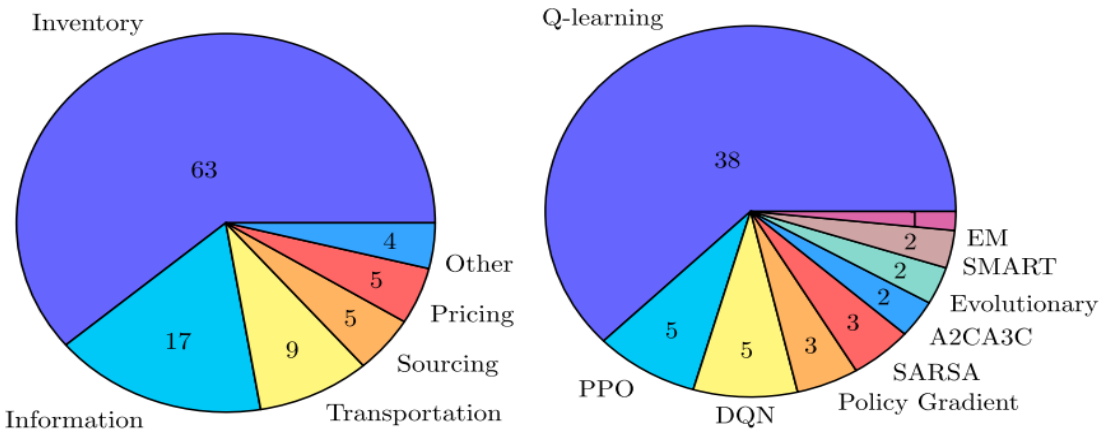
### 2.8.1 Traditional inventory control methods

Just-in-time (JIT) management orders and receives goods as needed rather than ordering too much at a single point in time [59]. It has the potential to reduce inventory costs and reduce wastage but is risky as a sudden spike in demand or small delays can reduce customer service level. The reorder point (ROP) inventory management style is based on placing an order when a certain minimum unit quantity is reached in a company [59]. It allows for products to usually arrive before stockouts occur but does not optimize the inventory turnover or the order quantity which can reduce the profitability and cash flow of the company. Days Sales of Inventory (DSI) is a financial ratio indicating the average days of inventory on hand [59]. It is able to visualize the speed of inventory movement and inventory turnover efficiency but treats all products equally irrespective of value and demand and does not consider supply chain factors such as lead time or production constraints. The Economic Order Quantity (EOQ) model is a traditional inventory method that assumes constant demand with complete information being available about the supply chain to provide rapid solutions [60]. However, real supply chains usually consist of fluctuating demands, uncertain lead times and only partial information being available about the supply chain causing demand and performance cycle uncertainty (time to replenish inventory with customers) making traditional inventory management styles less realistic and practical. RL has the potential to learn adaptive ordering strategies based on the current inventory position of a supply chain providing a more realistic solution.

### 2.8.2 Machine learning applications in supply chains

Figure 19a shows that 63% of the RL models focus on inventory management by training an agent to optimally control the ordering, storing and flow of inventory at various stages in the supply chain

[60]. This indicates that inventory management is complex and still a growing field with significant potential to further improve supply chain management. Figure 19b shows that the Q-learning algorithm is the most common since a model of the specific environment is not required (model free) allowing RL agents to quickly sample and learn using the expected rewards for various state and action combinations [60]. In Q-learning, Q-values converge to optimal values independent of how the RL agent performs while data is obtained. This exploration insensitive property of Q-learning makes it very popular [61]. However, Deep Reinforcement Learning (DRL) algorithms such as Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C) and Deep Q-Network (DQN) are more recently used which incorporate artificial neural networks allowing RL agents to handle large and unstructured input data. This indicates that DRL agents can be applied to more complex supply chains and provide a more efficient and effective training process.

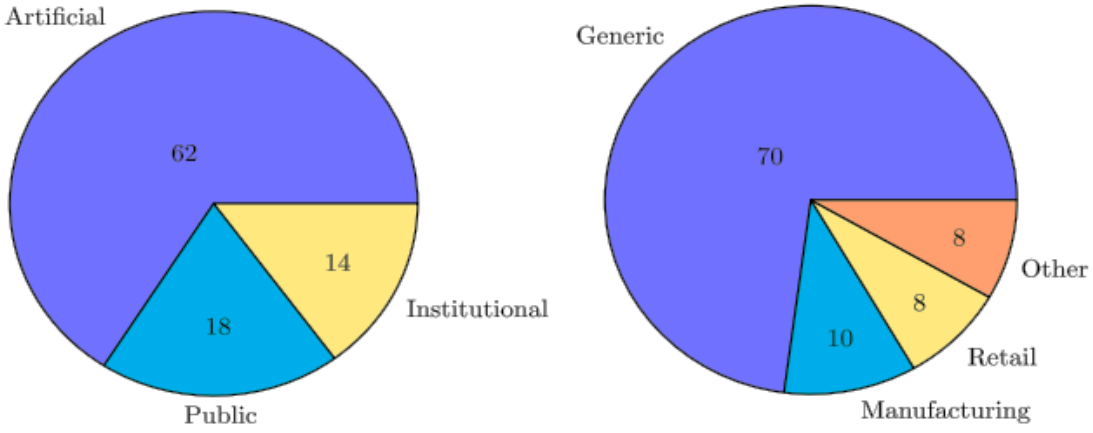



(a) Supply chain drivers (n=103). (b) Algorithms (n=61).

Figure 19: Statistics showing the most common application of RL (a) and algorithm used (b) in supply chains [60].

Useful data of a supply chain that can be used to design the actions, rewards and environment for an RL agent include the supply chain network structure, lead times, forecasted demand, suppliers available, costs and any other constraints. Figure 20a shows that artificial data (fictitious) is most commonly used for the RL model due to a lack of real supply chain data being available [60]. Artificial data contains many assumptions and simplifications making it less realistic. The application of RL using real supply chain data has the opportunity to improve on the conceptual level of design in past research sources to indicate whether the RL model is worth implementing in reality. Industry provides the most realistic data considering various requirements and constraints of a supply chain. Figure 20b shows that generic settings are most commonly used which are simple supply chains consisting of a few systems, little uncertainty and limited constraints reducing the complexity of real-world supply chains [60]. Manufacturing and retail supply chains are specific industrial

sectors that are most commonly considered but very limited research has been performed in this regard.




(a) Data sources (n=94). (b) Industrial sectors (n=96).

Figure 20: Statistics showing the type of data sources (a) and industrial sectors (b) that are most commonly used in literature [60].

### 2.8.3 Q-learning for inventory management

Table 4 presents how the Q-learning algorithm has been applied to the field of inventory management in literature. Three papers considered a linear supply chain network (every stock point has only one predecessor and one successor), three papers considered a divergent network structure (stock points having multiple successors but only one predecessor) and one paper considered a general network structure (stock points having many successors and many predecessors). Five of the seven papers considered a single product in the supply chain and used an infinite time horizon (simulate a very long period to obtain improved parameters for a predefined inventory policy). Most papers used an artificial demand dataset as input for the model with lost sales occurring in the case of unsatisfied demand (insufficient inventory to meet all demand at a specific time). Most papers focus on either minimizing costs or maximizing profits while one paper focuses on reaching the target service level. Q-learning was able to provide an adequate learning process with results comparable to benchmark models and heuristics but they do not converge to an optimal policy all the time. The recent advancements of DRL using neural networks have however, shown to provide improved performance in complex environments with fewer assumptions and simplifications being made.

Table 4: RL applications using Q-learning for inventory management.

| Paper | Network structure | Number of products | Unsatisfied demand | Goal | Horizon |
|---|---|---|---|---|---|
| [62] | Linear | Single | Backorders | Minimize costs | Finite |
| [63] | General | Multiple | Lost sales | Minimize costs | Infinite |

| | | | | | |
|---|---|---|---|---|---|
| [64] | Divergent | Single | Lost sales | Maximize profit | Infinite |
| [65] | Divergent | Single | Lost sales | Maximize profit | Infinite |
| [66] | Divergent | Multiple | Lost sales | Maximize profits | Infinite |
| [67] | Linear | Single | Backorders | Minimize costs | Finite |
| [68] | Linear | Single | - | Target service level | Infinite |

### 2.8.4 Deep reinforcement learning for inventory management

The following literature sources present how DRL has been applied to multi-echelon (optimization of inventory levels across all stock points in a supply chain) supply chains. The state variable was based on the inventory position, the action variable was based on the order quantity and the reward variable was based on profit increase or decrease at each timestep. At each timestep, the produced units are received from upstream, customer orders are received from downstream, current orders are fulfilled from inventory on hand if possible and finally a stock replenishment order is placed upstream. Various inventory costs (holding, ordering) and capacity constraints were considered.

Gijsbrechts et al. [69] used the Asynchronous Advantage Actor Critic (A3C) algorithm, a type of actor critic model, since it was the most popular at that time (July 2019) providing good performance with fast training times. The A3C algorithm showed improved performance of approximately 9% over the base stock policy. However, PPO is a more recently advanced algorithm that provides a more stable and convergent training process. Hubbs et al. [70] used the PPO algorithm with an existing RL environment named *InvManagement-v1* from the OR-GYM library. The RL model showed improved performance by 11% on average over the Derivative Free Optimization (DFO) method (which uses a fixed re-order policy) indicating the potential for RL in improving supply chain management. Xie et al. [71] also used the PPO algorithm to train a RL model and the results were compared to the (*s*, *S*) policy where *s* signifies the reorder point and *S* signifies the order-up-to level. This RL model improved the profit by approximately 18% over the (*s*, *S*) policy across 100 demand datasets. However, all three papers used a normal probability distribution with a certain mean and standard deviation to represent the customer demand and lead time making them unrealistic as they do not consider natural events and external factors. Xie et al. incorporated domain randomization to the demand dataset to experience situations that relate closer to the real world and the RL model was able to better generalize and produce higher profits.

Oroojlooyjadid et al. [72] used the DQN algorithm with a finite time horizon. This model was tested using input demand data from literature sources and two real world demand datasets. On average, the RL agent obtained lower costs in 75% of the cases compared to a fixed base-stock policy with

the best performing model able to reduce costs by approximately 13% compared to a base agent. Geevers et al. [73] conducted research on a particular company that is interested in lowering their inventory costs (holding and backorder costs), while maintaining a fill rate of 98%. Real historic demand data was extracted from the Enterprise Resource Planning (ERP) system of the CardBoard Company (CBC). The PPO algorithm was used with an infinite time horizon and a continuous action space. The results varied significantly (Figure 21) showing that the model is unstable as correct actions were learnt only about 50% of the time. Some simulations performed better than the heuristic benchmark by producing lower holding and backorder costs showing better distribution of its inventory. The best simulation was able to reduce total costs by approximately 20% compared to the benchmark. It was observed that the model was not able to observe the interval of the action set effectively. Both papers applied a multi-agent analysis to optimize the order quantity for each stock point independently considering the capacity constraints at each level.

| Case | DRL | Benchmark |
|---|---|---|
| CBC | 8,402 - 1,252,400 | **10,467** |

Figure 21: Total costs incurred using the DRL method and the benchmark heuristic (lower is better).

### 2.9 Proposed research approach

Considerable research has been performed on inventory management using RL yet, it remains difficult to determine when and how much to order due to the curse of dimensionality (exponential increase in data and computation as the number of variables increase). Most literature sources use many assumptions and simplifications such as artificial demand datasets, constant lead times, limited stock points (simple/linear supply chain network) and do not include their code making them less reproducible. DRL is a relatively new and advanced approach which has not been applied widely, hence the Q-learning algorithm was most frequently used. Heuristics are frequently used to manage and optimize inventory with the base-stock policy being the most common.

This research considers real supply chain data (customer demand and lead time) of a butter company in order to predict the forecasted demand which is used as input to a RL model. The company consists of a multi-echelon supply chain with a general network structure. A single product (butter) was considered in this research with the goal of simultaneously minimizing costs, maximizing profits and the order fill rate. A finite time horizon, continuous action space and a single-agent analysis was used which streamlines the RL model (fewer variables and dependencies) resulting in a more scalable

neural network which means it can efficiently handle an increase in size and complexity of the data. A custom RL environment using unique state, action and reward variables was created together with DRL algorithms (PPO and DDPG) to train an agent more effectively than existing RL environments in an attempt to improve the inventory management of the company.

# 3 Methodology

The following methodology as represented in Figure 22 was used including:

- Supervised learning to perform demand forecasting by learning the trends and features of the historical sales dataset. The demand forecasts provide useful input data to both the heuristic and RL models to help plan inventory ahead of time.
- Incorporating the company heuristic to the supply chain workflow to obtain a baseline inventory management model.
- Developing a custom RL environment with specific algorithms to plan, track and optimize inventory in the supply chain to maximize profits.

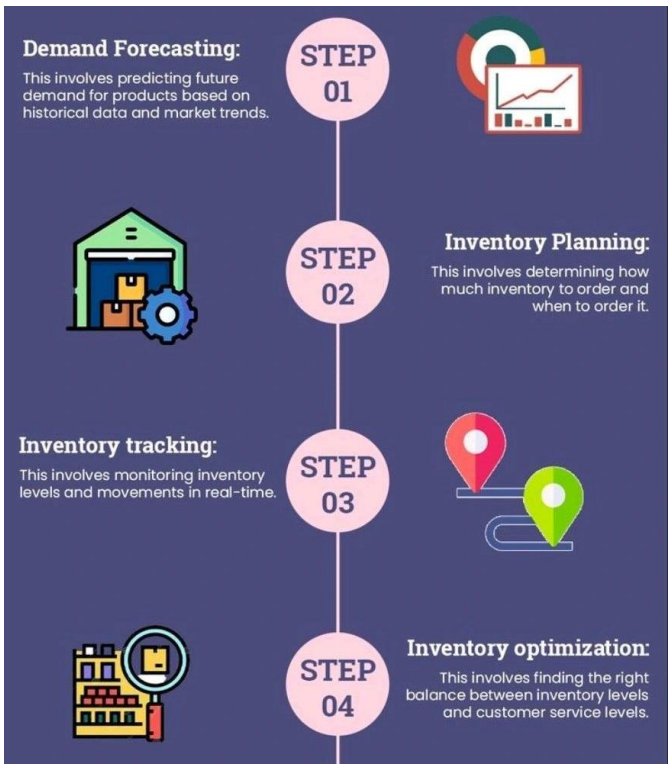



Figure 22: Key activities in inventory management [75].

## 3.1 Hardware and software used in this research

The hardware used was a Hewlett-Packard (HP) laptop with a Windows 10 operating system, 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz processor and a Random-Access Memory (RAM) of 16 GB. PyCharm developed by JetBrains was used which is an integrated development environment in which the Python programming language can be used to develop unique code. It provides a platform for python code to be easily be written, tested and altered. Version 2023.2 (Community Edition) of PyCharm is used incorporating version 3.11 of the Python programming software.

### 3.2 Supervised learning for demand forecasting

Supervised (labelled data) regression (continuous output) ML models were developed to determine the forecasted demand using the following process:

1. The raw time series data were pre-processed so that it is converted into a suitable form that can be understood by ML models.
    - An Excel file consisting of the input data is read into PyCharm using the *pandas* python library which helps with easily importing and managing datasets.
    - The dataset only had a date column and a demand (sales) column making it a univariate time series analysis. The date is used as an index and the sales column as the output label. Input labels were required for the model to learn relationships between the input and output data. The sales for the previous days were used to form the input label at each timestep in order to predict the output label which was the current demand. This type of model is known as auto-regression since a variable's past values are used to predict the future values. Two techniques are applied to determine a suitable length for the input label:
        - Auto correlation is used to identify a relationship of a variable with its previous time period values and is quantified using the Pearson's correlation coefficient which ranges between -1 (strong negative correlation) and 1 (strong positive correlation). The partial auto-correlation function (Figure 23) is used for auto-regression models by measuring only the direct effects of values in previous time lags. The y-axis represents the Pearson's correlation coefficient and the x-axis represents previous time lags [76]. Correlation values in the blue band are statistically insignificant and do not need to be considered. Hence the partial autocorrelation plot suggests that the previous three days is the most relevant and best to use as the input length for making future predictions.

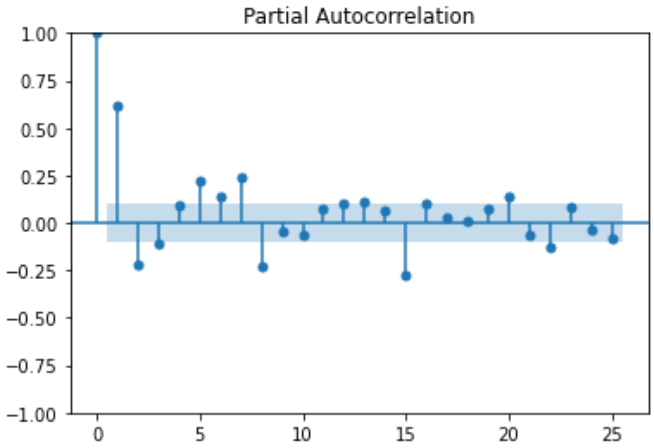



Figure 23: The partial autocorrelation relationship of the dataset.

    - The Auto Regression Integrated Moving Average (ARIMA) model converts non-stationary time series data into stationary time series data [77]. The *auto ARIMA* python function was used which automatically attempts different combinations of orders to minimize an error. Once it converges, it presents the order values of the best model. The order of the AR model indicates how many past values to consider to make a future prediction which again amounted to three for this dataset.
- Since the demand for the previous three days were used as input labels, missing data arises in the first three rows of the dataset. The missing data was handled by completely removing the incomplete rows using the *fillna* python function. The dataset is fairly large and stochastic making the process of deleting the incomplete rows suitable.
- The range of values in the input data are very large which can cause some features to dominate others. Normalization is used to address this issue by transforming features to be on a similar range and scale which helps improve the performance and training of the model. The *preprocessing* python function from the *sklearn* library was used to perform the normalization.
- The entire dataset was split into a training dataset, validation dataset and a test dataset to try make the ML model perform well on both existing and new data reducing the chances of underfitting or overfitting the data. As a common convention, approximately 80% of the data was used for training and the remaining 20% was used for testing.
- A pair plot was created using the *seaborn* python library to explore the distribution and relationships between the input variables in the dataset. It is shown in Figure 24 that there are mostly weak correlations (no fixed trend) between them linked to the highly stochastic nature of the dataset. Additionally, it indicates that none of the features are redundant allowing for a more efficient learning process.

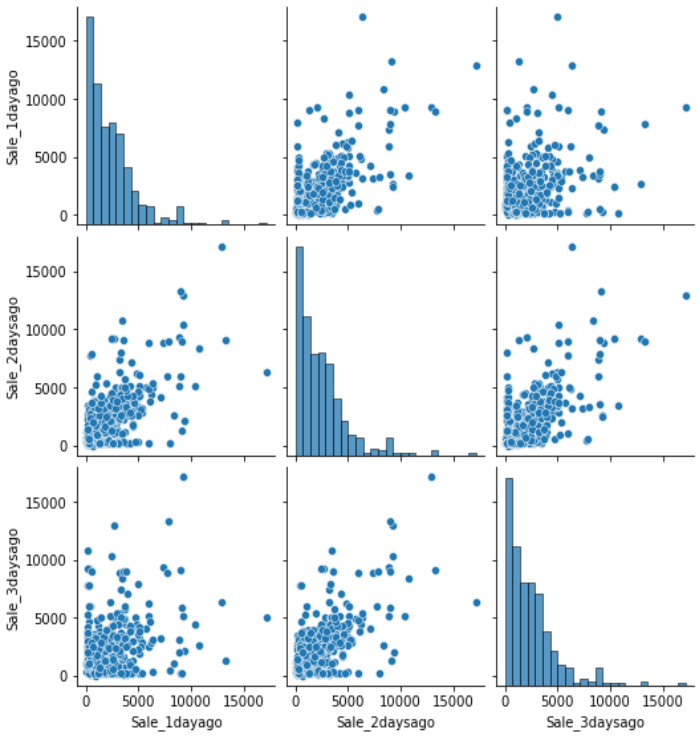



Figure 24: Pair plot showing the relationship between the input variables of the dataset.

2. Supervised learning algorithms are imported from the *sklearn* python library. A separate model was created using each of the algorithms. K-fold cross validation and randomized search cross validation was used to determine the best hyperparameters of each model based on the training dataset.
3. For each algorithm, the best hyperparameters were used to predict the demand based on the input of the test dataset. The MSE, RMSE, MAE and $r^2$ was calculated and compared for each model using the *sklearn* python library.

The demand forecasts are used as input when modelling the supply chain as indicated in the next section.

