Danmarks Tekniske Universitet

# Machine Learning for Energy Systems Assignment 2

**Authors**

Konstantinos Kountras - s242786
Jonas Wiendl - s243543
Thorri Jokull Thorsteinsson - s242760
Bjartur Jorfi Ingvason - s242754
Kristian Jerichau Nissen - s203895

December 3, 2025

# Contents

## Use of AI

Artificial Intelligence (AI), specifically LLMs, were used in the presented case study to an appropriate degree. Time-consuming tasks like creating a preliminary inclusive document layout of the report were supported by the use of ChatGPT. Furthermore, it assisted in coding scripts, used for data visualization and analysis. Finally, it streamlined the process of formatting of the LaTeX document, helping with efficient generation of tables and figures, and supported the overall writing process by drafting initial ideas for the sections of the report when necessary. It should be noted that all written material still remains an effort of personal and teamwork and is subject to copyright, securing the originality of the work done. When using AI tools, it is extremely important to rigorously verify the correctness of outputs. As expected, some inconsistencies were encountered. This happened most often with coding, where multiple iterations of running the script were needed before expected outputs were achieved.

## Participation Table

| **Student Code** | **Ch. 2 (%)** | **Ch. 3 (%)** | **Ch. 4 (%)** | **Ch. 5 (%)** | **Ch. 6 (%)** |
|---|---|---|---|---|---|
| s242786 | 20% | 20% | 20% | 20% | 20% |
| s243543 | 20% | 20% | 20% | 20% | 20% |
| s242760 | 20% | 20% | 20% | 20% | 20% |
| s242754 | 20% | 20% | 20% | 20% | 20% |
| s203895 | 20% | 20% | 20% | 20% | 20% |
| **Total** | **100%** | **100%** | **100%** | **100%** | **100%** |

Table 1: Participation percentages per task across 5 students. Each column sums to 100%.

## Github Repository

The code used during this assignment can be found in the linked GitHub Repository.

# List of Figures

# List of Tables

# Mathematical notation

The following notation is used consistently throughout the report:

### General

- $t = 1, \dots, T$: Time index (hours).
- $\lambda_t$ (DKK/MWh): Electricity price at hour $t$.
- $E_{\max} = 400$ MWh: Maximum energy capacity of the BESS.
- $P_{\max} = 100$ MW: Maximum charge/discharge power.

### Battery dynamics

- $E_t$ (MWh): State of charge (SOC) at time $t$.
- $P_t \in [-P_{\max}, P_{\max}]$: Charging/discharging power (positive = charging).
- $\Delta(a_t) \in \{+100, 0, -100\}$: SOC change resulting from action $a_t$ (used in the MDP).
- $\pi_t = -\lambda_t P_t$: Instantaneous profit at time $t$.
- $\Pi = \sum_{t=1}^{T} \pi_t$: Total cumulative profit.

### Discrete action variables (Model 1 and Model 2)

- $a_t^{ch}, a_t^{di}, a_t^{id} \in \{0, 1\}$: Binary charge, discharge, idle indicators (optimization model).
- $a_t \in \{\text{charge}, \text{discharge}, \text{idle}\}$: Action at time $t$.

### Features and labels for logistic regression (Model 1)

- $\text{SOC}_t = \frac{E_t}{E_{\max}}$: Normalized state of charge.
- $x_t = [\,\text{SOC}_t,\ \lambda_{t-1}\,]^\top$: Feature vector at time $t$.
- $y_t \in \{0, 1\}$: Binary label (1 = charge, 0 = discharge) after removing idle samples.
- $p_t = P(y_t = 1 \mid x_t)$: Predicted probability of the charge action.
- $\beta_0, \beta_1, \beta_2$: Logistic regression coefficients.
- $\hat{P}$: Probability threshold used to convert $p_t$ to an action.

### Markov Decision Process (Model 2)

- $S$: State space.
- $A(s)$: Feasible action set in state $s$.

- $s_t = (E_t, b_{t-1})$: MDP state (SOC and previous price bin).
- $(q_0, \ldots, q_K)$: Quantile edges used for price discretization.
- $b_t \in \{1, \ldots, K\}$: Price bin at time $t$.
- $P(b_t = j \mid b_{t-1} = i)$: Empirical price transition probabilities.
- $\bar{\lambda}_j$: Mean price within bin $j$.
- $\hat{\lambda}(i)$: Expected price given previous bin $i$ (used in the reward).
- $R(s, a)$: Reward function.
- $\gamma$: Discount factor.
- $V(s)$: Value function.
- $\pi^*(s)$: Optimal policy obtained from value iteration.

# 1 Introduction

This assignment investigates two data-driven control strategies for operating a battery energy storage system (BESS) in a real-time electricity market. The analysis is based on historical wholesale price data from the *DK2* bidding zone and described along the problem setting in 2. The first control approach, presented in Chapter 3, uses supervised learning: an integer optimization model is solved to generate ground-truth action labels, and a logistic regression classifier is trained to predict whether the battery should charge, discharge, or remain idle. The second approach, introduced in Chapter 4, formulates the control problem as a Markov Decision Process (MDP). A discrete state and action representation enables the use of value iteration to compute an optimal stationary policy for real-time operation. Both models aim to maximize cumulative arbitrage profit while respecting the physical constraints of the battery. Finally, Chapter 5 benchmarks the learned policies against the theoretical maximum profit attainable under perfect foresight, allowing a direct comparison between the supervised and reinforcement learning strategies.