### 3.3 Modelling the particular supply chain

The general supply chain workflow that was used in the heuristic and RL models is shown in Figure 25. The production facility produces a certain number of units in between the minimum and maximum production limit while considering its storage capacity and the production processing time before they turn into finished inventory. The warehouse orders a certain number of units from the production facility whereas the outlets order a certain number of units from either the warehouse or the production facility. The order quantity considers the units available at the upstream location, storage capacity of the warehouse/outlet, truck capacity and transport time between the respective facilities. The orders that are placed from a stock point to an upstream facility is placed in a backlog

variable as units can only be sent when the upstream facility has sufficient units available. Revenue is gained by fulfilling customer demand at the warehouse and each of the outlets. The product lifespan of available inventory is reduced and any obsolete inventory is discarded. Due to the perishable nature of the product, the First In, First Out (FIFO) inventory method was applied with the aim of reducing the chances for obsolete inventory to occur. The overall cost including manufacture, delivery and storage costs is calculated and finally the overall net profit and fill rate is calculated. Various assumptions about this particular supply chain were made including:

- Unlimited raw material supply
- A fixed sales price
- Single butter product type
- Perpetual inventory control (Fast-Moving Consumer Goods (FMCG) style)
- Backorders not considered
- Quality maintained

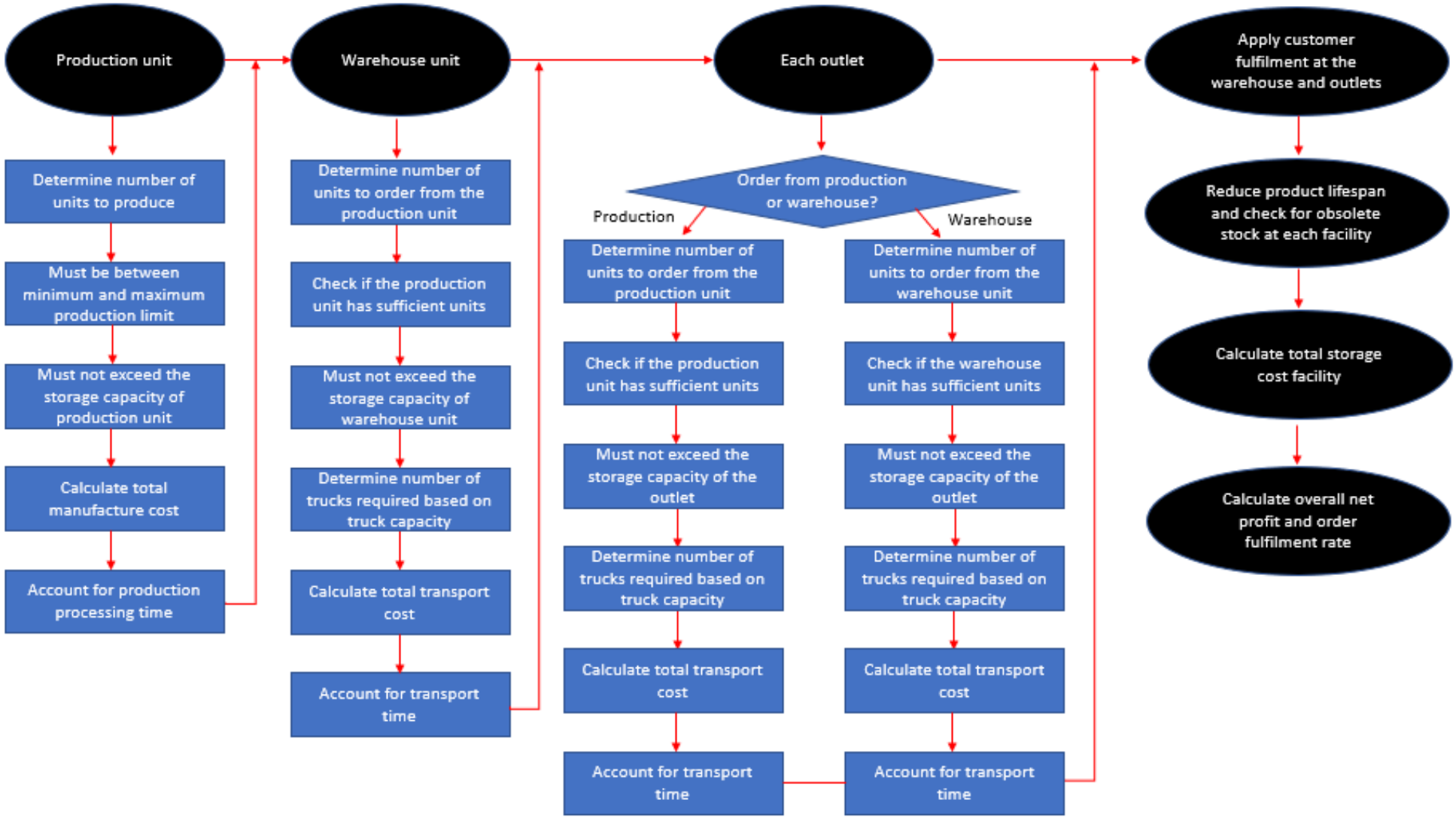



Figure 25: Supply chain workflow.

### 3.4 Benchmark heuristic for inventory management

A heuristic model was developed using the unique supply chain workflow from Figure 25 together with the company heuristic involving storing inventory based on the sum of the demand for next few days at each stock point. For each of the outlets, seven days of their respective forecasted demand was used while the warehouse was based on its twelve-day forecasted demand. This replicates the current approach used in the company. The initial units available at each of the outlets and the warehouse on day 1 to begin the simulation was based on the same idea.

### 3.5 Reinforcement learning for inventory management

The OpenAI Gym structure was used to create a custom RL environment incorporating the supply chain workflow as shown in Figure 25. The same starting conditions, supply chain parameters and operations were used as the heuristic to allow both methods to be comparable. Multiprocessing in SB3 was used to run multiple instances of the environment simultaneously to speed up training and decrease the overall computational time. A random seed was defined to provide reproducibility of results.

#### 3.5.1 Action space

Every time step, an ordering quantity (action) is placed from the warehouse and each of the outlets to an upstream facility in the supply chain whereas the production decides on a certain number of units to produce (action). These actions are based on the learned policy of the RL models. The number of units produced at the production is scaled between the minimum and maximum production limit and also depends on the units available at the production facility and the storage capacity of the production unit as shown in pseudocode (Equation 18).

$$
\begin{aligned}
&Production\ action = (production\ action * (maximum\ production\ limit - \\
&minimum\ production\ limit)) + minimum\ production\ limit \\
&if\ production\ units\ available + production\ action > production\ storage\ capacity: \\
&\quad production\ action = production\ storage\ capacity - production\ units\ available \\
&\quad if\ production\ action < minimum\ production\ limit: \\
&\quad\quad production\ action = 0
\end{aligned}
\tag{18}
$$

A random choice based on the probabilities in Table A.5 was used to determine whether the outlets order from the production or the warehouse. The orders are placed in a backlog and only processed when the upstream facility has sufficient units and the addition of units does not exceed the storage

capacity of the facility. A simple example using the Pretoria outlet in pseudocode is shown in Equation 19 but the warehouse and the remaining outlets follow a similar approach.

$$
\begin{aligned}
&Pretoria\ choice = random\ value\ from\ 0\ to\ 100 \\
&if\ Pretoria\ choice > distribution\ percent\ Production\ for\ Pretoria: \\
&\quad send\ units\ (Pretoria\ action)\ from\ warehouse\ to\ Pretoria \\
&if\ Pretoria\ choice < distribution\ percent\ Production: \\
&\quad\quad send\ units\ (Pretoria\ action)\ from\ production\ to\ Pretoria \\
&Backlog\ +=\ Pretoria\ action \\
&if\ upstream\ facility\ has\ sufficient\ units\ and\ if\ Pretoria\ action \\
&\quad\quad + Pretoria\ units\ available < Pretoria\ storage\ capacity: \\
&\quad Send\ units\ from\ upstream\ facility\ to\ Pretoria
\end{aligned}
\tag{19}
$$

A continuous action space is used that outputs a normalized value between 0 and 1 hence, it needs to be scaled up to represent a realistic ordering quantity. Different methods based on the average demand, storage capacity, truck capacity or unique optimization can be used to determine the scaling factor for the warehouse and each outlet as summarized in Table 5.

Table 5: Methods used to define the action space of the RL models.

| Option | Action space |
|---|---|
| 1 | Average demand of the facility |
| 2 | Storage capacity of the facility |
| 3 | Truck capacity |
| 4 | Unique value for each facility |

### 3.5.2 State space

Table 6 shows relevant supply chain parameters including current inventory levels, demand information, order information, cost information and truck utilization that were included in the state space one at a time to determine the optimal combination that provides the highest net profit for this research.

Table 6: Methods used to define the state space of the RL models.

| Option | State space |
|---|---|
| 1 | Units available at each facility |
| 2 | Choice of ordering from production or warehouse |
| 3 | Number of trucks in operation between the facilities |
| 4 | Units in transit between the facilities |
| 5 | Historic and forecasted demand |
| 6 | Number of units in the backlog |
| 7 | Fill rate |
| 8 | Current profit, revenue gained |
| 9 | Current day |
| 10 | Current units satisfied/unsatisfied |
| 11 | Current storage/manufacture/delivery cost |
| 12 | Units expiring in 1, 2, 3 days |

### 3.5.3 Reward function

The objective of the RL model is to maximize the profits in the supply chain of Company X. Different methods were used to develop and test the reward function including profit per timestep (Equation 20), fill rate (Equation 21), maximizing number of units sold (Equation 22), minimizing unsatisfied demand (Equation 23), minimizing number of units available (Equation 24) and minimizing obsolete inventory (Equation 25) as summarized in Table 7. Reward shaping is applied by including scaling factors to each reward to give priority to certain scenarios. Different combinations of these reward functions with unique reward shaping were tested to find the best reward function that provides the highest net profit for this research.

$$reward \mathrel{+}= current\ revenue - current\ cost \tag{20}$$

$$if\ order\ fulfilment\ rate > 90\%$$
$$reward \mathrel{+}= order\ fulfilment\ rate \tag{21}$$

$$reward \mathrel{+}= current\ units\ satisfied\ at\ each\ facility \tag{22}$$

$$reward \mathrel{-}= current\ units\ unsatisfied\ at\ each\ facility \tag{23}$$

$$reward \mathrel{-}= units\ available\ at\ each\ facility \tag{24}$$

$$reward -= obsolete\ inventory \quad (25)$$

Table 7: Methods used to define the reward function of the RL models.

| Option | Reward function |
|---|---|
| 1 | Profit per timestep |
| 2 | Fill rate |
| 3 | Maximizing number of units sold |
| 4 | Minimizing unsatisfied demand |
| 5 | Minimizing number of units available |
| 6 | Minimizing obsolete inventory |

Extremely large values in the state space and reward function can lead to unstable and unpredictable behaviour causing high variance which makes it difficult for the agent to learn and generalize to new conditions [78]. It can also cause numerical instability in the algorithms potentially leading to large errors. Hence, the variables used in the state space and the reward function are normalized using min-max normalization into a fixed range so that they are well-scaled to provide a balanced signal allowing for a more efficient training process. The hyperparameters of the PPO and DDPG algorithms used in this research were tuned in an attempt to improve the performance of the RL model.

### 3.5.4 Hyperparameters

The most important hyperparameters of the PPO and DDPG algorithms that were presented in Section 2.5.3 and Section 2.5.4 respectively were tuned to improve the performance of the RL model. Table 8 and 9 show the different hyperparameters and their ranges that were tested for the PPO and DDPG models respectively to optimize their performance. The randomized search cross validation technique was used to test different hyperparameter combinations from Table 8 and 9.

Table 8: Different hyperparameters and the ranges in which they were altered to potentially improve the performance for the PPO model.

| Hyperparameter | Range |
|---|---|
| Entropy coefficient | [0.005, 0.01, 0.05, 0.1, 0.3] |
| Learning rate | [0.0001, 0.001, 0.003, 0.005, 0.01, 0.1] |
| Clip range | [0.1, 0.2, 0.3] |
| Number of steps | [32, 128, 256, 512, 2048, 5000] |
| Batch size | [32, 64, 128, 4096] |

| | |
|---|---|
| Number of epochs | [3, 5, 8, 10, 30] |
| Discount factor (gamma) | [0.9, 0.95, 0.99] |

Table 9: Different hyperparameters and the ranges in which they were altered to potentially improve the performance for the DDPG model.

| **Hyperparameter** | **Range** |
|---|---|
| Learning rate | [1e-4, 1e-3, 1e-2] |
| Buffer size | [1e4, 1e5, 1e6] |
| Batch size | [32, 64, 100, 128] |
| Tau | [0.001, 0.005, 0.01] |
| Discount factor (gamma) | [0.9, 0.95, 0.99] |

# 4 Results and Discussion

## 4.1 Results of the demand forecasting process

Figure 26 shows the performance comparison of different supervised learning models on the validation dataset after k-fold cross validation and hyperparameter tuning were performed. The validation dataset used to evaluate the models was from a similar distribution (mean, variance) to the training dataset. It can be seen that most of the models were able to predict a similar trend to the actual values indicating that the models are potentially capturing the essential patterns in the data. The RF model was able to best predict the demand by achieving the closest trend and the least error for most of the evaluation metrics (MSE, RMSE, $r^2$) as shown in Table 10. The MAE is less sensitive to outliers compared to the MSE. The MAE of the RF model was marginally higher than the SVR model which possibly indicates that the SVR model is marginally more conservative with the RF model willing to take higher risks. The exploratory nature of RF could ultimately be a better choice for a stochastic dataset such as the one in this research. However, both the RF and SVR models outperformed the other models as expected based on their abilities of better handling non-linear effects in a dataset. The $r^2$ value was not very high because of the non-linear nature of the dataset but the RF model did show a 79% improvement in explaining the variance over the baseline LR model.

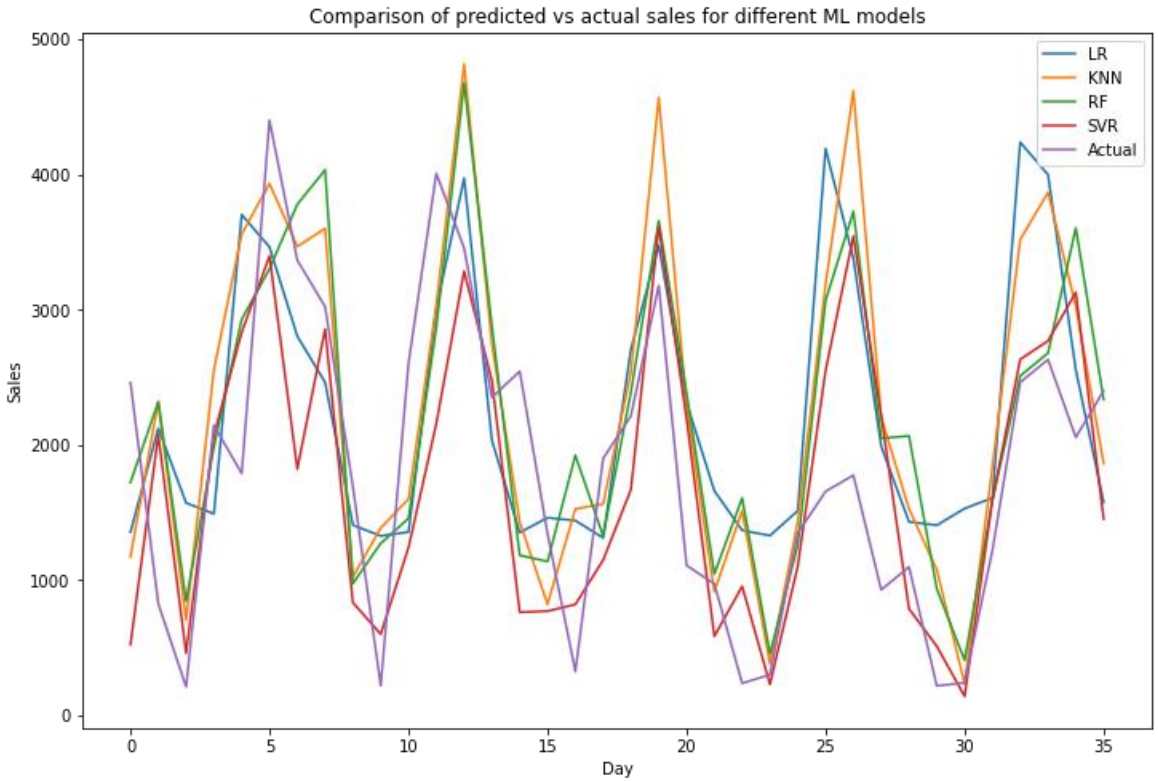



Figure 26: Performance of different ML models on the validation dataset for the Pretoria outlet.

Table 10: Evaluation metrics for the various regression models on the validation dataset for the Pretoria outlet.

| Model | MSE | RMSE | MAE | $r^2$ |
|---|---|---|---|---|
| LR | 1 156 705 | 1076 | 937 | 0.075 |
| KNN | 1 087 652 | 1043 | 864 | 0.130 |
| RF | 810 042 | 900 | 741 | 0.352 |
| SVR | 856 255 | 925 | 735 | 0.315 |

The training and validation datasets were combined for the final training (including k-fold cross validation and hyperparameter tuning) of the RF model and fit to the test dataset with the trend shown in Figure 27 and the evaluation metrics shown in Table 11. Once again, the model was able to follow a similar trend to the actual values but large evaluation metrics (errors) were observed which are due to the highly stochastic nature of the dataset, consideration of a short time period and the inclusion of a sudden fluctuation of high amplitude which could possibly have been a random promotion for the product in the company. Possible reasons for the large difference between the actual and predicted sales on day 17 may include:

- The large fluctuation in demand may not be well-represented in the training data making the model unable to learn such extreme events accurately. This indicates the limitation of attempting to make future predictions based only on the historical time series data. In future, real-time demand data and its relationship with market trends, company status, seasonal patterns, etc can be obtained directly from the ERP system of the company together with relevant websites.
- The predictions of the RF model are bound by the highest and lowest values in the training dataset with the model unable to extrapolate.
- RF tends to smooth out noise in a dataset making it less sensitive to sudden fluctuations.
- The predictions from each tree of the RF algorithm can result in decision boundaries that make it challenging to predict large fluctuations.

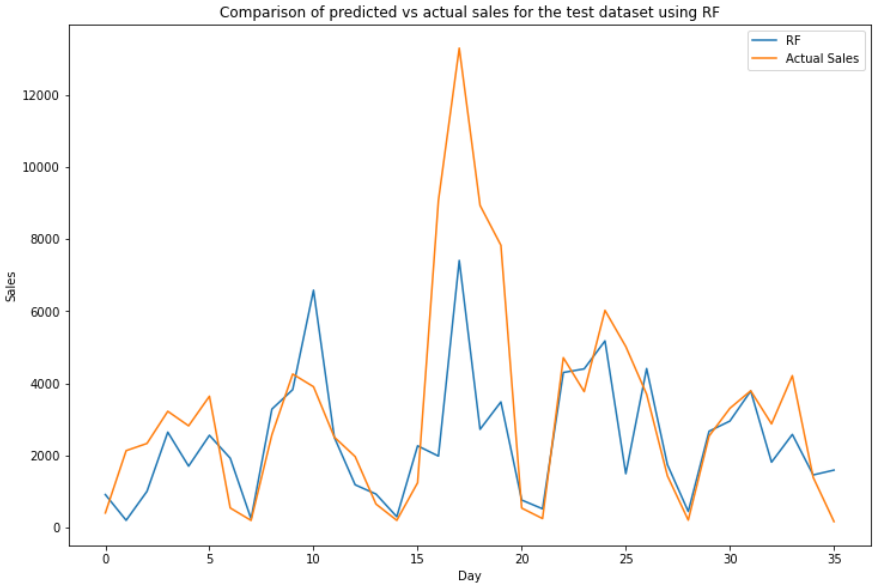



Figure 27: Performance of the RF model on the test dataset for the Pretoria outlet.

Table 11: Evaluation metrics for the RF model on the test dataset for the Pretoria outlet.

| **Model** | **MSE** | **RMSE** | **MAE** | $r^2$ |
|---|---|---|---|---|
| RF | 5 382 070 | 2320 | 1346 | 0.347 |

The RF algorithm was shown to best predict the forecasted demand for a single outlet (Pretoria) and was hence, used to determine the forecasted demand for the remaining facilities for consistency. Additionally, supervised learning was not the main focus in this research hence, all the algorithms were not tested at every location in the supply chain. The comparison between the actual and predicted sales over the period of a year at each location using the RF algorithm is shown in Appendix B.

Recurrent neural networks (RNN) are acknowledged as state-of-the-art for predicting time-series data but due to time constraints, computational costs and the need for further optimization, it was not a feasible option for this study. RNNs also require large datasets for effective training and usually do not perform well with smaller datasets. Since the time series data is limited to one year in this study, the RF algorithm was an effective starting point. RF was also observed to be more interpretable than RNN which can have an effect on the efficiency of the RL model. The main focus for this research was to test and optimize various RL models to possibly improve the inventory management process. However, in the future with more powerful machines and larger input datasets, RNN would definitely be tested as it has the potential to provide more accurate demand forecasts.

### 4.2 Benchmark heuristic results

The value of the minimum production limit used in the company was found to be excessive for the way the supply chain was modelled in this research using the heuristic technique. The original minimum production limit of 30 000 units each day led to a large number of units being available at the production facility reaching the production storage capacity. This increased the storage costs and obsolete inventory which significantly limited the net profit. Table 12 shows the demand information at each location of the supply chain. The sum of the average demand across all the locations was 6 765 units which was much lower than the original minimum production limit resulting in excessive units being available. The high values in the sample standard deviation and maximum demand columns emphasize the high variance and stochastic nature of the demand dataset. A value of 5000 units for the minimum production limit provided the best performance for the heuristic model.

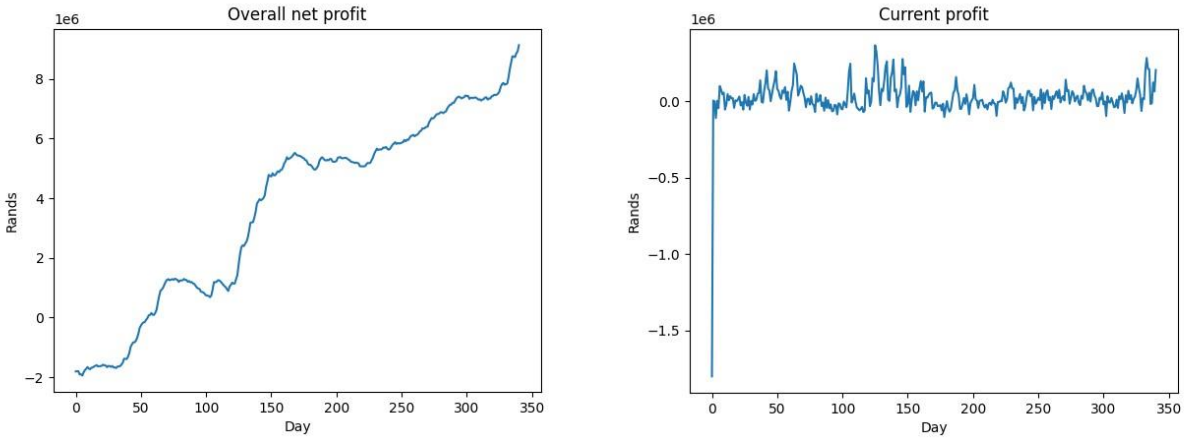
Table 12: Demand information at each location.

| Location | Average demand (units) | Sample standard deviation | Maximum demand (units) |
|---|---|---|---|
| Bloemfontein | 1 077 | 824 | 3 903 |
| Durban | 766 | 297 | 1 437 |
| Cape Town | 112 | 117 | 867 |
| East London | 223 | 187 | 1 318 |
| Pretoria | 2 403 | 2 374 | 17 125 |
| Johannesburg | 2 184 | 2 250 | 13 939 |
| **Total** | **6 765** | | |

### 4.3 RL results

#### 4.3.1 State space

Using cumulative variables in the state space were found to be ineffective as they result in an infinite state space leading to an inefficient learning process. Variables based on the current value for example, current profit as opposed to overall profit as shown in Figure 28 was more practical to use as they are bounded allowing the RL agent to experience similar states more frequently during training. Table 13 shows the variables that were found to be useful to include in the state space. Other variables were also tested as shown in Appendix C but these were found to be ineffective in this particular RL model.




(a) (b)

Figure 28: Comparison of the (a) overall net profit with the (b) current profit at each timestep.

Table 13: Variables included in the state space.

| Variables that provided good performance | Variable benefits |
|---|---|
| Units available at each stock point. | This gave the model an idea of how much demand can be fulfilled from on-hand inventory. |
| The choice of ordering from either the production or the warehouse from each outlet. | This gave the model an idea of which upstream location is better to order from based on units available, lead times and costs. |
| Number of trucks in operation between production, warehouse and each of the outlets. | This gave the model an idea of how consolidated the shipments are. |
| Number of units in transit between production, warehouse and each of the outlets. | This gave the model an idea of how much inventory will arrive after the lead time has elapsed. |
| The demand for the previous 4 days and the forecasted 4 days at each location. | This gave the model an idea of the current trend in demand for each location. |

### 4.3.2 Reward function

It was investigated whether there was a difference between maximizing a positive reward and minimizing a negative reward. Attempts were made to maximize a positive reward by adding the current units satisfied to the reward function but it did not provide any performance improvement. However, subtracting the current units unsatisfied provided improved performance. Hence, it was concluded that it was better to minimize a negative reward in the RL model for this application. A reward function based on a combination of profit per timestep, fill rate, units available on hand and units unsatisfied provided the highest net profit at the end of the year. It was also found that using a

constant reward value for a particular range of states was not always effective as static conditions can become highly non-linear. Thus, unique scaling factors (reward shaping) were used in the reward function to indicate certain scenarios were better than others which proved to be effective in improving the net profit of the RL model. However, the reward function was very sensitive and had a significant effect on the performance of the RL models. The final reward function is shown in pseudocode (Equation 26).

$$reward \mathrel{+}= current\ revenue - current\ cost$$

$$\begin{aligned}
&if\ fill\ rate > 90:\\
&\quad reward \mathrel{+}= fill\ rate\\
&else\ if\ fill\ rate > 80:\\
&\quad reward \mathrel{+}= 50\\
&else\ if\ fill\ rate > 70:\\
&\quad reward \mathrel{+}= 0\\
&else\ if\ fill\ rate > 60:\\
&\quad reward \mathrel{-}= 10\\
&else\ if\ fill\ rate > 50:\\
&\quad reward \mathrel{-}= 20\\
&else\ if\ fill\ rate > 40:\\
&\quad reward \mathrel{-}= 30\\
&else\ if\ fill\ rate > 30:\\
&\quad reward \mathrel{-}= 40\\
&else\ if\ fill\ rate > 20:\\
&\quad reward \mathrel{-}= 50\\
&else\ if\ fill\ rate > 0:\\
&\quad reward \mathrel{-}= 100
\end{aligned}$$

$$\begin{aligned}
&reward \mathrel{-}= Bloemfontein\ units\ available\\
&reward \mathrel{-}= Current\ Bloemfontein\ units\ unsatisfied\\
&reward \mathrel{-}= Cape\ Town\ units\ available * 2\\
&reward \mathrel{-}= Current\ Cape\ Town\ units\ unsatisfied\\
&reward \mathrel{-}= Durban\ units\ available\\
&reward \mathrel{-}= Current\ Durban\ units\ unsatisfied * 10\\
&reward \mathrel{-}= East\ London\ units\ available
\end{aligned}$$

$$reward -= Current\ East\ London\ units\ unsatisfied$$
$$reward -= Pretoria\ units\ available$$
$$reward -= Current\ Pretoria\ units\ unsatisfied$$
$$reward -= Warehouse\ units\ available$$
$$reward -= Current\ Warehouse\ units\ unsatisfied$$
$$reward -= Production\ units\ available/1000 \quad (26)$$

### 4.3.3 Action space

Scaling up of the action space was tested using the value of the storage/truck capacity and a factor of the average demand at each location. Using a factor of the storage/truck capacity provided a higher fill rate but achieved a lower net profit when compared to using a factor of the average demand. The RL code was highly sensitive to the upscaling of the action space due to the current profit being included in the reward function. The RL agent learns to send consolidated trucks most of the time to save on costs as delivery costs usually exceed storage costs on a single day. A challenge was that large amounts of time are required before upstream locations store sufficient goods in order to send full trucks thereby, reducing the fill rate. Possible solutions included increasing the starting inventory, increasing the minimum production limit, altering the reward function and using a unique scale up of the action space for each location. The RL model was run multiple times by changing the scaling factor for one facility while keeping the others constant. Once the best scaling factor was found for the first facility, the process was repeated for the remaining facilities. A unique scale up of the action space for each location provided the best performance with the scaling factors shown in Table 14.

Table 14: Optimized action scaling factor results.

| **Facility** | **Action scaling factor for both PPO and DDPG models** |
|---|---|
| Warehouse | 28 000 |
| Bloemfontein | 10 000 |
| Durban | 5 000 |
| East London | 10 000 |
| Cape Town | 1 000 |
| Pretoria | 10 000 |

### 4.3.4 Hyperparameters

Hyperparameter tuning was performed using the random search method where hyperparameters were randomly sampled from a defined distribution. The final hyperparameter values for the PPO and DDPG models are presented in Table 15. The entropy coefficient of PPO and the learning rate and discount factor of both PPO and DDPG were the most sensitive to the model's performance. Since hyperparameter tuning was not the main focus of this research and the results of the optimization were lengthy, the final values are only presented in this research.

Table 15: Optimized hyperparameter results.

| PPO hyperparameter | Value | DDPG hyperparameter | Value |
|---|---|---|---|
| Entropy coefficient | 0.1 | Learning rate | 0.003 |
| Learning rate | 0.005 | Buffer size | 1e6 |
| Clip range | 0.2 | Batch size | 100 |
| Number of steps | 2048 | Tau | 0.005 |
| Batch size | 64 | Discount factor (gamma) | 0.99 |
| Number of epochs | 10 | | |
| Discount factor (gamma) | 0.99 | | |

## 4.4 Comparing the heuristic and RL model results for scenario 1: Supply chain structure in its current state

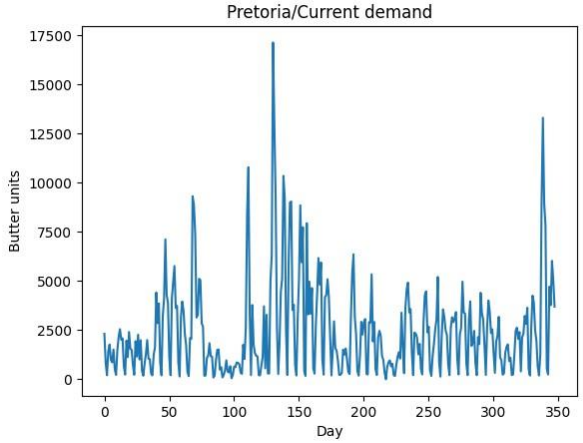
Table 16 shows that there were significantly less units unsatisfied and obsolete inventory occurring resulting in a higher fill rate in the heuristic model compared to the PPO and DDPG models. Both PPO and DDPG models achieved a marginally lower net profit but at a significantly lower fill rate compared to the heuristic model. Further increasing or decreasing the fill rate in the PPO and DDPG models resulted in a decrease in net profit being achieved. This indicates that the optimal condition was reached and that a higher fill rate in a supply chain does not necessarily lead to a higher net profit. The extremely high number of units unsatisfied in the PPO and DDPG models limited their profitability but it was interesting to note that it still achieved a net profit relatively close to the heuristic model. However, the PPO and DDPG models can help optimize the supply chain structure, operation and parameters to potentially further increase the net profit beyond that of the heuristic in the future. The heuristic model improved the net profit by 17% and 8% compared to the PPO and DDPG model respectively. DDPG achieved a higher net profit compared to the PPO model but detailed results regarding each model are presented in the following sections.

Table 16: Comparison of overall performance between different models in scenario 1.

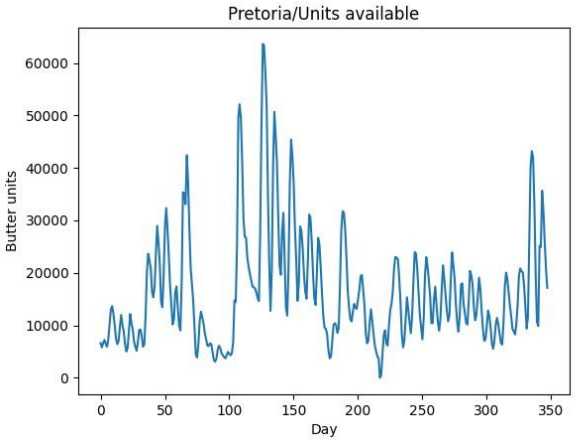
| | **Heuristic** | **PPO** | **DDPG** |
|---|---|---|---|
| Net profit | R11 042 078 | R9 125 155 | R10 149 107 |
| Fill rate | 97.41% | 64.90% | 55.37% |
| Obsolete inventory | 56 945 | 228 858 | 224 000 |
| Units unsatisfied | 62 315 | 811 399 | 1 030 000 |

### 4.4.1 Results showing the comparison of actions used in the heuristic and RL models

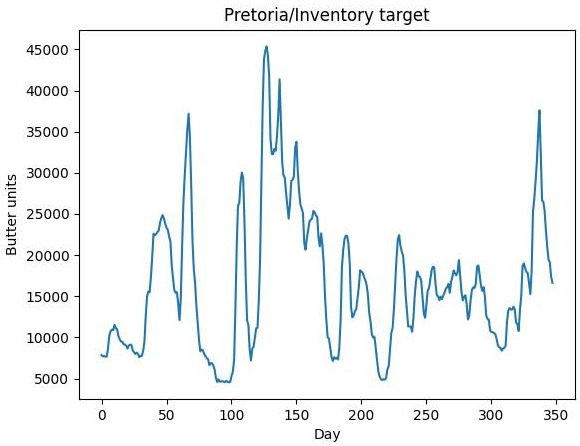
The results for a single outlet (Pretoria) in the heuristic model are presented in Figure 29, with the remaining outlets functioning in a similar way achieving a good overall fill rate and net profit. On each day, the outlet satisfies customer demand which reduces the number of units available. An inventory target was calculated based on the sum of the forecasted demand for the next seven days and an order (action) was placed upstream based on the difference between the inventory target and the units available. There was a large fluctuation in demand on day 135 possibly due to a promotion in the company. However, the results also indicate that the model calculates a larger inventory target and orders more units before day 135 in an attempt to store sufficient units to satisfy the large fluctuation in demand.



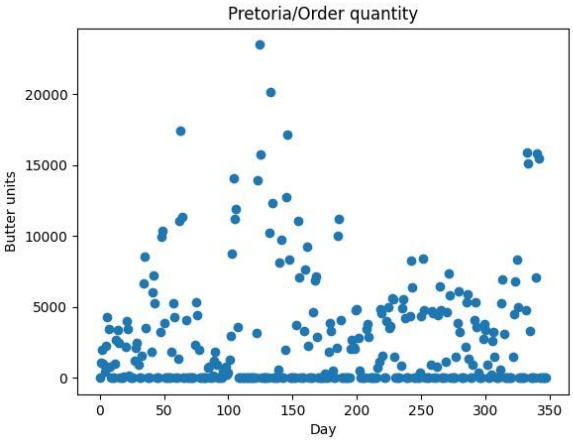

(a)




(b)



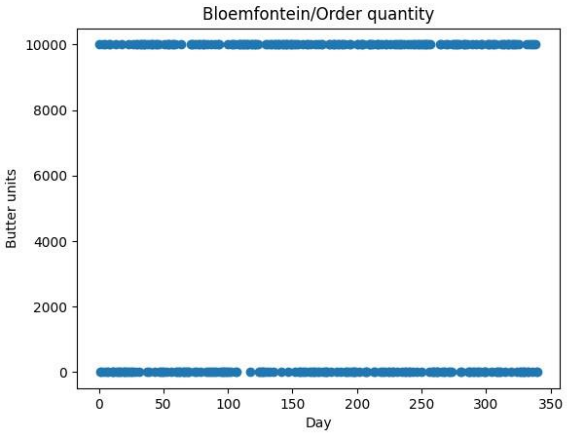

(c)



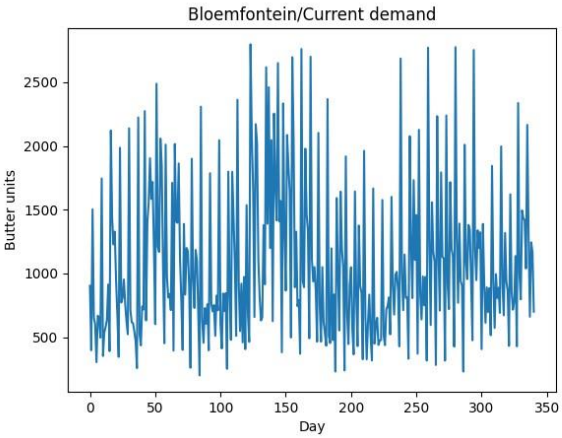

(d)

Figure 29: Results showing the (a) current demand, (b) units available, (c) inventory target and (d) order quantity of the Pretoria outlet in the heuristic model.

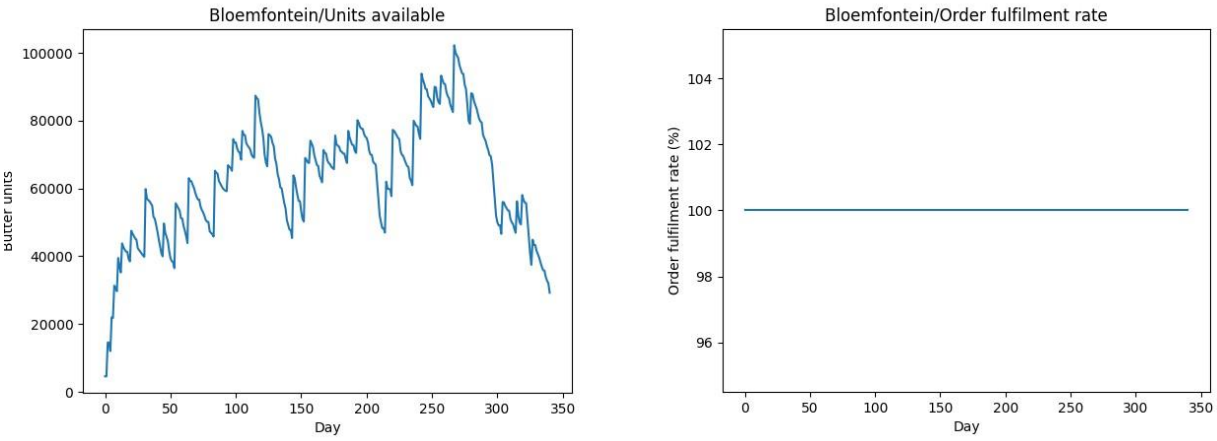
The results for the RL model using the PPO algorithm are shown below. The Bloemfontein and Cape Town outlets had similar results achieving a 100% fill rate but store too many excess units. The Pretoria and East London outlets achieved a fill rate of between 95-98% while attempting to store fewer units to save on storage costs. The Durban outlet only achieved a fill rate of approximately 66% since it stored very few units. The results for the Bloemfontein, Pretoria and Durban outlets is shown in Figure 30, Figure 31 and Figure 32 respectively. The inclusion of the current profit in the reward function caused the RL models to learn to order either 0 units or the maximum ordering quantity to save on delivery costs for each outlet as delivery costs exceeded storage costs (Table A.4). This is in contrast to the ordering quantity of the outlets in the heuristic model where the ordering quantity fluctuated daily.




(a)

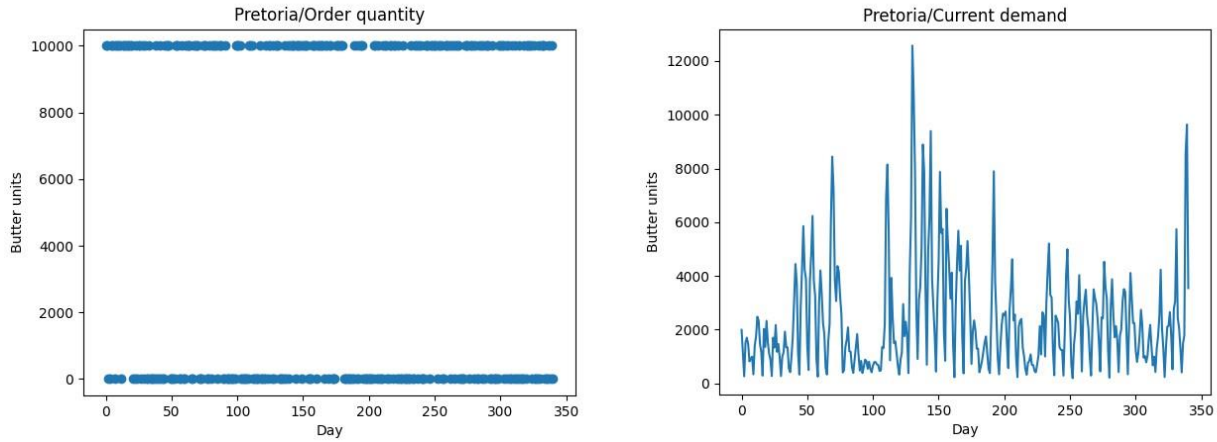



(b)

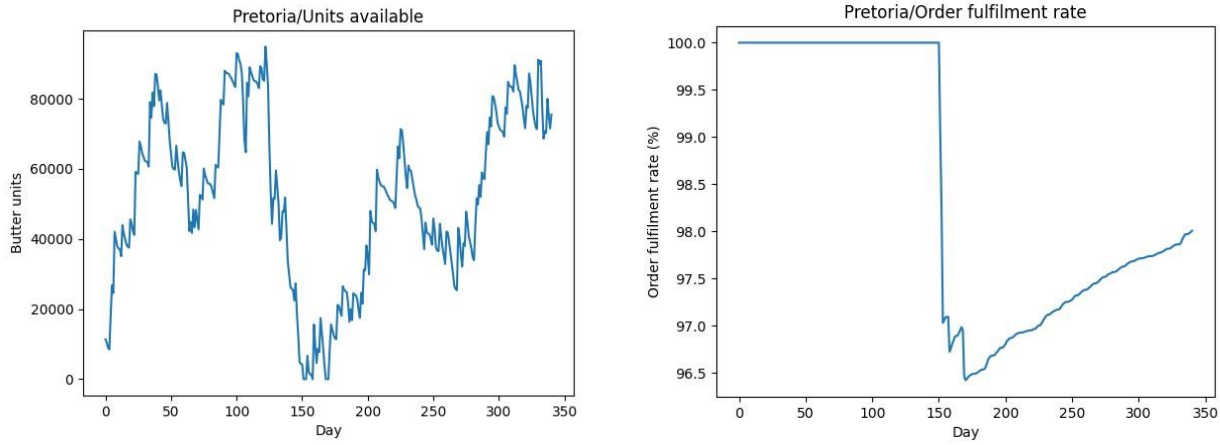



(c) (d)

Figure 30: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Bloemfontein outlet in the PPO model.




(a) (b)




(c) (d)

Figure 31: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Pretoria outlet in the PPO model.

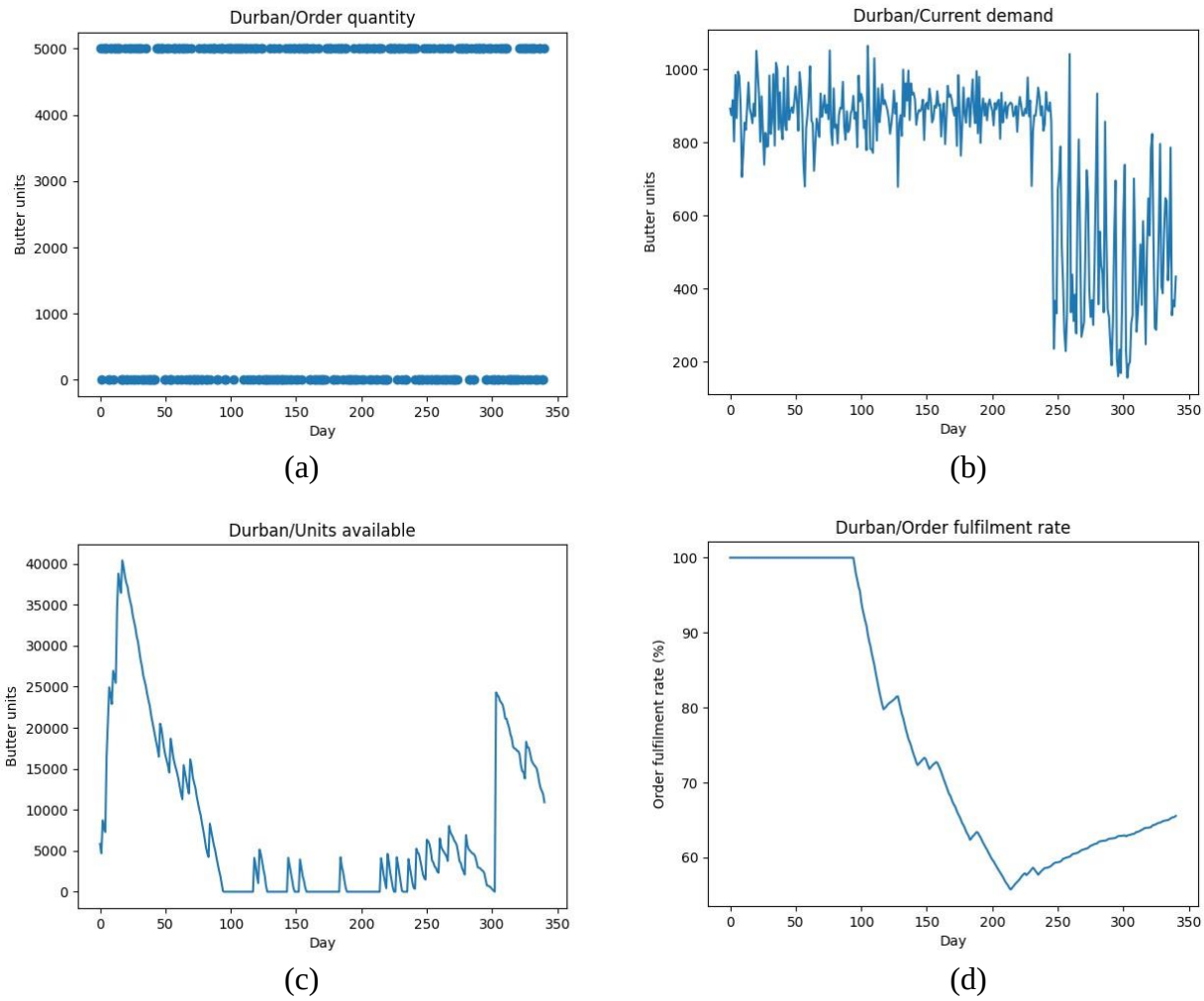



Figure 32: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Durban outlet in the PPO model.

The results for the RL model using the DDPG algorithm are shown below. The Durban, East London and Cape Town outlets performed in a similar way where no orders were placed causing insufficient units being available, resulting in low fill rates. This is shown in Figure 33 using the Durban outlet as an example. A possible reason for this is the higher delivery costs that occur since these three locations are the furthest from both the production and warehouse in Johannesburg. Additionally, these three locations have the lowest average demand across the year (Table 12) hence, the model may have learned that the inventory investment in sending units to these locations exceed the revenue obtained from satisfying customer demand.

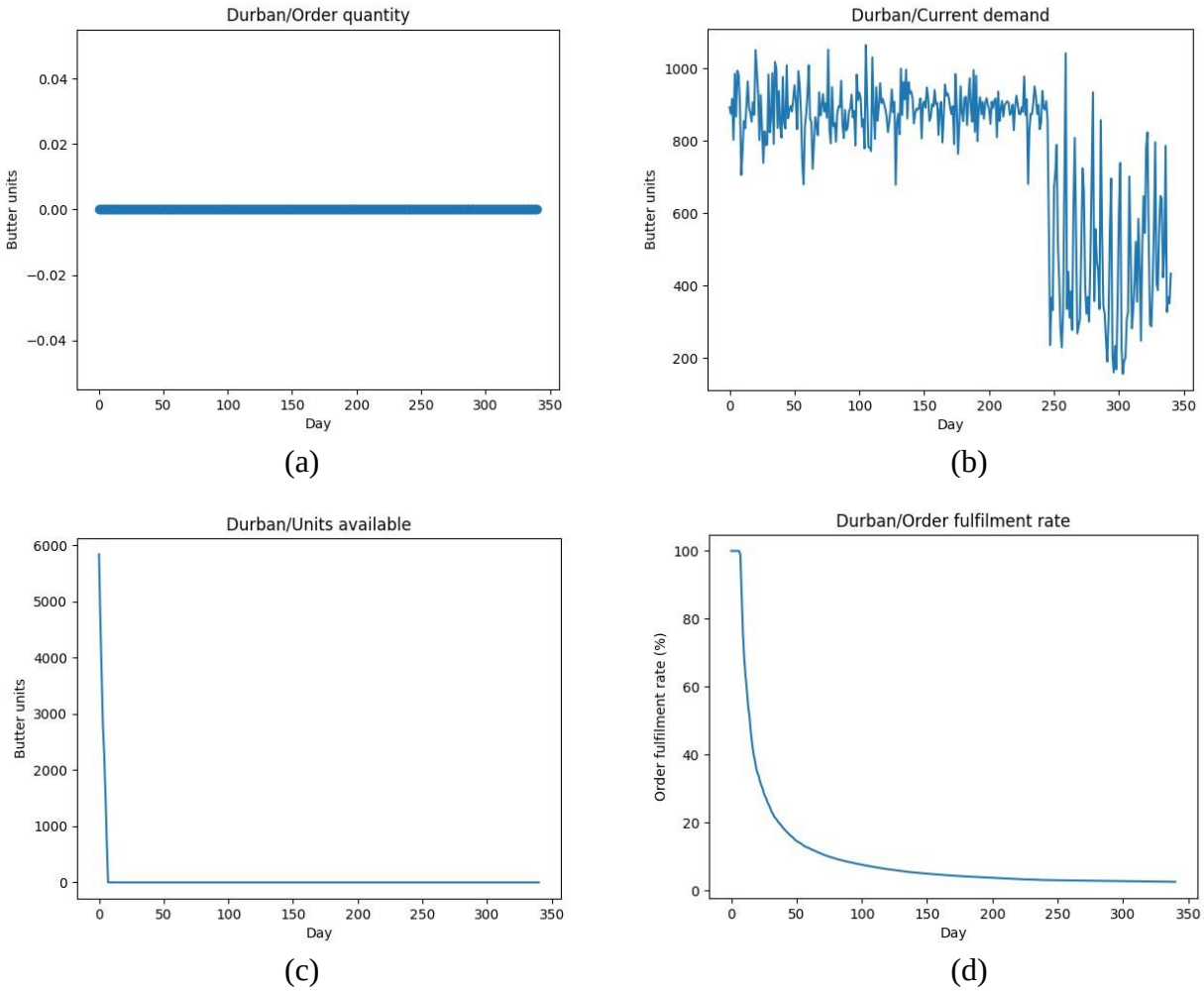



Figure 33: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Durban outlet in the DDPG model.

The Bloemfontein and Pretoria outlets performed in a similar way where a constant order (maximum ordering quantity) was placed throughout the year. This resulted in these outlets having a 100% fill rate but the number of units stored gradually increased towards the end of the year. This is shown in Figure 34 using the Pretoria outlet as an example. The average number of units held by the Pretoria outlet in the last 100 days for the DDPG model was approximately three times higher than that of the PPO model. This could have significantly limited the DDPG model from achieving an even higher net profit and potentially improve on the heuristic. The constant action used in these outlets differs from the actions used in both the heuristic and RL models with the DDPG algorithm resulting in unnecessarily high storage costs for these outlets.

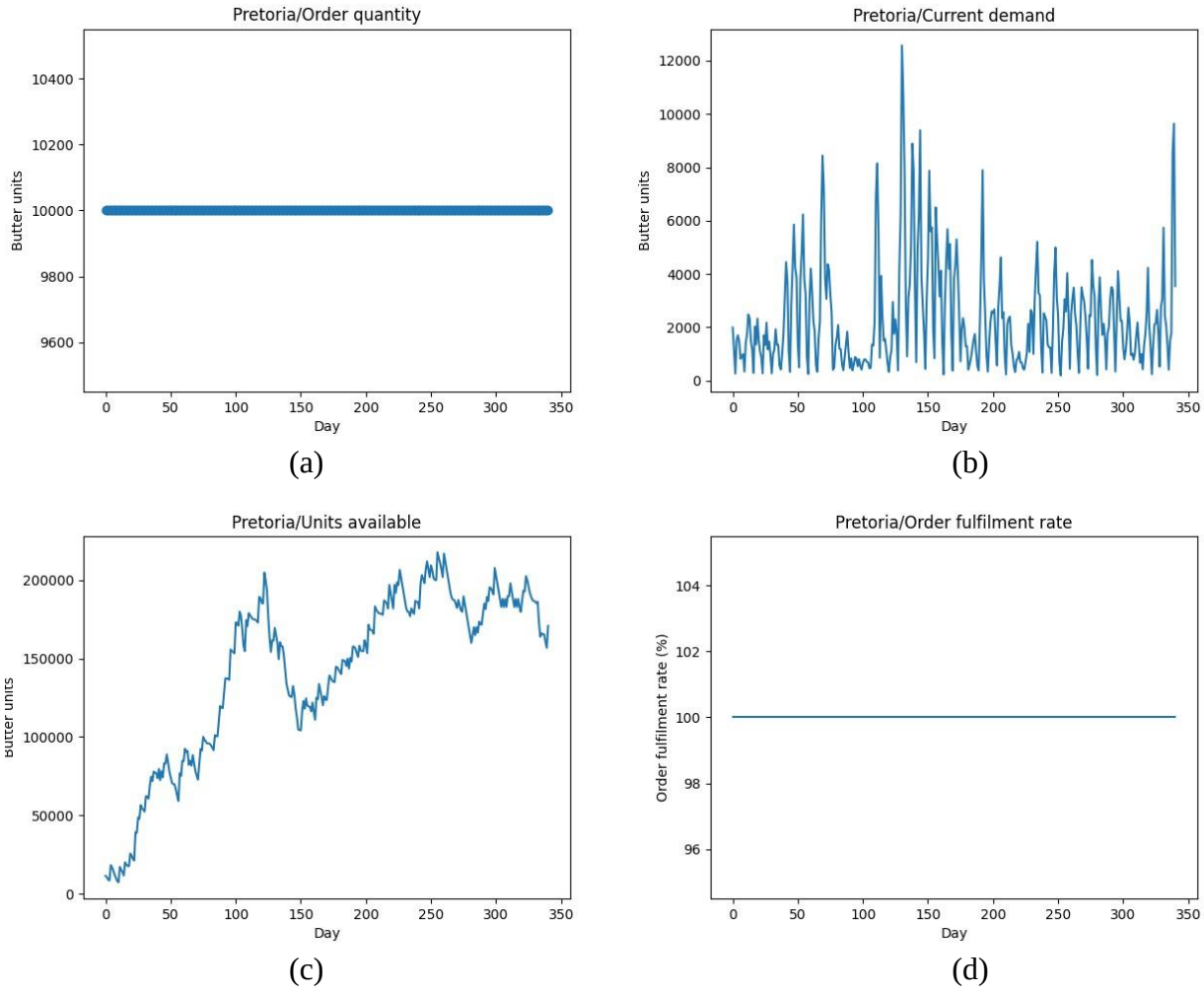


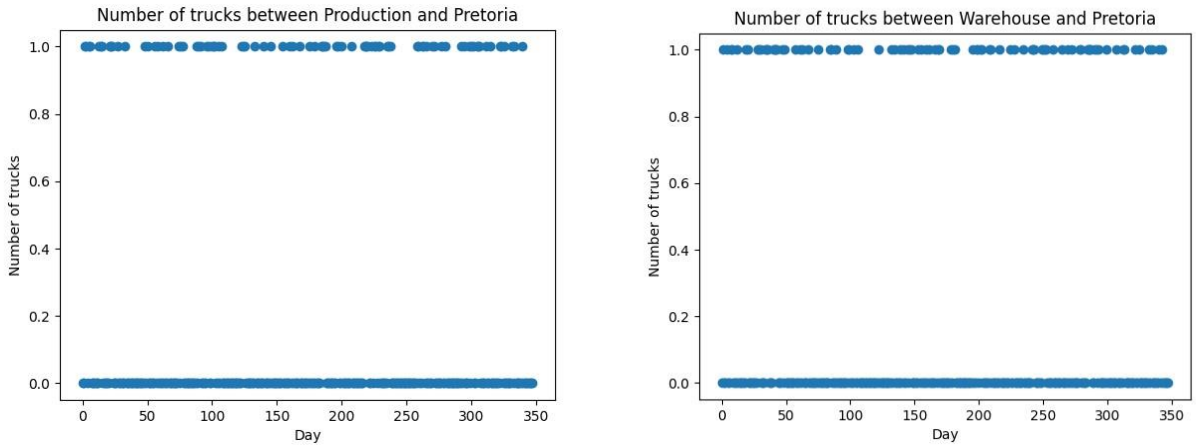

Figure 34: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Pretoria outlet in the DDPG model.

### 4.4.2 Results showing reasons for low fill rate at certain outlets

The Durban outlet performed worse than the remaining outlets in both RL models as it achieved a lower fill rate. The root cause was linked to this outlet ordering 95% of the time from the warehouse and the warehouse not having sufficient units to satisfy customer demand and send units to downstream locations. Possible reasons for this included:

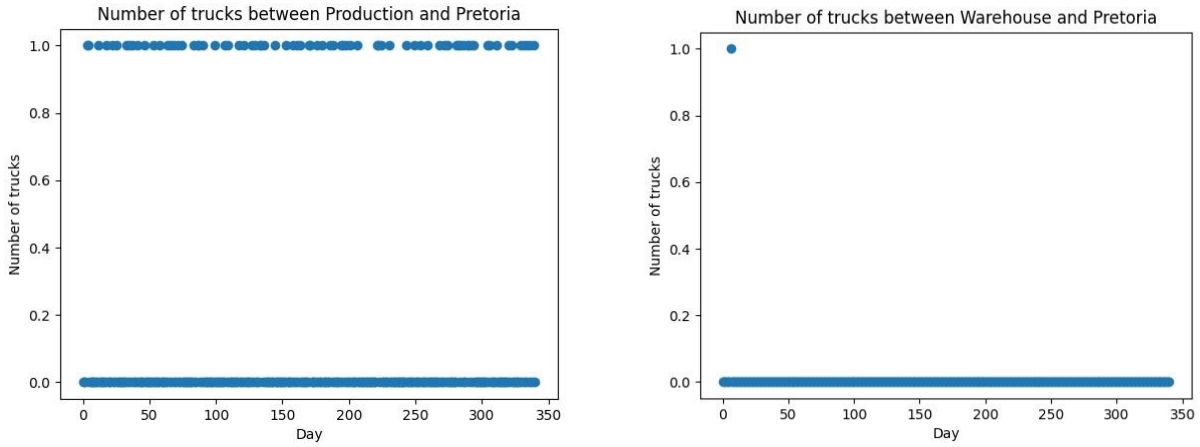
1. The number of times that the Pretoria outlet ordered from the production and warehouse facilities was almost equal hence, the number of trucks utilized from the production and warehouse to the Pretoria outlet was analysed. The number of trucks used between the production unit and Pretoria and between the warehouse and Pretoria are similar for the heuristic as shown in Figure 35. However, in the PPO (Figure 36) and the DDPG models (Figure 37), the Pretoria outlet was mostly supplied by the production unit. The RL models possibly learned that sending units directly from the production unit to the outlets was

cheaper and quicker while only sending a little bit of surplus stock to the warehouse for storage.



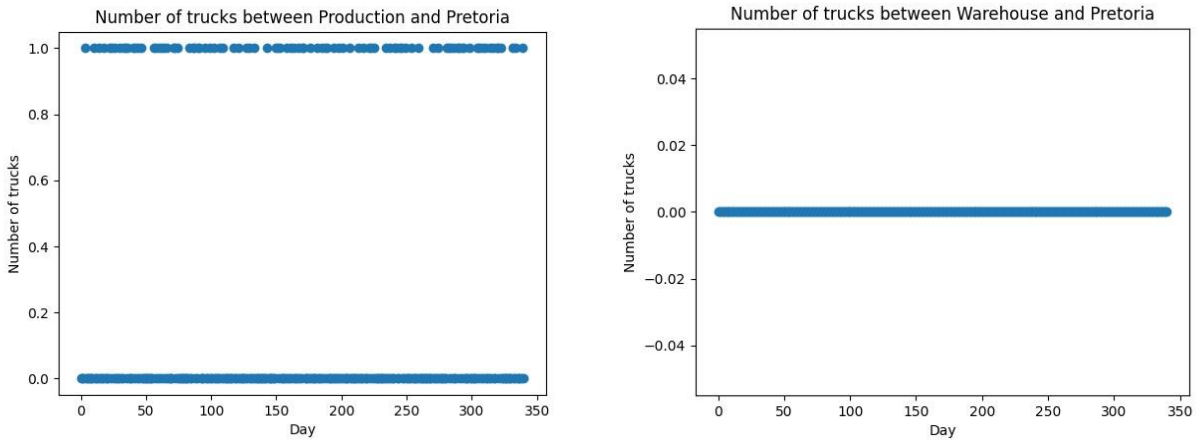

(a) (b)

Figure 35: Trucks from (a) production to Pretoria and from (b) warehouse to Pretoria for the heuristic model.




(a) (b)

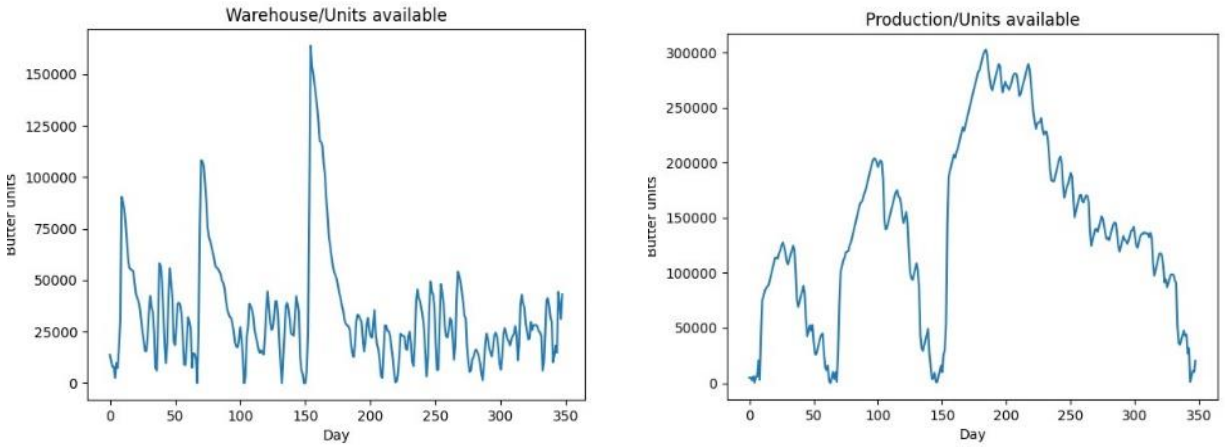
Figure 36: Trucks from (a) production to Pretoria and from (b) warehouse to Pretoria for the PPO model.



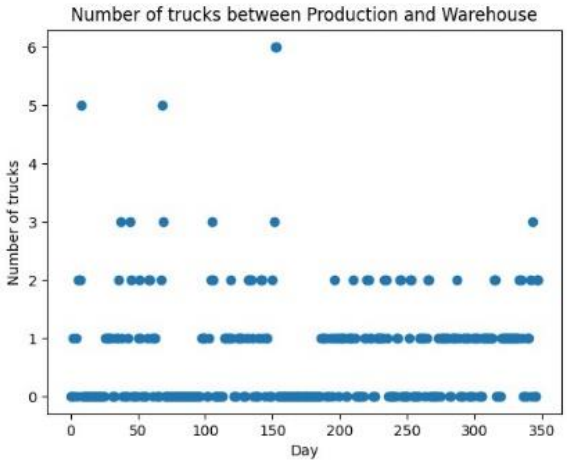
(a) (b)

Figure 37: Trucks from (a) production to Pretoria and from (b) warehouse to Pretoria for the DDPG model.

2. The number of units available at the warehouse and production and the number of trucks in operation between these two facilities for the heuristic, PPO and DDPG models are shown in Figure 38, Figure 39 and Figure 40 respectively. The heuristic model stored sufficient units at the warehouse most of the time but the RL models did not. In the heuristic model, it was shown that the production unit had sufficient units and sent trucks frequently from the production unit to the warehouse. However, in the RL models, it was shown that the production unit did not store sufficient units and could not fill trucks completely hence, very few trucks were sent from the production unit to the warehouse. The RL models possibly learned to minimize the number of units stored at the production unit since the storage costs at the production unit were four times higher than the warehouse (Table A.3).

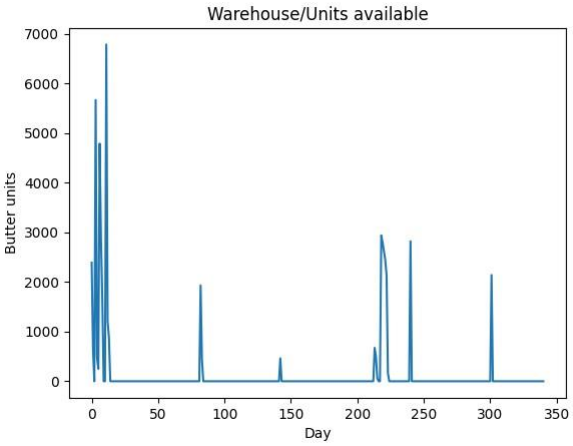



(a) (b)

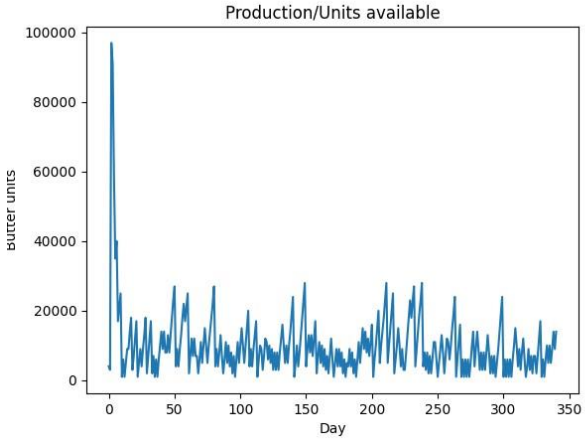



(c)

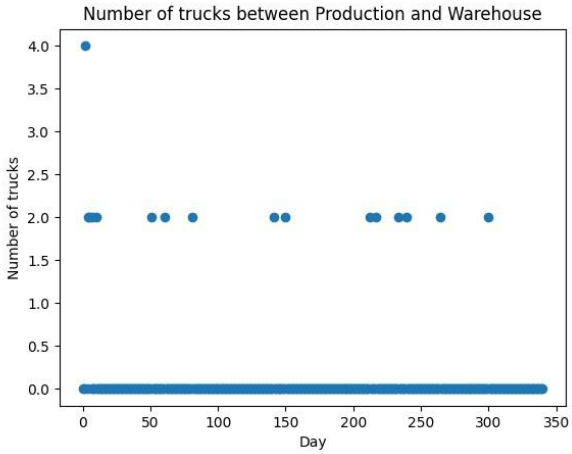
Figure 38: (a) Warehouse units available (b) Production units available and (c) Trucks between production and warehouse for the heuristic model.




(a)

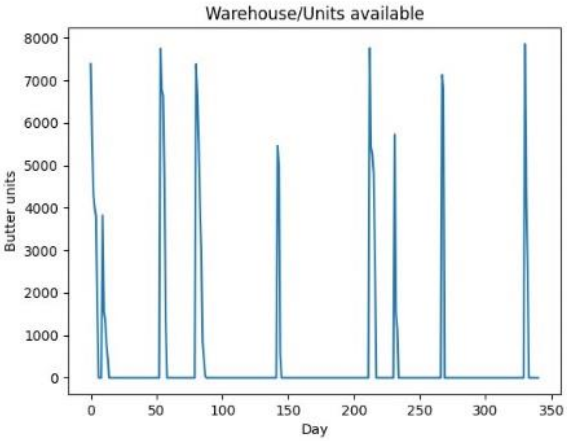



(b)

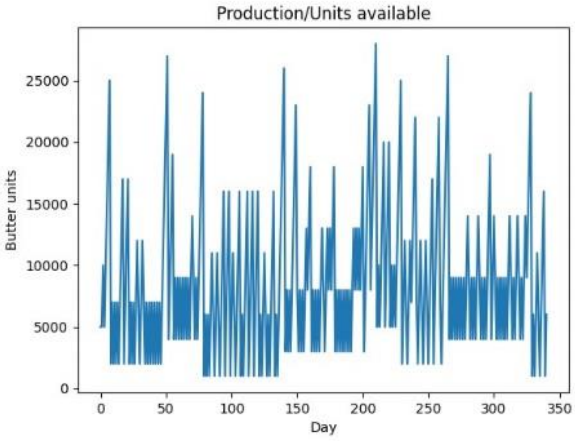



(c)

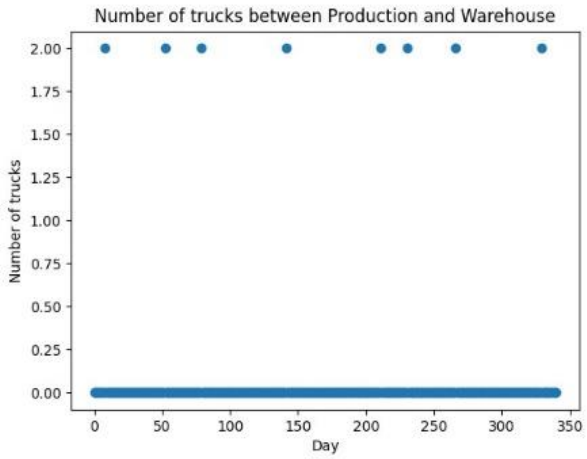
Figure 39: (a) Warehouse units available (b) Production units available and (c) Trucks between production and warehouse for the PPO model.




(a)




(b)

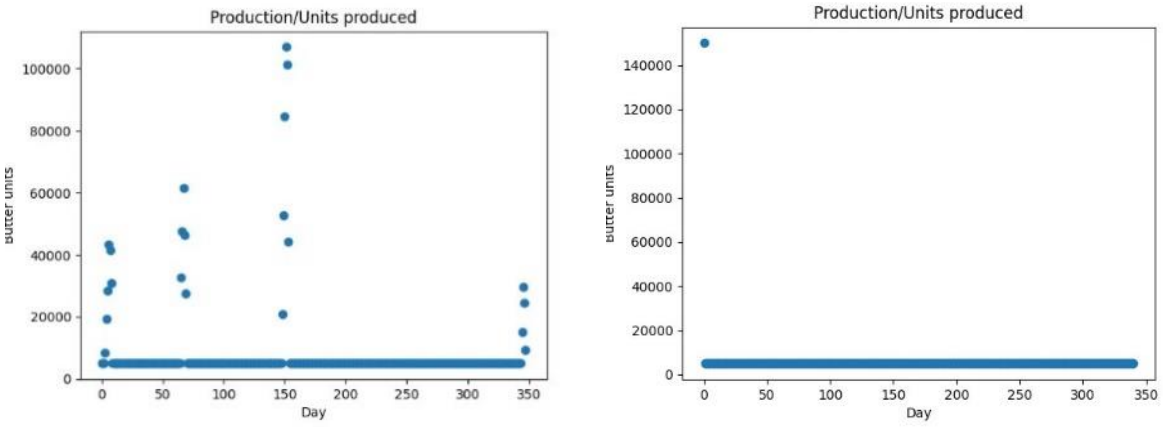



(c)

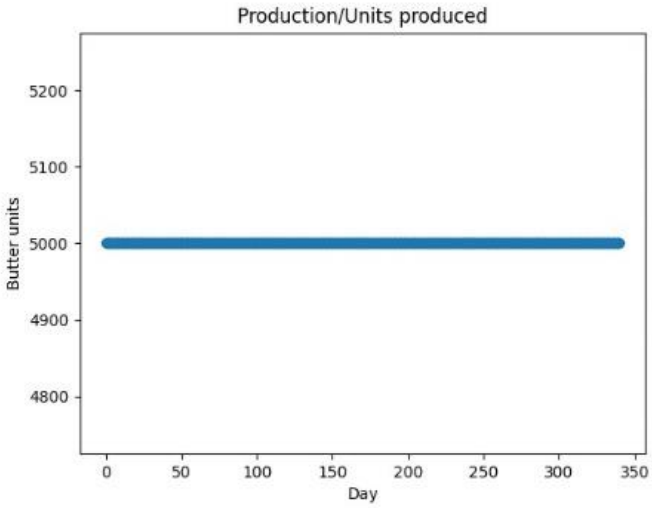
Figure 40: (a) Warehouse units available (b) Production units available and (c) Trucks between production and warehouse for the DDPG model.

### 4.4.3 Results showing the comparison of units produced in the heuristic and RL models

The units produced at the production unit are compared for the heuristic and RL models as shown in Figure 41. The DDPG algorithm produced a constant value of 5000 units (minimum production limit) daily whereas the PPO algorithm produced 150 000 units (value of the maximum production limit) on day 1 and used that to supply the rest of the supply chain for the rest of the year together with a constant value of 5000 units being produced daily from day 2. This differed from the heuristic where the production quantities varied more frequently between the minimum and maximum production limit. The RL models showed potential to provide a value-add to the company by consolidating the production through consistent production volumes to provide better asset utilization (sending full trucks) and cost efficiency (minimizing inventory on hand). However, the production quantities in the RL models were found to be extremely small resulting in a lower overall fill rate being achieved in the supply chain.



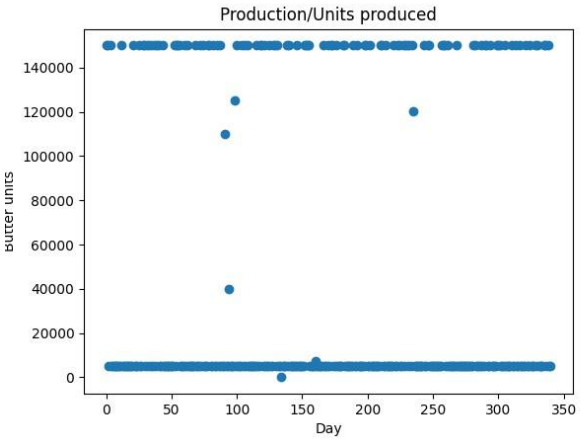

(a) (b)



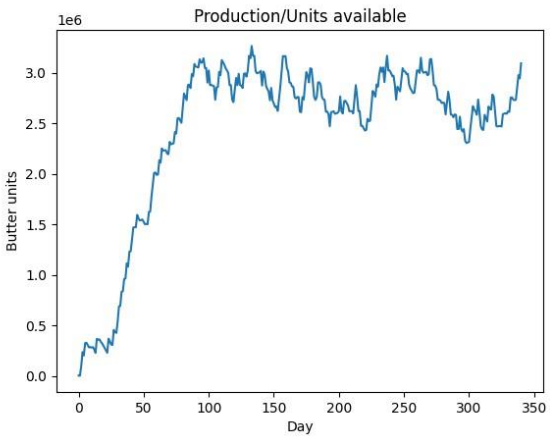

(c)

Figure 41: Units produced at the production unit for the (a) heuristic model (b) PPO model and (c) DDPG model.

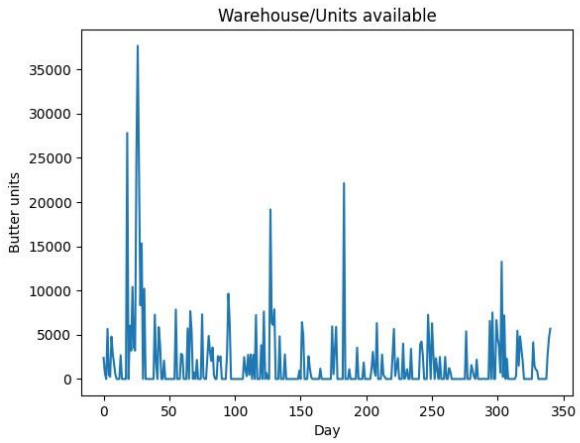
The RL model using the PPO algorithm was tested again after reducing the production storage cost by a factor of 4 with the results shown in Figure 42. The results show that the production quantities varied more frequently and more units became available at the warehouse increasing its fill rate. Additionally, even though the production unit had sufficient units, the warehouse fill rate was still not very high. However, this model can be optimized in the future to determine whether it is able to provide higher profits compared to the original supply chain. Even though this model was not profitable (possibly due to the extremely high number of units being available at the production unit), it was concluded that the production storage cost plays a significant role in determining the number of units being available at the production unit.



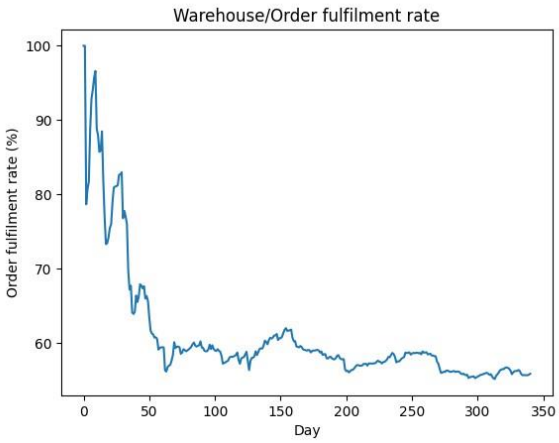

(a)




(b)




(c)




(d)

Figure 42: (a) The number of units produced, (b) production units available, (c) warehouse units available and (d) warehouse fill rate using a quarter of the production storage cost for the PPO model.

Since the warehouse was not really utilized in the RL models, it was logical to remove it from the supply chain. The standard deviation of the warehouse was higher than most of the outlets which means larger fluctuations in demand occur at the warehouse (Table 12). This could have been a contributing factor to the lower fill rate being achieved by the warehouse. Sending units from the production unit to the warehouse which are both in the same city could have led to unnecessary additional delivery and storage costs. The use of two distribution channels to send units to the outlets and having lots of units held in a backlog could have added additional complexity which makes training of the model more challenging. Thus, a new scenario was tested where the production and warehouse facilities were merged to determine whether it is still true that a lower fill rate provides a higher net profit as well as check if the supply chain is more responsive to the customer demands.

### 4.5 Comparing the heuristic and RL model results for scenario 2: Merging the production and warehouse facilities within the supply chain

In scenario 2, the production and warehouse facilities were merged, the customer demand at the warehouse moved to the production and the parameters related to the production unit were used. The sum of units in the backlog between each facility in the supply chain was included in the state space. This provided improved performance as it possibly helped the model better understand the inventory position of the upstream locations. The action and reward space remained similar but unique action scaling, reward shaping and hyperparameter tuning were performed to improve the RL models for this particular scenario.

Table 17 shows the performance comparison of the RL models in scenario 2 with the benchmark heuristic. Both PPO and DDPG models obtained a higher fill rate as less units were unsatisfied compared to scenario 1. This shows that merging the production and warehouse facilities provide improve customer accommodation. The PPO model in scenario 2 struggled to learn to store an optimal number of units. A large number of units were stored possibly due to the reduced costs and time required to transport units to the outlets in this scenario. A large number of units became obsolete which primarily resulted in an overall loss at the end of the year. However, the DDPG model learned to store a certain number of units that prevented obsolete inventory from occurring across the supply chain throughout the year. This primarily resulted in the model achieving a higher fill rate compared to scenario 1 and the benchmark heuristic. Detailed results regarding the comparison of the RL models between scenario 1 and scenario 2 are presented in the following sections.

Table 17: Comparison of overall performance between different models in scenario 2.

| | **Heuristic** | **PPO** | **DDPG** |
|---|---|---|---|
| Net profit | R11 042 078 | - R3 882 116 | R14 036 803 |
| Fill rate | 97.41% | 80.55% | 70.60% |
| Obsolete inventory | 56 945 | 1 145 234 | 0 |
| Units unsatisfied | 62 315 | 449 565 | 679 602 |

### 4.5.1 Results showing the comparison of the PPO models between scenario 1 and scenario 2

The fill rate of the Bloemfontein outlet reduced significantly from scenario 1 to scenario 2 as the model learned to order too much units initially leading to large amounts of stock available. The production unit then significantly reduced the number of units it produced since there were excess units across all locations in the supply chain. However, the large number of units at the outlets eventually turned into obsolete inventory resulting in insufficient units being available to satisfy customer demand towards the end of the year. This results for the Bloemfontein outlet are shown in Figure 43 with the Pretoria outlet functioning in a similar way.

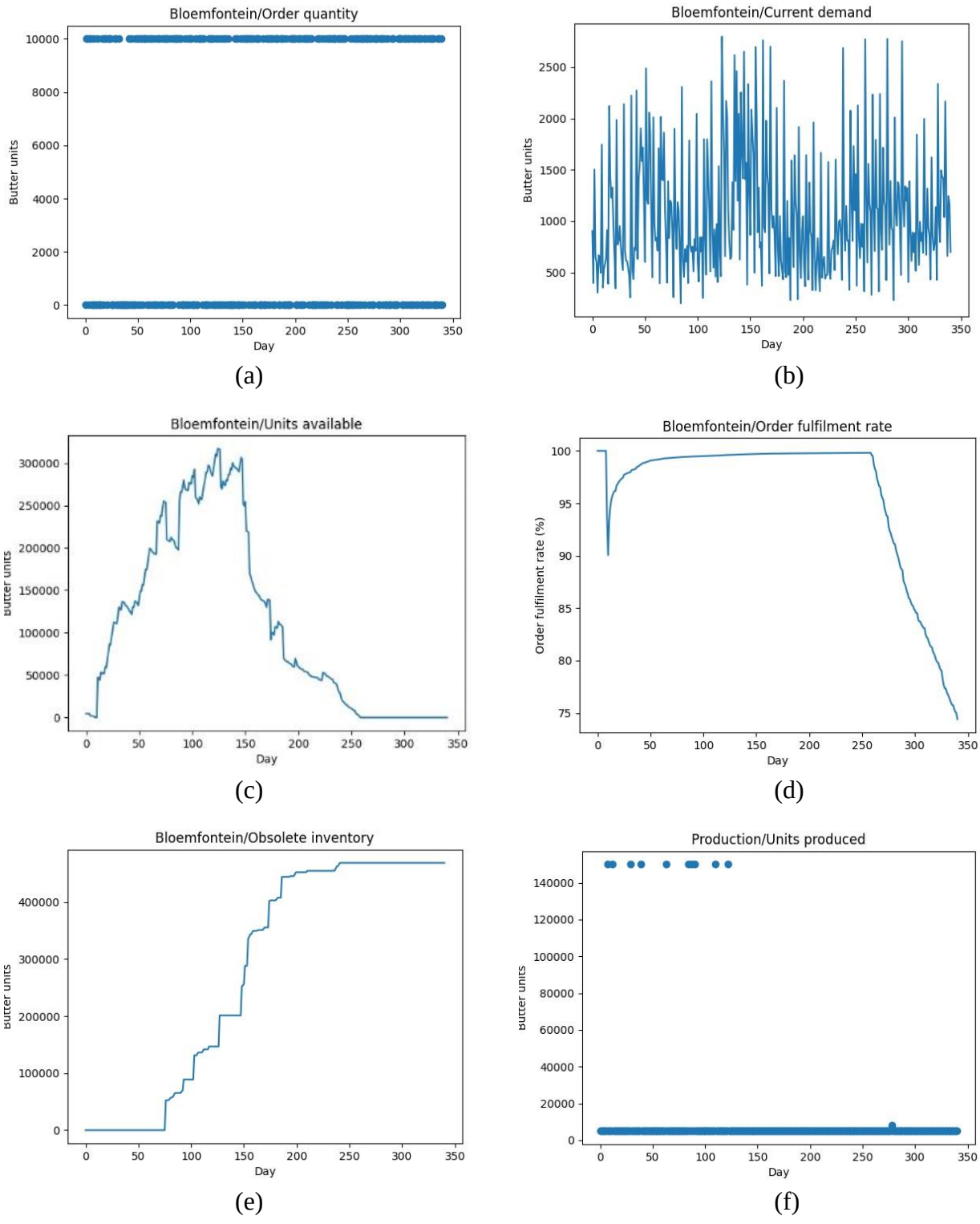
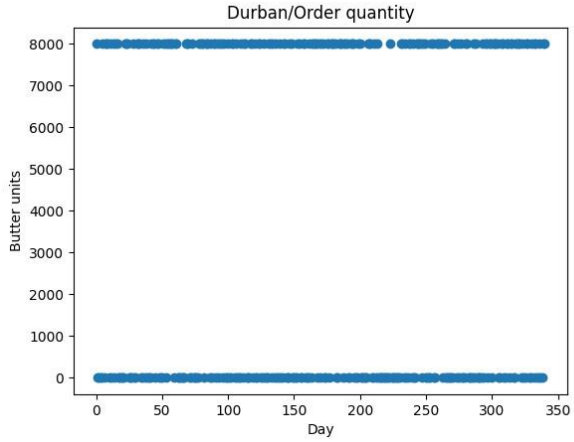


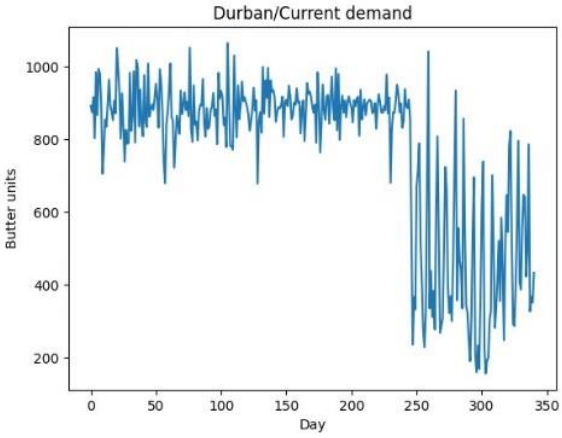

Figure 43: Results showing the (a) order quantity, (b) current demand, (c) units available, (d) fill rate and (e) obsolete inventory of the Bloemfontein outlet and (f) the number of units produced at the production unit in the PPO model for scenario 2.

The Durban, Cape Town and East London outlets performed in a similar way achieving a 100% fill rate but at the expense of storing a significant number of units which eventually turned into obsolete inventory. The results for the Durban outlet are shown in Figure 44 as an example. The increased

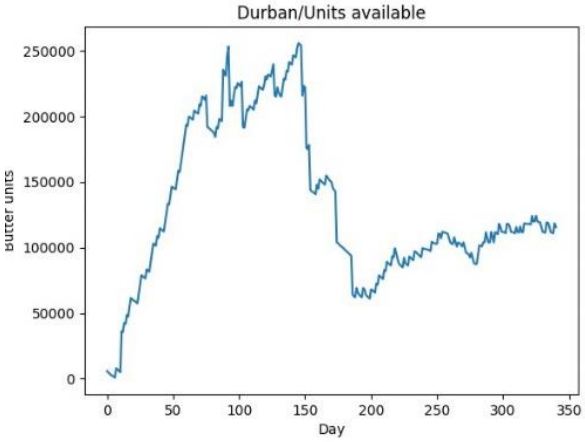
amount of obsolete inventory occurring in the PPO model of scenario 2 compared to scenario 1 was the primary reason for the significant drop in net profit. The trend in the order quantity of the outlets in scenario 2 was similar to scenario 1 where the model attempted to order either 0 units or the maximum ordering quantity to save on delivery costs.



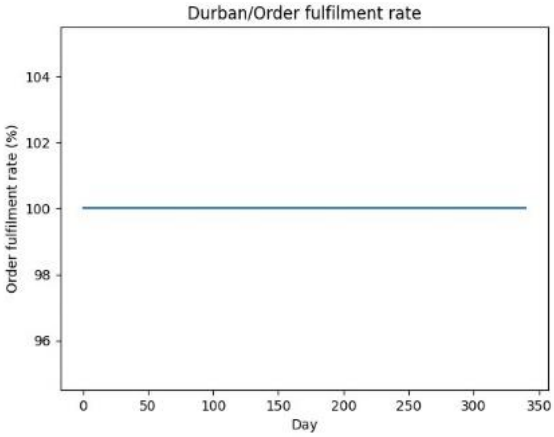

(a)



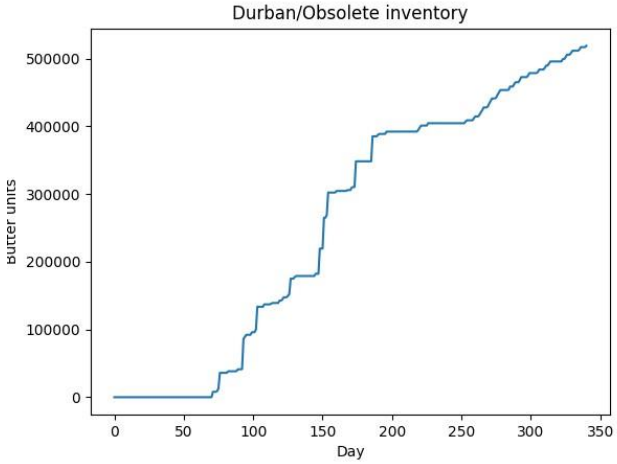

(b)




(c)



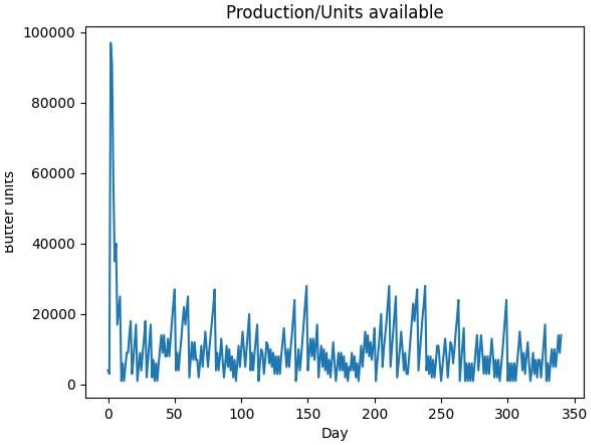

(d)



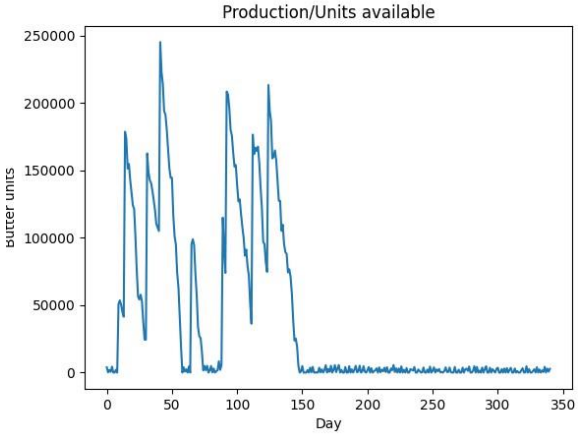

(e)

Figure 44: Results showing the (a) order quantity, (b) current demand, (c) units available, (d) fill rate and (e) obsolete inventory of the Durban outlet in the PPO model for scenario 2.

Figure 45 shows the different approaches used by the production unit between scenario 1 and scenario 2. In scenario 1, the production unit produced the maximum production capacity on day 1 and the minimum production capacity from day 2 onwards and used that to supply the rest of the supply chain. In scenario 2, the production unit produced the maximum production capacity a few times in the first 125 days and the minimum production capacity on every other day. This resulted in the PPO model in scenario 2 having a higher total manufacturing, delivery and storage cost with more obsolete inventory occurring compared to the PPO model in scenario 1 as shown in Table 18. Figure 46 shows that moving customer demand from the warehouse to the production unit increased the fill rate as it was easier, cheaper and quicker to satisfy the customer demand. This resulted in an increase in the overall fill rate providing more revenue for the PPO model in scenario 2 compared to the PPO model in scenario 1.

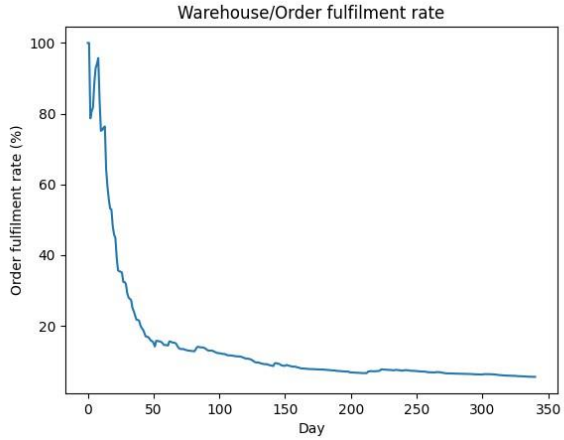



(a)

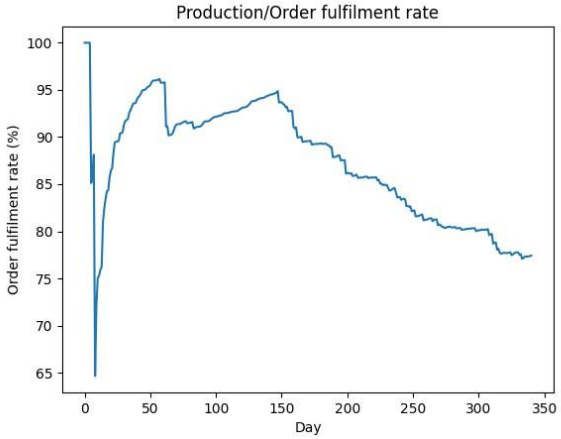



(b)

Figure 45: Production units available in (a) scenario 1 and (b) scenario 2 for the PPO models.




(a)




(b)

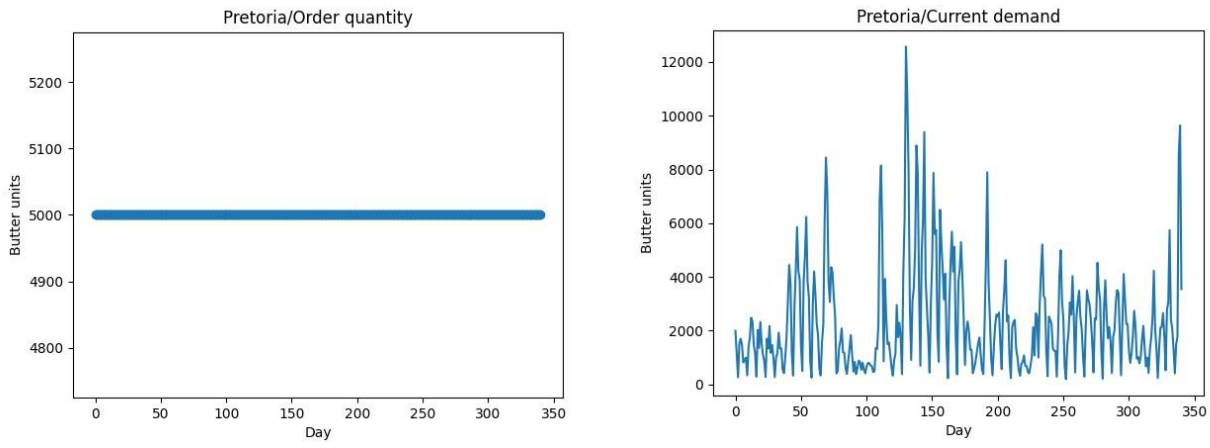
Figure 46: The fill rate at the (a) Warehouse in scenario 1 and (b) Production unit in scenario 2 for the DDPG models.

Table 18: Comparison of performance between the unmerged and merged supply chain using PPO.

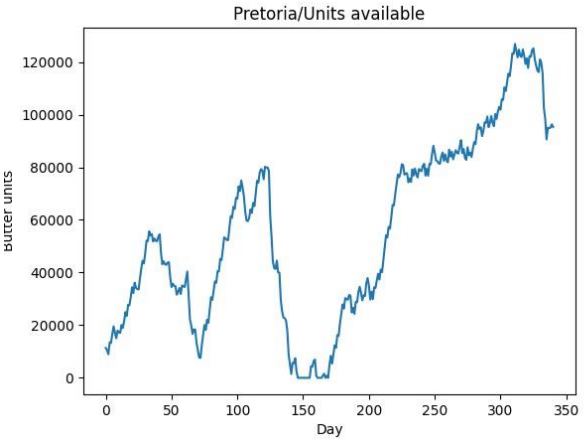
| Parameter | PPO model in scenario 1 (unmerged supply chain) | PPO model in scenario 2 (merged supply chain) |
|---|---|---|
| Revenue [in millions] | 37.5 | 46.5 |
| Total delivery cost [in millions] | 6.05 | 12.1 |
| Total manufacturing cost [in millions] | 22.0 | 37.6 |
| Total storage cost [in millions] | 0.304 | 0.712 |
| Obsolete inventory [in millions] | 0.229 | 1.15 |

### 4.5.2 Results showing the comparison of the DDPG models between scenario 1 and scenario 2

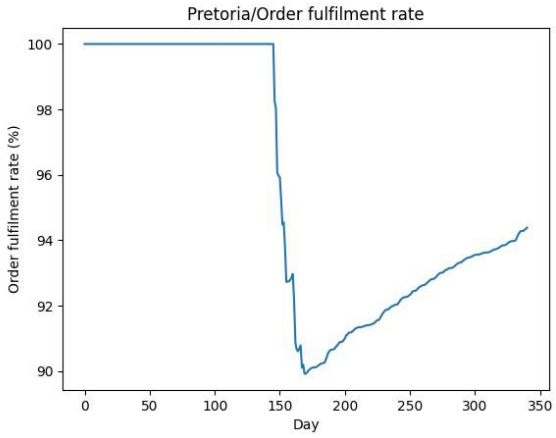
The Durban, East London and Cape Town outlets of the DDPG model performed similar in both scenario 1 and scenario 2 where no actions were placed resulting in extremely low fill rates for similar reasons as discussed in Section 4.4.1. The Pretoria and Bloemfontein outlets of the DDPG model performed similar in both scenario 1 and scenario 2. The fill rate at these two locations marginally dropped as much fewer units were stored which reduced the storage costs and the amount of obsolete inventory that occurred. This contributed to the higher overall net profit in the DDPG model of scenario 2 compared to the DDPG model in scenario 1. Figure 47 shows an example using the Pretoria outlet of the DDPG model in scenario 2 where the ordering quantity was half compared to the Pretoria outlet of the DDPG model in scenario 1 (Figure 34). This resulted in less units being stored while still being able to satisfy most of the customer demand. Similarly, the units available at the production facility in scenario 2 was significantly less compared to that in scenario 1 as shown in Figure 48 resulting in lower storage costs and an increase in overall net profit.




(a) (b)

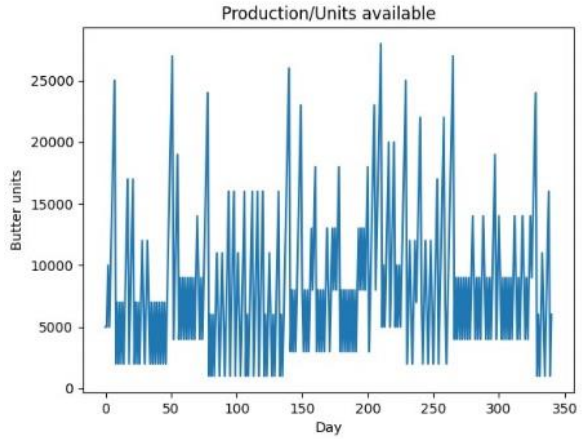



(c)

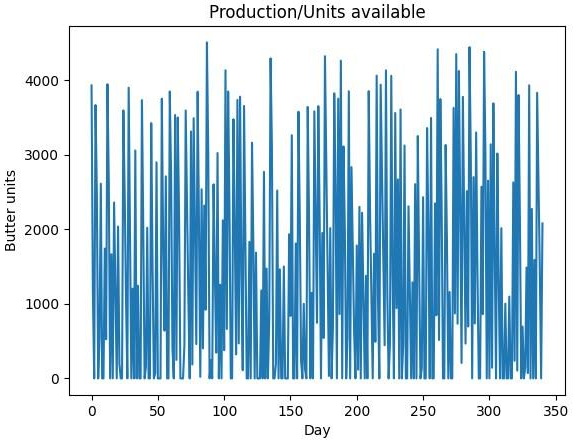



(d)

Figure 47: Results showing the (a) order quantity, (b) current demand, (c) units available and (d) fill rate of the Pretoria outlet in the DDPG model for scenario 2.



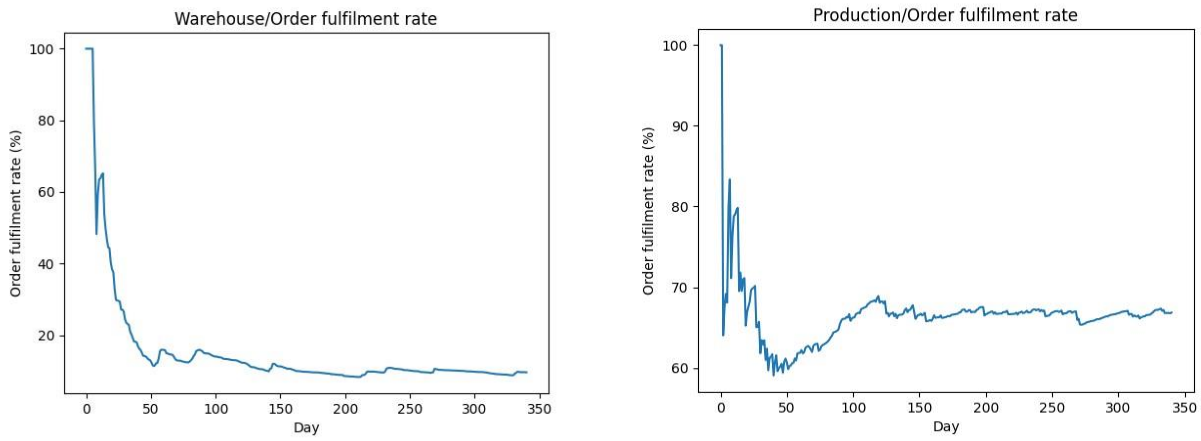

(a)




(b)

Figure 48: Production units available in (a) scenario 1 and (b) scenario 2 for the DDPG models.

The effect of moving the customer demand from the warehouse to the production resulted in a larger proportion of the customer demand being met contributing to the higher overall fill rate being achieved in the DDPG model of scenario 2 compared to scenario 1. Figure 49 shows the increase in fill rate as customer demand moves from the warehouse in scenario 1 to the production unit in scenario 2. The higher fill rate resulted in higher revenue for the DDPG model in scenario 2 as shown in Table 19. Smaller shipments were used in the DDPG model in scenario 2 which led to higher delivery costs. However, this minimized the amount of inventory available throughout the supply chain which reduced storage costs. Additionally, no obsolete inventory occurred which contributed to the higher net profit being achieved. The same number of units were manufactured in both scenarios but scenario 2 managed to distribute the inventory across the supply chain more effectively.




(a) (b)

Figure 49: The fill rate at the (a) Warehouse in scenario 1 and (b) Production unit in scenario 2 for the DDPG models.

Table 19: Comparison of performance between the unmerged and merged supply chain using DDPG.

| **Parameter** | **DDPG model in scenario 1 (unmerged supply chain)** | **DDPG model in scenario 2 (merged supply chain)** |
|---|---|---|
| Revenue [in millions] | 32.0 | 40.8 |
| Total delivery cost [in millions] | 1.20 | 6.36 |
| Total manufacturing cost [in millions] | 20.3 | 20.3 |
| Total storage cost [in millions] | 0.347 | 0.101 |
| Obsolete inventory [in millions] | 0.224 | 0 |

### 4.6 Training results of PPO and DDPG

The PPO and DDPG models have distinct training metrics which were briefly described in Section 2.7.2. Interesting training charts related to the PPO algorithm are presented in Figure 50. The trends in the graphs are of primary interest with the values not providing useful information as there is no baseline for them to be compared to. Others charts were also considered but did not provide useful insights hence, they are included in Appendix D. Additionally, the training metrics are only presented for the PPO and DDPG algorithms in scenario 1 as the graphs followed a similar trend in scenario 2 which is included in Appendix E.

The *approx_kl* parameter was low indicating that large policy updates did not occur which maintains stability during the training process. The absolute value of the *entropy loss* parameter gradually increased with no sudden spikes making the model stable as it balances the exploration-exploitation trade-off. The entropy loss value did not converge possibly due to the complex and stochastic nature

of the environment resulting in the model to continue explore different actions. The *policy gradient loss* decreased before stabilizing indicating that the model converged to a reasonable policy. The value of the *value loss* started off high but the model quickly learned to minimize it. The *value loss* decreased and stabilized over time indicating that more reliable predictions of future rewards linked with different state-action pairs occurred which helped the agent make better decisions. The *value loss* was very large which could possibly be linked to a large reward function or the vastness and complexity of the continuous action and state space making it difficult to experience identical states frequently. The *explained variance* attempts to increase allowing for a more accurate prediction but constantly fluctuates due to the model giving priority to minimize the *value loss* as it has a much larger value.

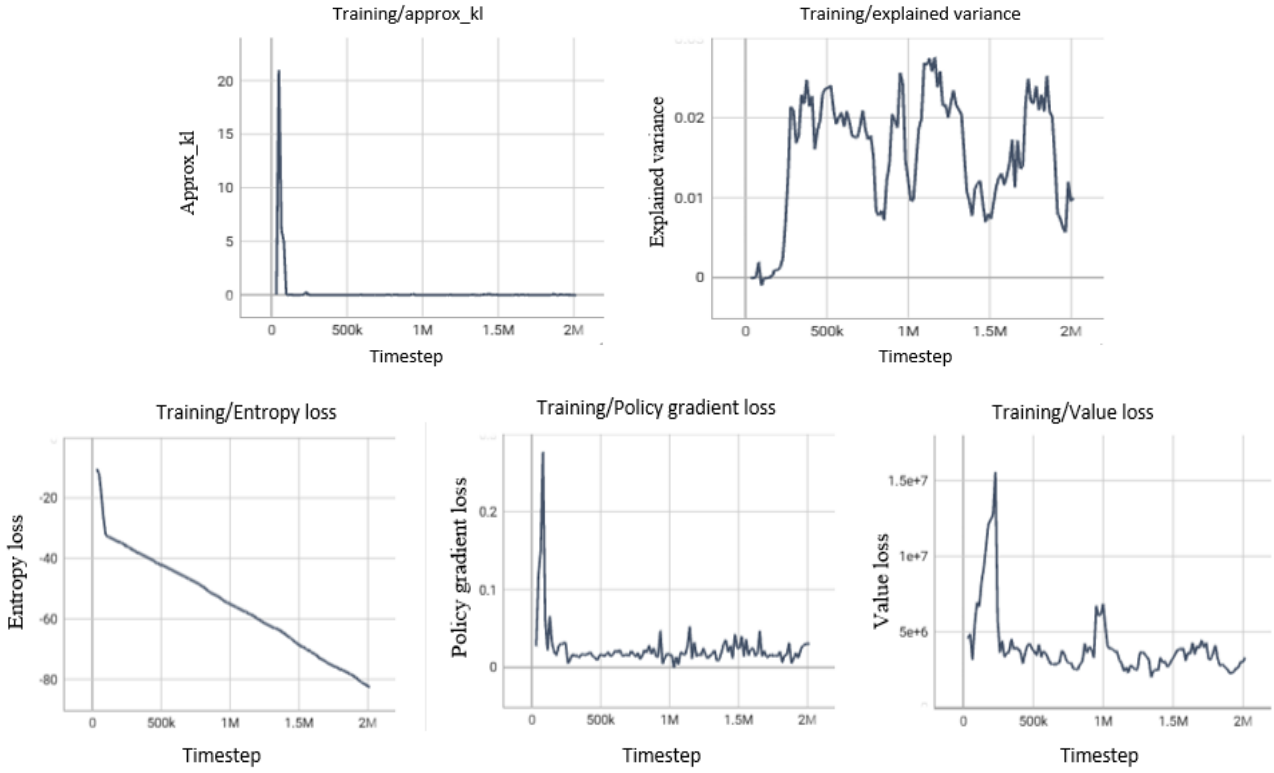



Figure 50: Various training parameters of the PPO model.

The training charts relating to the DDPG model are presented in Figure 51. It is shown in the *actor loss* graph that the value of the expected return increased before stabilizing indicating that the model learned the best actions that maximized the rewards. The *critic loss* graph indicated that the model learned to make accurate estimates of the Q-values over time providing better stability and convergence of the critic NN.

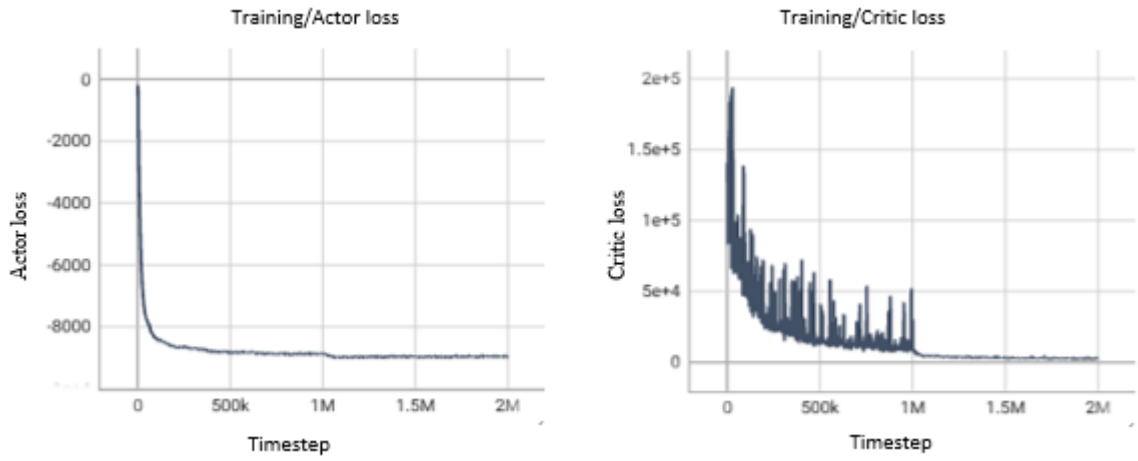



Figure 51: Training parameters of the DDPG model.

### 4.7 Comparing the performance between the PPO and DDPG models

A summary of the differences between the PPO and DDPG algorithms are presented in Section 2.5.4. The training net profit, fill rate and reward graphs are only presented for the PPO (Figure 52) and DDPG algorithms (Figure 53) in scenario 1 as the graphs followed a similar trend in scenario 2 which was included in Appendix F. The PPO model was simpler to perform with less complex and sensitive hyperparameters. Both PPO and DDPG were able to learn a reasonable policy as the training net profit, fill rate and reward graphs showed convergent behaviour. However, the PPO model was more stable as the graphs fluctuated less compared to the DDPG model. The PPO model provided a better balance between the net profit and fill rate whereas the DDPG model exploited the environment by further increasing the net profit at the expense of a reduction in the fill rate. This showed that the PPO model provided better customer accommodation whereas the DDPG model prioritized the profitability of the supply chain. Large fluctuations in the reward graphs indicated that the reward function was very sensitive which can be optimized further to potentially improve the model's performance.


(a) (b) (c)

Figure 52: The training (a) net profit, (b) fill rate and (c) reward graphs of the PPO models.


(a) (b) (c)

Figure 53: The training (a) net profit, (b) fill rate and (c) reward graphs of the DDPG model.

Even though the DDPG model was more sample efficient as it quickly learned a good policy with less timesteps (interactions within the environment), it was much more time and resource intensive compared to the PPO model. The off-policy nature of the DDPG algorithm was shown to provide enhanced sample efficiency in this particular supply chain. The DDPG algorithm's abilities of better handling continuous action spaces and enhanced exploration capabilities through noise addition was evident as it was able to explore the action space more effectively to achieve a higher net profit compared to the PPO model in both scenarios. It was interesting to note that the deterministic policy (suited for tasks with precise control) of the DDPG algorithm performed better than the stochastic policy of the PPO algorithm for the complex supply chain in this study. Stochastic policies usually perform better in highly uncertain environments but also require large amounts of input data and training time to converge to an optimal policy which could have been a limitation in this study.

### 4.8 Comparing the heuristic and RL model results with literature

Literature sources in the field of inventory management simulated unique supply chains with distinct operations and parameters. Additionally, most literature sources did not provide their final values for example, the overall net profit and fill rate achieved. This made it difficult to quantitatively compare this research with prior works. However, certain qualitative comparisons were made. RL showed potential to improve on traditional inventory control methods by being able to simulate real and complex supply chain data using less assumptions [60]. The RL model learned a unique ordering strategy (when and how much stock to order) rather than using a fixed or general inventory management strategy. This helps maximize the specific goal in a unique supply chain. Past research indicated that Q-learning struggled to converge to an optimal policy. However, the DRL algorithms in this research showed abilities of converging to a reasonable policy. Literature showed that the PPO algorithm had provided improved results (more stable and convergent behaviour) compared to

other RL algorithms and benchmark models. However, this research showed that the DDPG could outperform the PPO algorithm by learning aspects of the supply chain more effectively such as finding profitable locations and minimizing obsolete inventory.

The research by Geevers et al. [73] most closely aligned with this research as real historic demand, a multi-echelon supply chain, the PPO algorithm and a custom RL environment was used. A continuous action space representing the ordering quantity was used in both researches but the model by Geevers et al. struggled to learn the interval of the action space effectively. The PPO model in this research learned to order either 0 units or the maximum ordering quantity at each outlet whereas the DDPG model learned to order a constant amount but only at certain outlets to save on delivery costs. However, the RL models also struggled to vary the actions in this study resulting in large amounts of inventory stored (storage costs) and obsolete inventory at various locations.

The state space in previous research papers varied based on the amount of supply chain information that was available. The research by Geevers et al. focused solely on reducing inventory costs hence, the holding and backorder costs were only included in the reward function. This research aimed to simultaneously maximize net profit and fill rate in the supply chain without considering backorders. Thus, the reward function was based on a combination of profit per timestep, fill rate, units available on hand and units unsatisfied with each contributing towards improved performance. The reward function of Geevers et al. was also tested in this research using the holding costs and the units unsatisfied instead of the backorder costs but it was not able to provide better results in this particular supply chain. However, both researches showed that it was more useful to minimize a negative reward.

Both researches incorporated the available inventory and the inventory in transit at each point in the supply chain. However, Geevers et al. further included the total backorders and total inventory in the supply chain possibly due to a multi-agent analysis being used. This research was based on a single-agent analysis and it was found that it was not useful to include the total inventory in the supply chain into the state space. The inclusion of the number of trucks in operation and the number of units in transit between various locations in this supply chain provided improved performance as it helped the model understand how best to utilize the trucks. Both researches showed that it was more useful to use current variables rather than cumulative variables to prevent an infinite state space.

Most literature sources were able to outperform their respective benchmark models as shown in Table 20. However, the RL models in this research could not outperform the benchmark heuristic in the original supply chain. This could possibly be due to the highly stochastic nature of the demand

dataset, simulation of a finite time horizon, the reduction in the minimum production limit and the complexity of the supply chain network structure. However, RL had shown potential by improving on the heuristic when a modification to the supply chain structure was simulated. The DDPG algorithm in scenario 2 was able to achieve a 21% higher net profit compared to the benchmark heuristic.

Table 20: Comparison of RL models with their respective benchmarks in different research papers.

| **Research** | **% improvement of the best RL model over their respective benchmark model** |
|---|---|
| Gijsbrechts et al. | 9 |
| Hubbs et al. | 11 |
| Xie et al. | 18 |
| Oroojloojadid et al. | 13 |
| Geevers et al. | 20 |
| This research (DDPG model in scenario 2) | 21 |

# 5 Conclusion and Recommendations

Reinforcement learning has provided an opportunity to learn an ordering strategy to maximize the net profit and fill rate in a multi-echelon supply chain with stochastic constraints. The DDPG algorithm has shown improved capabilities of handling an inventory management environment compared to the PPO algorithm by achieving a higher net profit which could potentially be due to DDPG's enhanced exploration capabilities and increased sensitivity to the reward function. The RL models could not outperform the benchmark heuristic when simulating the supply chain of Company X in its current state as the heuristic model achieved a 17% and 8% increase in net profit compared to the PPO and DDPG model respectively. However, RL was able to find that the merged supply chain improved performance as the DDPG model provided a 21% increase in net profit over the benchmark heuristic. RL has shown to help identify root causes and challenges in the supply chain which provide useful information for further areas of improvements in the future. Different modifications of the supply chain can also be tested to potentially further improve the net profit and fill rate. The performance of the RL models were also very sensitive to the reward function which can be further optimized in the future.

## 5.1 Research insights

The RL model provided certain value-adds for Company X including:

- Product integrity (no expired stock is stored).
- Visibility of inventory across the supply chain (transparency).
- Improved asset utilization (production, transportation, inventory).
- Minimal stock outs.
- Minimal wastage of inventory.
- Balancing inventory investment with the return on investment.
- Reducing manual stock-taking and assisting with supply chain decision making.
- Improving the inventory turnover by reducing holding costs.

A summary of useful insights obtained from the RL model include:

- Shipment consolidation is preferred (ensuring trucks are mostly full when in operation).
- The truck and storage capacity were excessive but this may be due to the company having other products in their supply chain.
- The minimum production limit was found to be excessive.
- The production storage costs were too high resulting in less units being available and a reduced overall fill rate in the supply chain.

- Increasing the number of training timesteps improved the performance up to a certain point after which the performance remained similar.
- There was an optimal fill rate that provided the highest net profit. Further increases or decreases in the fill rate resulted in a loss of profit.
- The input supply chain data consisted of decimals which could have added additional complexity to the RL model. The RL model was tested by removing the decimals but the performance remained similar.

### 5.2 Challenges/Limitations

Some challenges that were faced in the RL models of this study include:

- The stock points in the supply chain were ordering in minute quantities resulting in insufficient inventory being available to satisfy customer demand. However, as the amount of training increased, the models learned to order in larger quantities over time.
- The computational time to run the RL models were large however, multiprocessing was used to speed up the training process.
- Using a continuous action space, the actions are normalized to a value between 0 and 1. Thus, the actions needed to be scaled up to provide a realistic ordering quantity for the supply chain. Different methods were used to scale up the action space but using a unique value for each stock point provided the best performance.
- Since the current profit was included in the reward function, the stock points decided to order the maximum action quantity most of the time to save on delivery costs. However, upstream facilities do not always have sufficient units to satisfy large orders which can delay the process of units arriving at certain locations.
- The RL models were sensitive to multiple parameters simultaneously making it challenging to optimize their overall performance.
- The RL models struggled to learn an adaptive ordering strategy with varying actions which resulted in excess units being available and large amounts of obsolete inventory occurring at various locations.
- The societal impact and customer perception of the company as a result of the lower fill rate in the RL models were not considered as it is dependent on many external factors such as ethical considerations and the control of shareholders which is beyond the scope of this study.
- RL was able to perform reasonably well when simulating the supply chain in short term but there still needs to be an investigation as to how well it can perform in the long term and directly within the ERP system of the company before it can be applied in reality. However,

the RL models helped determine root causes (for example, larger production storage costs led to less units being stored) for the way certain operations take place in the current supply chain.

### 5.3 Recommendations

Further research and experimentation based on this study that can be conducted in the future include:

- A larger input dataset (historic demand data) can potentially improve the forecasting accuracy thereby helping the RL models to learn more realistic trends. RNN's can also be tested in the future using more powerful computers with an increased number of training timesteps to potentially provide more accurate time-series predictions.
- Additional products in the supply chain of Company X can be incorporated into the RL models to determine whether it can further improve their profitability.
- The performance of the models can be sensitive to the sequence of events in the supply chain for example, less units may be available to satisfy the final outlet in the simulation. It is very difficult to simulate the supply chain exactly like real life where events happen at the same time ("no model is perfect but some are useful"). It can be useful to shuffle the order in which inventory is sent to the outlets to determine whether it makes a difference. Future extensions of how components of the supply chain can be modelled to operate synchronously can be researched.
- Test new and more advanced RL algorithms to determine whether they are able to capture the stochastic trends in the supply chain more effectively. The reward function can also be further optimized to possibly increase the net profit and fill rate in the future.
- Previous research of RL in different applications have reduced the learning rate parameter with each time step as the model trains. It can be useful to test a similar approach in the RL models of this study to determine whether the RL models can learn an improved policy.
- The choice of ordering from the warehouse or production for each outlet can be included in the action space to determine whether the model can effectively learn where to order from to provide better overall supply chain efficiency.
- Regularization and random sampling techniques introduce variations and transformations to the training data. This can be applied to the RL models in the future to determine whether it can provide a more efficient training process by generalizing better to different scenarios.
- Changes to the supply chain structure, pricing and distribution of goods can be tested in the RL models to advise certain business decisions based on current trends.

### 5.4 Data availability

The complete code for the heuristic and RL models is available on the following GitHub page: arnavgarg25/InventoryManagement (github.com).

## References

[1] D. J. Bowersox, D. J. Closs, J. C. Bowersox and M. B. Cooper, Supply chain logistics management, New York: McGraw-Hill Education, 2020.

[2] D. Vaidya, "What is bullwhip effect?," WallStreetMojo, [Online].
Available: https://www.wallstreetmojo.com/bullwhip-effect/. [Accessed 21 April 2023].

[3] "Nine common inventory control problems and solutions," Mindover Software, [Online].
Available: https://mindovercorp.com/9-common-inventory-control-problems-and-solutions.
[Accessed 21 April 2021].

[4] A. Eira, "Fourteen supply chain trends for 2022/2023," FinancesOnline, 15 April 2023. [Online].
Available: https://financesonline.com/supply-chain-trends/. [Accessed 21 April 2023].

[5] "Poor inventory management: what's causing it and how to stop it," Cin7 Core, 6 January 2018. [Online].
Available: https://dearsystems.com/poor-inventory-management/. [Accessed 21 April 2023].

[6] "Six common inventory problems and how to deal with them," Etaily, 15 April 2022. [Online].
Available: https://www.etaily.com/insights/common-inventory-problems/. [Accessed 21 April 2023].

[7] A. Lim, "Heuristics: The Psychology of Mental Shortcuts," ThoughtCo., 9 November 2018. [Online]. Available: https://www.thoughtco.com/heuristics-psychology-4171769. [Accessed 27 July 2023].

[8] S. Mirshekarian and D. Sormaz, "Machine Learning Approaches to Learning Heuristics for Combinatorial Optimization Problems," *ScienceDirect,* vol. 17, no. 28, pp. 102-109, 2018.

[9] J. Hinchman, et al. "Top 10 Supply Chain Trends," Association for Supply Chain Management, 2024. [Online]. Available: https://www.ascm.org/globalassets/ascm_website_assets/docs/top-10-trends-report-2024.pdf. [Accessed 24 August 2024].

[10] M. Andrei, "AI can now outperform humans in 5 key cognitive ways," ZME Science, 2 November 2023. [Online]. Available: https://www.zmescience.com/science/news-science/ai-can-now-outperform-humans-in-5-key-cognitive-ways/. [Accessed 21 January 2024].

[11] E. Rennie, "Supply Chain Trends 2024: From AI, Data to Resilience," Association for supply chain management, 12 June 2023. [Online]. Available: https://www.ascm.org/ascm-insights/ascms-top-10-supply-chain-trends-in-2024/. [Accessed 21 January 2024].

[12] "Supervised Learning vs Unsupervised Learning vs Reinforcement Learning," IntelliPaat, 12 September 2023. [Online]. Available: https://intellipaat.com/blog/supervised-learning-vs-unsupervised-learning-vs-reinforcement-learning/. [Accessed 21 January 2024].

[13] F. Fourati, V. Aggarwal and M.-S. Alouini, "Stochastic Q-learning for Large Discrete Action Spaces," *arXiv*, pp. 1-26, 2024.
[14] R. Khandelwal, "Supervised, Unsupervised, and Reinforcement Learning," Medium, 20 July 2022. [Online]. Available: https://arshren.medium.com/supervised-unsupervised-and-reinforcement-learning-245b59709f68. [Accessed 21 January 2024].
[15] D. Mwiti, "10 Real-Life Applications of Reinforcement Learning," neptune.ai, 1 September 2023. [Online]. Available: https://neptune.ai/blog/reinforcement-learning-applications. [Accessed 21 January 2024].
[16] D. Unzueta, "AlphaGo: How AI Mastered the Game of Go," Medium, 29 August 2022. [Online]. Available: https://towardsdatascience.com/alphago-how-ai-mastered-the-game-of-go-b1355937c98d. [Accessed 21 January 2024].
[17] Hari, "Supervised Learning," tadalytics, 21 September 2014. [Online]. Available: https://tadalytics.com/2014/09/21/supervised-learning/. [Accessed 21 January 2024].
[18] D. Varghese, "Comparative Study on Classic Machine learning Algorithms," *Medium,* 2018.
[19] "Linear Regression in Machine learning," GeeksforGeeks, 21 December 2023. [Online]. Available: https://www.geeksforgeeks.org/ml-linear-regression/. [Accessed 21 January 2024].
[20] P. Rishith, "Understanding K-Nearest Neighbors: A Simple Approach to Classification and Regression," Towards AI, 2 June 2023. [Online]. Available: https://towardsai.net/p/machine-learning/understanding-k-nearest-neighbors-a-simple-approach-to-classification-and-regression. [Accessed 21 January 2024].
[21] V. Lavrenko, "kNN.8 Nearest-neighbor regression example," YouTube, 15 September 2015. [Online]. Available: https://www.youtube.com/watch?v=3lp5CmSwrHI. [Accessed 21 January 2024].
[22] J. Howard, "Lesson 6 - Random Forest Deep Dive," dslectures, 16 June 2021. [Online]. Available: https://lvwerra.github.io/dslectures/lesson06_random-forest-deep-dive.html. [Accessed 21 January 2024].
[23] "Support Vector Regression," TheClickReader, [Online]. Available: https://www.theclickreader.com/support-vector-regression/. [Accessed 21 January 2024].
[24] J. Brownlee, "Regression Metrics for Machine Learning," Machine Learning Mastery, 16 February 2021. [Online]. Available: https://machinelearningmastery.com/regression-metrics-for-machine-learning/. [Accessed 21 January 2024].
[25] A. M. Bastin, "Cross-validation and hyperparameter tuning," Data Science Central, 14 October 2020. [Online]. Available: https://www.datasciencecentral.com/model-evaluation-model-selection-and-algorithm-selection-in/. [Accessed 21 January 2024].

[26] J. Bergstra and Y. Bengio , "Random Search for Hyper-Parameter Optimization," Journal of Machine Learning Research, vol. 13, pp. 281-305, 2 December 2012.
[27] S. Bhatt, "Reinforcement Learning 101," Medium, 19 March 2018. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292. [Accessed 21 January 2024].
[28] "Part 1: Key Concepts in RL," OpenAI, 2018. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html. [Accessed 21 January 2024].
[29] J. Eschmann, "Reward Function Design in Reinforcement Learning," *Springer Link*, vol. 883, pp. 25-33, 2021.
[30] A. d. S. Mignon and R. L. d. A. da Rocha , "An Adaptive Implementation of ε-Greedy in Reinforcement Learning," Procedia Computer Science, vol. 109, pp. 11146-1151, 2017.
[31] K. S. Ganesh, "What's The Role Of Weights And Bias In a Neural Network?," Medium, 24 July 2020. [Online]. Available: https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f. [Accessed 21 January 2024].
[32] "What are neural networks?," IBM, [Online]. Available: https://www.ibm.com/topics/neural-networks. [Accessed 21 January 2024].
[33] akruyhau, "Bellman Equation," GeeksforGeeks, 27 September 2021. [Online]. Available: https://www.geeksforgeeks.org/bellman-equation/. [Accessed 21 January 2024].
[34] Y. Li, Z. Zheng and W. Zheng, "Deep Robust Reinforcement Learning for Practical Algorithmic Trading," IEEE explore, 5 August 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8786132. [Accessed 31 October 2022].
[35] D. Soper, "Foundations of Q-Learning," YouTube, 23 April 2020. [Online]. Available: https://www.youtube.com/watch?v=__t2XRxXGxI. [Accessed 31 October 2022].
[36] Z. Golabi, "What are the advantages and disadvantages of value methods vs policy methods for Learning in reinforcement Learning (RL)?," Quora, [Online]. Available: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-value-methods-vs-policy-methods-for-Learning-in-reinforcement-Learning-RL. [Accessed 31 October 2022].
[37] Y. Li, W. Zheng and Z. Zheng, "Deep Robust Reinforcement Learning for Practical Algorithmic Trading," *IEEE*, vol. 7, pp. 108014 - 108022, 2019.
[38] I. G. B. Petrazzini and E. A. Antonelo, "Proximal Policy Optimization with continuous bounded action space via the Beta Distribution," *Arxiv*, vol. 1, pp. 1-8, 2021.
[39] J. Schulman, F. Wolski, P. Dhariwal, . A. Radford and . O. Klimov, "Proximal Policy Optimization Algorithms," *Arxiv*, vol. 2, pp. 1-12, 2017.
[40] V. Mnih, K. Kavukcuoglu, D. Silver, et al, "Human-level control through deep reinforcement learning," *nature*, vol. 518, pp. 529-533, 2015.

[41] M. Leeburn , "Reinforcement Learning Algorithms in TORCS," University of the Witwatersrand , Johannesburg, 2021.
[42] "PPO Hyperparameters and Ranges," Medium, 25 July 2018. [Online]. Available: https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe. [Accessed 21 January 2024].
[43] "Deep Deterministic Policy Gradient," OpenAI, [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/ddpg.html. [Accessed 21 January 2024].
[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and . D. Wierstra, "Continuous control with deep reinforcement learning," *Arxiv*, vol. 6, pp. 1-14, 2019.
[45] "Twin Delayed DDPG (TD3): Theory," Saasha Nair, [Online]. Available: https://saashanair.com/blog/blog-posts/twin-delayed-ddpg-td3-how-does-the-algorithm-work. [Accessed 21 January 2024].
[46] H. Singh, "Deep Deterministic Policy Gradient (DDPG)," Keras, 21 September 2020. [Online]. Available: https://keras.io/examples/rl/ddpg_pendulum/. [Accessed 21 January 2024].
[47] N. M. Ashraf, . R. R. Mostafa, . R. H. Sakr and M. Z. Rashad, "Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm," *National Center for Biotechnology Information*, vol. 16, pp. 1-6, 2021.
[48] "Reinforcement Learning Agents," MathWorks, [Online]. Available: https://ww2.mathworks.cn/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html. [Accessed 21 January 2024].
[49] S. Siboo, A. Bhattacharyya, R. N. Raj and S. H. Ashwin, "An Empirical Study of DDPG and PPO-Based Reinforcement Learning Algorithms for Autonomous Driving," *IEEE Xplore*, vol. 11, pp. 125094 - 125108, 2023.
[50] M. Andrychowicz, et al. "What Matters In On-Policy Reinforcement Learning?," *arXiv*, pp. 1-48, 2020.
[51] David, "Do we use validation and test sets for training a reinforcement learning agent?," StackExchange, 7 December 2023. [Online]. Available: https://ai.stackexchange.com/questions/32497/do-we-use-validation-and-test-sets-for-training-a-reinforcement-learning agent#:~:text=No%2C%20we%20typically%20don%27t%20use%20a%20validation%2Ftest%20data,algorithms%20don%27t%20even%20have%20a%20data-set%20a. [Accessed 21 January 2024].
[52] "What is Python?," w3schools, [Online]. Available: https://www.w3schools.com/python/python_intro.asp. [Accessed 31 October 2022].

[53] "The Importance of Libraries in Python, data science, and the Applications They Facilitate!," gopract, [Online]. Available: https://gopract.com/Pages/Importance-of-Libraries-Python-data-science-Applications-They-Facilitate.aspx#:~:text=Python%20libraries%20are%20a%20great%20way%20to%20data,their%20models%2C%20regardless%20of%20their%20size%20or%20complexity.. [Accessed 31 October 2022].

[54] "Libraries in Python," geeksforgeeks, 18 October 2021. [Online]. Available: https://www.geeksforgeeks.org/libraries-in-python/. [Accessed 31 October 2022].

[55] D. Lee, "https://www.linkedin.com/pulse/exploring-reinforce-learning-bit-coin-trading-dennis-lee/," LinkedIn, 4 August 2021. [Online]. Available: https://www.linkedin.com/pulse/exploring-reinforce-learning-bit-coin-trading-dennis-lee/. [Accessed 31 October 2022].

[56] A. King, "Create custom gym environments from scratch — A stock market example," Towards Data Science, 10 April 2019. [Online]. Available: https://towardsdatascience.com/creating-a-custom-openai-gym-environment-for-stock-trading-be532be3910e. [Accessed 31 October 2022].

[57] "Policy Networks," Stable baselines 3, [Online]. Available: https://stable-baselines.readthedocs.io/en/master/modules/policies.html. [Accessed 31 October 2022].

[58] D. Karunakaran, "The Actor-Critic Reinforcement Learning algorithm," Artificial Intelligence, 30 September 2022. [Online]. Available: https://medium.com/intro-to-artificial-intelligence/the-actor-critic-reinforcement-learning-algorithm-c8095a655c14. [Accessed 31 October 2022].

[59] A. Hayes, "Inventory Management Defined, Plus Methods and Techniques," Investopedia, 28 March 2023. [Online]. Available: https://www.investopedia.com/terms/i/inventory-management.asp. [Accessed 30 June 2023].

[60] B. Rolf, et al. "A review on reinforcement learning algorithms and applications in supply chain management," International Journal of Production Research, 2022.

[61] Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4 (9), 237–285. doi:10.1613/jair.301

[62] S. K. Chaharsooghi, et al. "A reinforcement learning model for supply chain ordering management: An application to the beer game," ScienceDirect, vol. 45, no. 4, pp. 949-959, 2008.

[63] M. Çimen, et al. "Approximate dynamic programming algorithms for multidimensional flexible production-inventory problems," International Journal of Production Research, vol. 55, no. 7, pp. 2034-2050, 2016.

[64] K. K. Ravulapati, et al. "A simulation-based approach to study stochastic inventory-planning games," International Journal of Systems Science, vol. 34, no. 13, pp. 717-730, 2010.

[65] J. Rao, et al. "A reinforcement learning approach to stochastic business games," IIE Transactions, vol. 36, no. 4, pp. 373-385, 2010.

[66] Z. Sui, et al. "A Reinforcement Learning Approach for Inventory Replenishment in Vendor-Managed Inventory Systems With Consignment Inventory," Engineering Management Journal, vol. 22, no. 4, pp. 44-53, 2015.
[67] T. v. Tongeren, et al. "Q-learning in a competitive supply chain," IEEE Xplore, pp. 1211-1216, 2008.
[68] S. Yang, et al. "Adaptive inventory control and bullwhip effect analysis for supply chains with non-stationary demand," IEEE Xplore, pp. 3903-3908, 2015.
[69] J. Gijsbrechts, et al. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Dual Sourcing, Lost Sales and Multi-Echelon Problems," SSRN Electronic Journal, 2019.
[70] C. Hubbs, "How to Improve your Supply Chain with Deep Reinforcement Learning," TowardsDataScience, 8 October 2020. [Online].
Available: https://towardsdatascience.com/deep-reinforcement-learning-for-supply-chain-optimization-3e4d99ad4b58. [Accessed 21 April 2023].
[71] G. Xie, "Reinforcement Learning for Inventory Optimization Series II: An RL Model for A Multi-Echelon Network," TowardsDataScience, 30 December 2022. [Online].
Available: https://towardsdatascience.com/reinforcement-learning-for-inventory-optimization-series-ii-an-rl-model-for-a-multi-echelon-921857acdb00. [Accessed 21 April 2023].
[72] A. Oroojlooyjadid, et al. "A Deep Q-Network for the Beer Game: Deep reinforcement learning for inventory optimization," Manufacturing and service operations management, 2020.
[73] K. Geevers, "Deep reinforcement learning in inventory management," University of Twente, December 2020. [Online].
Available: https://essay.utwente.nl/85432/1/Geevers_MA_BMS.pdf. [Accessed 21 April 2023].
[74] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," Springer Nature, vol. 110, pp. 2419-2468, 2021.
[75] "Meaning of Inventory Management," Pinterest, [Online]. Available: https://in.pinterest.com/pin/management-assignment-help-and-writing-service-in-australia--714594665892954028/. [Accessed 21 January 2024].
[76] J. Brownlee , "A Gentle Introduction to Autocorrelation and Partial Autocorrelation," Machine learning mastery, 14 August 2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/. [Accessed 27 July 2023].
[77] M. Auhl, "What is an ARIMA Model?," TowardsDataScience, 6 August 2021. [Online].
Available: https://towardsdatascience.com/what-is-an-arima-model-9e200f06f9eb. [Accessed 28 July 2023].

[78] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," Springer Nature, vol. 110, pp. 2419-2468, 2021.

# Appendix

## A. Relevant supply chain parameters and their values of Company X

Table A.1 General parameters of the supply chain

| Parameter | Value |
|---|---|
| Small truck capacity (operating between the production and warehouse facility) | 14 000 units |
| Large truck capacity (operating from the production/warehouse to each outlet) | 58 800 units |
| Product lifespan | 60 days |
| Selling price | R25 |
| 1 pallet | 70 cases |
| 1 case | 30 units |

Table A.2 Parameters primarily related to the production facility of the supply chain

| Parameter | Value |
|---|---|
| Production processing time | 2 days |
| Manufacture cost per pallet | R25 000 |
| Minimum production limit | 30 000 units (5 000 units used in this research) |
| Maximum production limit | 150 000 units |

Table A.3 Parameters related to storage for each stock point of the supply chain

| Location | Storage capacity (units) | Storage cost per pallet per week (Rands) |
|---|---|---|
| Cape Town | 745 500 | 65.748 |
| Pretoria | 3 990 000 | 65.748 |
| Port Elizabeth | 987 000 | 65.748 |
| Warehouse (Johannesburg) | 14 175 000 | 54.79 |
| Durban | 630 000 | 65.748 |
| Bloemfontein | 1 268 400 | 65.748 |
| Production (Johannesburg) | 3 175 200 | 224.29 |

Table A.4 Parameters related to transport for each stock point of the supply chain

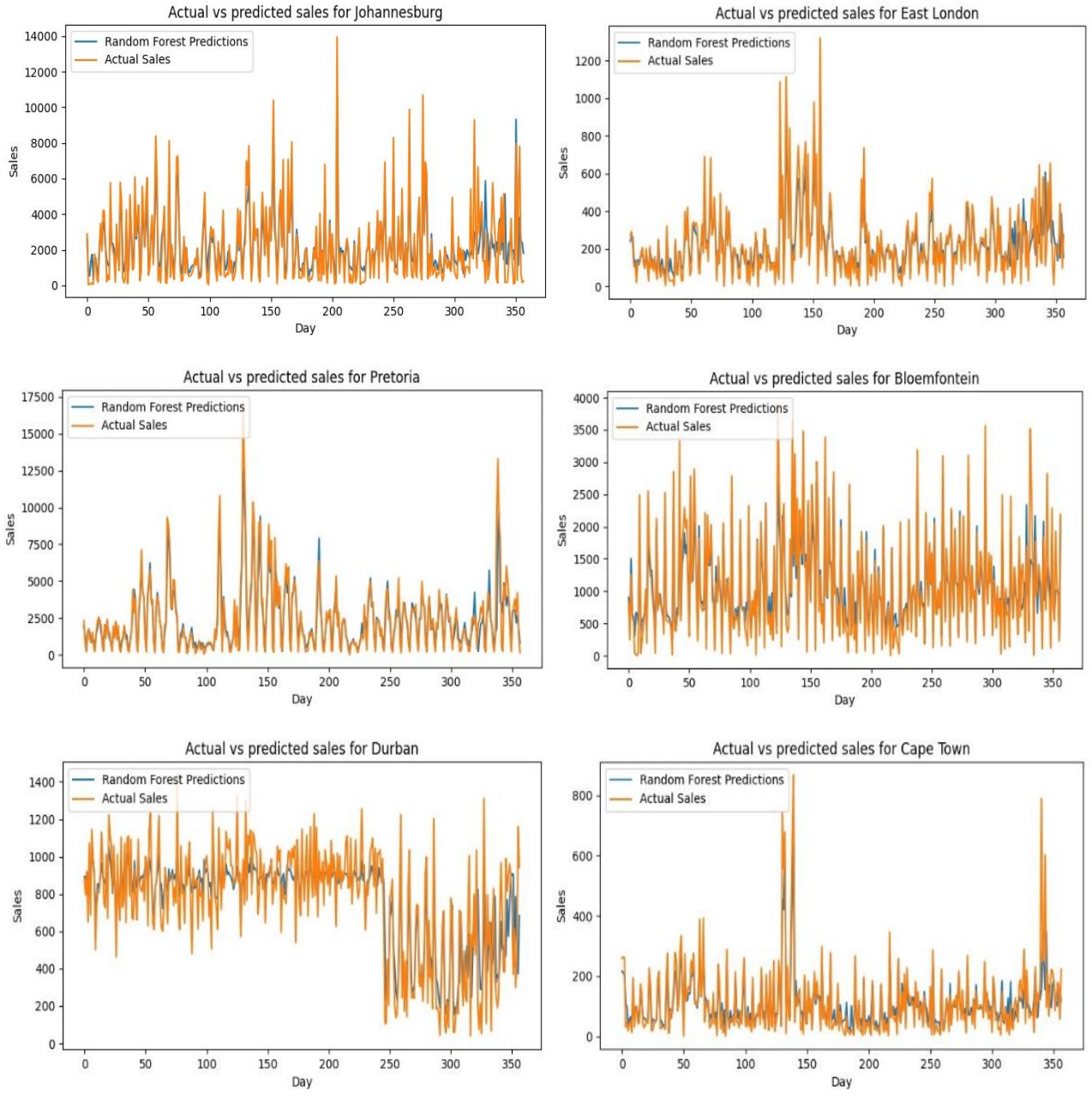
| Location | Transport time from Production (days) | Transport time from Warehouse (days) | Transport cost from Production (Rands) | Transport cost from Warehouse (Rands) |
|---|---|---|---|---|
| Cape Town | 3 | 3 | 32 701.39211 | 32 701.39211 |
| Pretoria | 1 | 1 | 5 031.942839 | 8 739.340276 |
| Port Elizabeth | 3 | 3 | 27 968.98425 | 27 968.98425 |
| Warehouse (Johannesburg) | 1 | 0 | 5 031.942839 | 0 |
| Durban | 2 | 2 | 14 042.60801 | 14 042.60801 |
| Bloemfontein | 2 | 2 | 16 116.92355 | 16 116.92355 |

Table A.5 Parameters related to the probability of sending units from either the production or warehouse to each outlet in the supply chain

| Location | % distribution from Production | % distribution from Warehouse |
|---|---|---|
| Cape Town | 75 | 25 |
| Pretoria | 46 | 54 |

| | | |
|---|---|---|
| Port Elizabeth | 7 | 93 |
| Durban | 5 | 95 |
| Bloemfontein | 18 | 82 |

## B. Actual vs predicted sales at each of the outlets




## C. Variables excluded in the state space

| Variables that provided poor/no improvement in performance |
|---|
| Fill rate |
| Current profit |
| Current day |

| |
|---|
| Current units satisfied/unsatisfied |
| Current storage/manufacture/delivery cost |
| Units expiring in 1, 2, 3 days |
| Current revenue gained |

## D. Training charts of the PPO algorithm

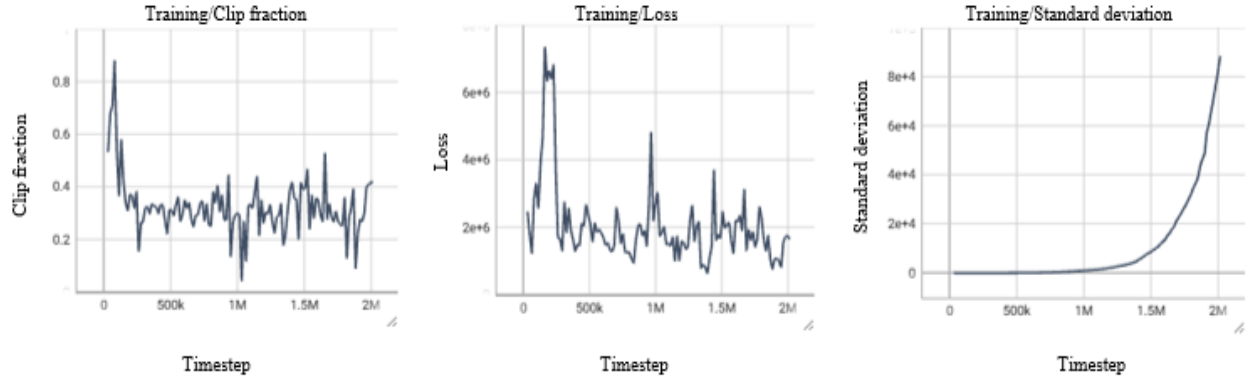
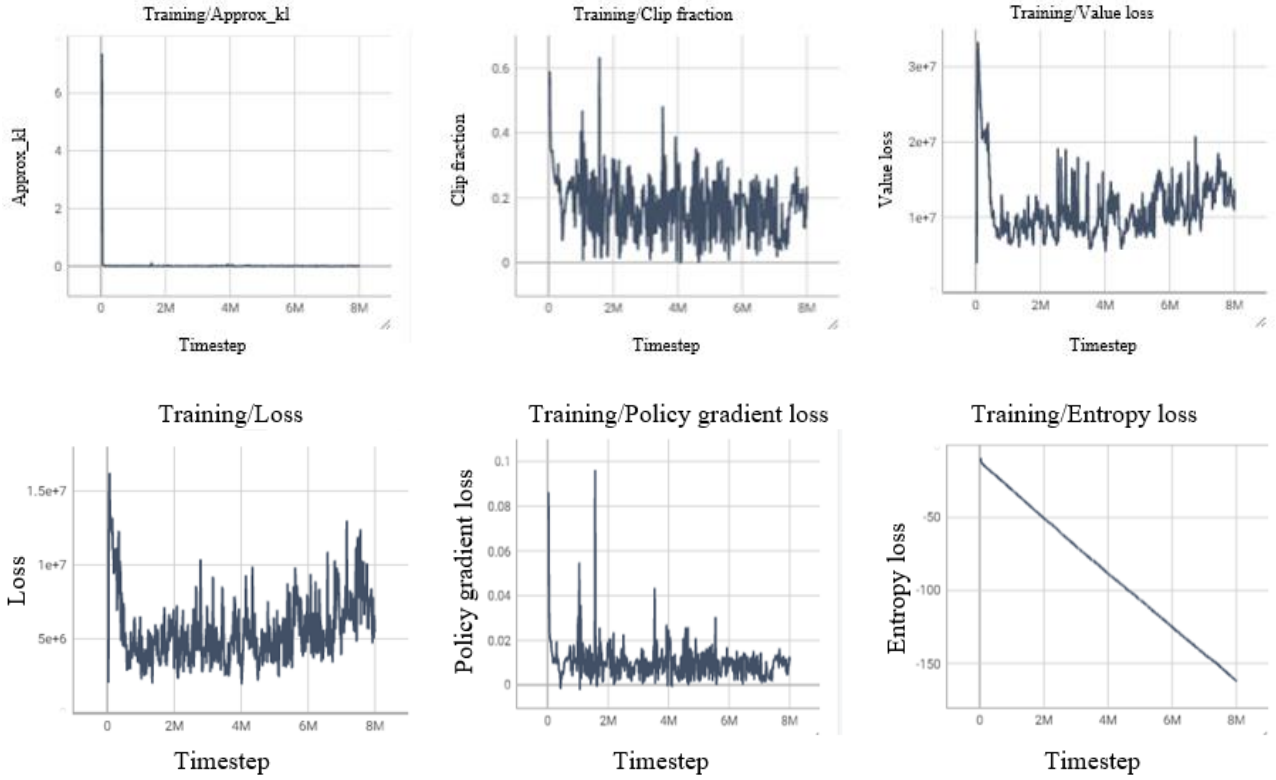



Figure D.1: The remaining PPO training charts in scenario 1



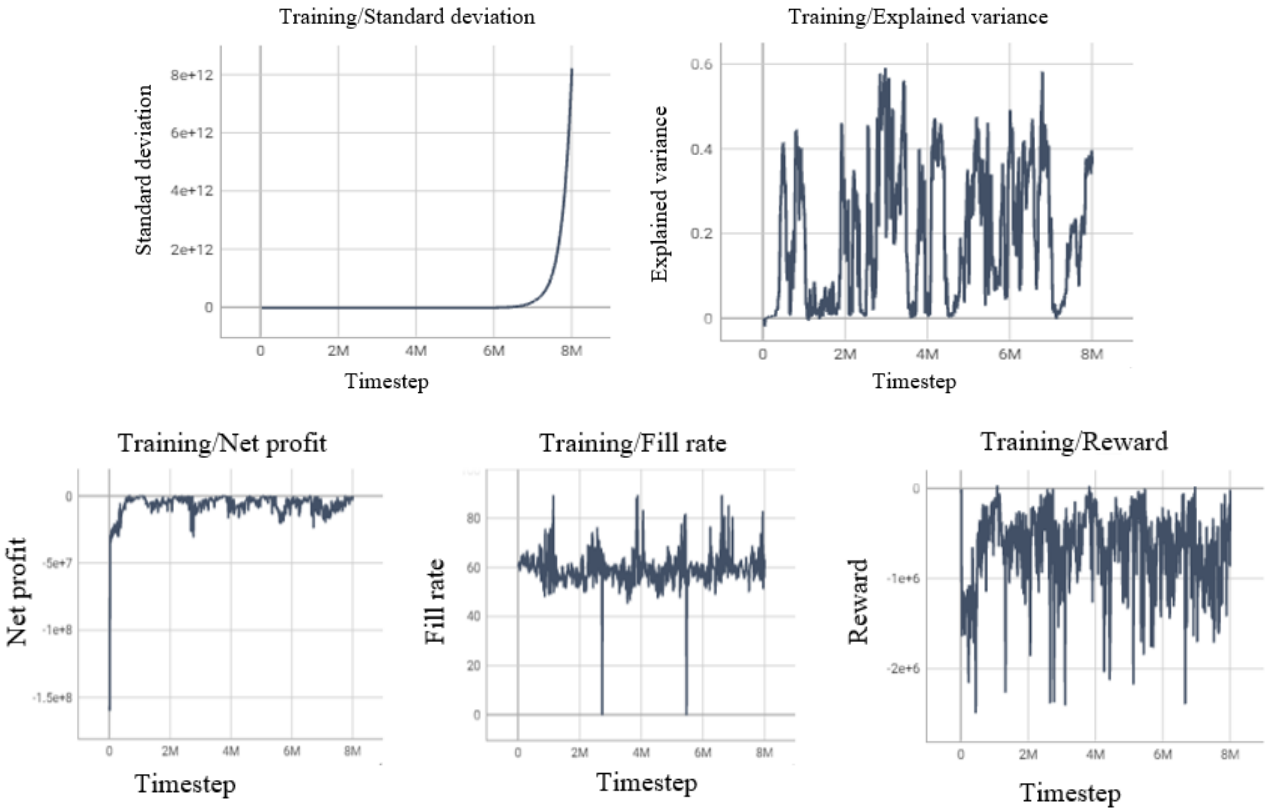



Figure D.2: PPO training charts of scenario 2

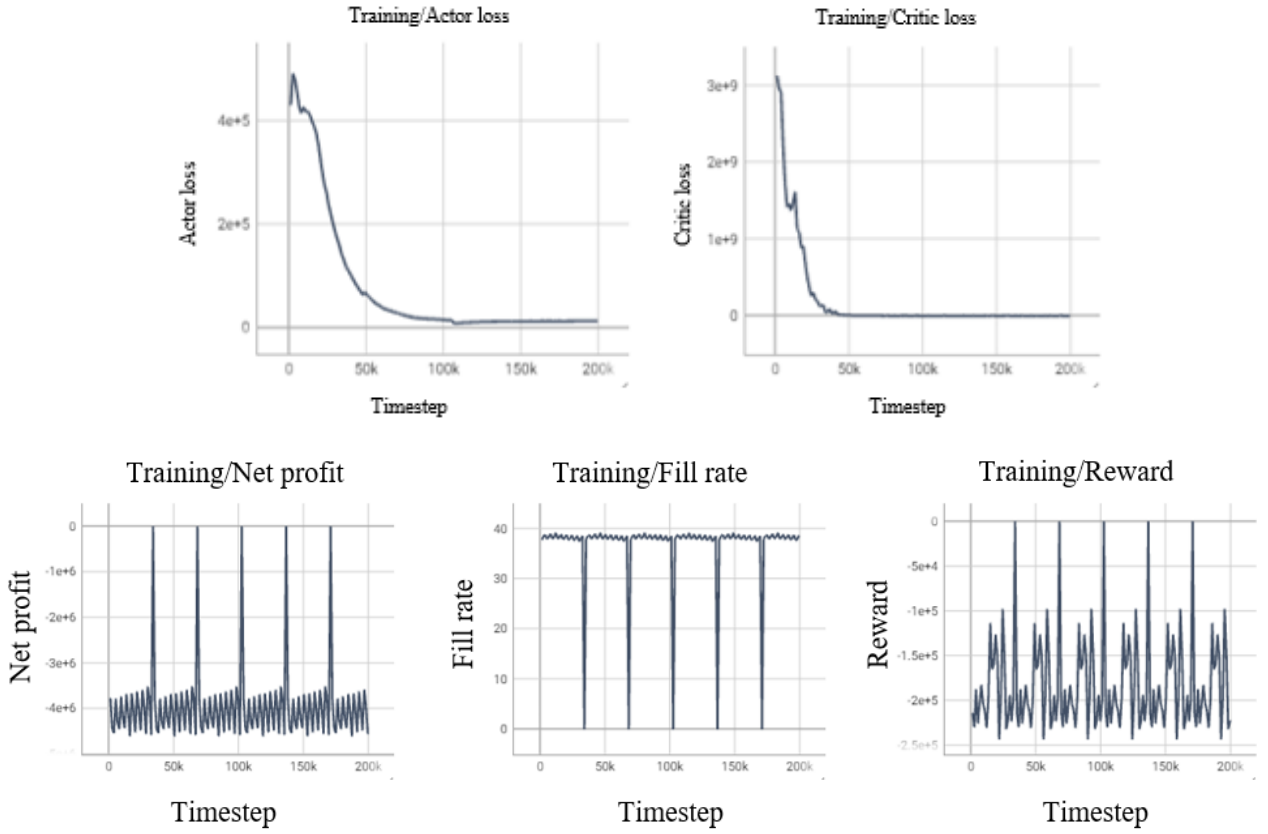



Figure D.3: DDPG training charts of scenario 2