# 2 Problem Setting and Data

We consider a battery energy storage system (BESS) operating over a discrete hourly time horizon $t = 1, \dots, T$, driven by the observed day-ahead electricity prices $\lambda_t$. The price data consist of hourly wholesale prices from the *DK2* bidding zone spanning nearly 22 months (2021-01-01 to 2023-09-18). These prices constitute the only exogenous input to the control problem and serve as the basis for both training and evaluating the data-driven strategies introduced later.
The BESS has a maximum usable energy capacity $E_{\max}$ and a maximum charge/discharge power $P_{\max}$. The state of charge (SOC) evolves according to

$$E_{t+1} = E_t + P_t \times \Delta t, \tag{1}$$

where $P_t \in [-P_{\max}, , P_{\max}]$ is the applied power, positive for charging and negative for discharging. The SOC is constrained by

$$0 \leq E_t \leq E_{\max}, \qquad \forall, t. \tag{2}$$

The objective is to exploit temporal price fluctuations by charging at low prices and discharging at high prices. The instantaneous profit is

$$\pi_t = -\lambda_t \times P_t, \tag{3}$$

and cumulative arbitrage profit is given by

$$\Pi = \sum_{t=1}^{T} \pi_t. \tag{4}$$

To allow for out-of-sample evaluation, the price series is divided into training, validation, and test periods. Because the learning approaches differ, the exact splitting procedure is described separately in the relevant chapters. In both cases, the test set is based on the last 15 % of the price data to ensure comparability.
For evaluation purposes the battery is assumed to start empty ($E_1 = 0$), but the learning methods introduced in the following chapters operate over the full feasible SOC range. Both the logistic-regression-based controller and the Markov decision process formulation aim to determine a sequence of charge, discharge, and idle decisions that maximizes expected profit while satisfying the BESS physical constraints.

# 3 Logistic Regression Control (Model 1)

At each hour of the aforementioned discretized hourly time horizon, the battery energy storage system (BESS) may *charge*, *discharge*, or remain *idle*. The electricity price at hour $t$ is denoted by $\lambda_t$ (DKK/MWh). The battery has a maximum energy capacity of $E_{\max} = 400\,\text{MWh}$ and a maximum charge/discharge power of $P_{\max} = 100\,\text{MW}$.
The state of charge (SOC) at time $t$ is represented by $E_t$, and the binary variables

$$a_t^{ch}, \quad a_t^{di}, \quad a_t^{id} \in \{0, 1\}$$

indicate whether the battery is charging, discharging, or idle, respectively. Only one of these actions may be selected at each time step.
For the supervised learning model, we use the normalized SOC,

$$SOC_t = \frac{E_t}{E_{\max}},$$

and the electricity price of the previous hour $\lambda_{t-1}$ as input features. The feature vector is therefore defined as

$$x_t = \begin{bmatrix} SOC_t \\ \lambda_{t-1} \end{bmatrix},$$

and the corresponding label $y_t \in \{\text{charge}, \text{discharge}, \text{idle}\}$ is derived from solving the optimization problem described below.

## 3.1 Optimization Model for Generating Action Labels

To obtain the optimal battery actions over the training horizon, we formulate an integer optimization problem that maximizes the cumulative profit. Discharging the battery yields revenue $\lambda_t P_{\max}$, while charging incurs a cost of the same magnitude. The optimization objective is therefore

$$\max_{E_t, a_t^{ch}, a_t^{di}, a_t^{id}} \sum_{t=1}^{T_{\text{train}}} \lambda_t P_{\max} \left( a_t^{di} - a_t^{ch} \right). \tag{5}$$

The SOC evolves according to the energy balance equation

$$E_{t+1} = E_t + P_{\max} a_t^{ch} - P_{\max} a_t^{di}, \quad \forall t = 1, \dots, T_{\text{train}} - 1. \tag{6}$$

The battery must satisfy its physical limits, described in Equation 2. Exactly one action (binary constraint) must be chosen at each hour, enforced by:

$$a_t^{ch} + a_t^{di} + a_t^{id} = 1, \quad \forall t. \tag{7}$$

Solving this mixed-integer linear program yields a unique optimal action at each time step. From these optimal decisions, we define the labels for the logistic regression classifier training.

$$y_t = \begin{cases} \text{charge}, & \text{if } a_t^{ch} = 1, \\ \text{discharge}, & \text{if } a_t^{di} = 1, \\ \text{idle}, & \text{if } a_t^{id} = 1. \end{cases}$$

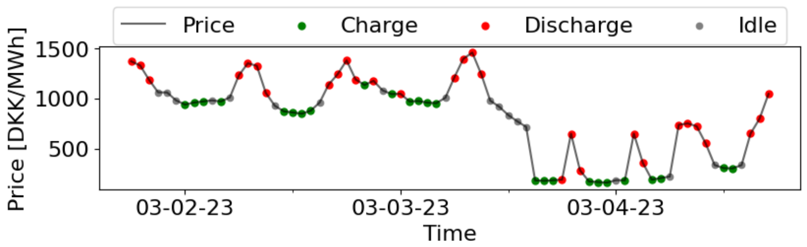


Figure 1: Optimal action chosen by the optimization on the price curve

## 3.2 Logistic Regression Classifier

Using the feature vector $x_t = [SOC_t, \ \lambda_{t-1}]^\top$ and the labels obtained above, we train a logistic regression model to estimate the probability that *charging* is the optimal action at time $t$. The classifier models the conditional probability

$$p_t = P(y_t = \text{charge} \mid x_t) = \sigma(\beta_0 + \beta_1 SOC_t + \beta_2 \lambda_{t-1}) \,. \tag{8}$$

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function, and $\beta = (\beta_0, \beta_1, \beta_2)$ are model parameters learned via maximum likelihood estimation.

$$\mathcal{L}(\beta) = -\sum_{t=1}^{T} [y_t \log p_t + (1 - y_t) \log(1 - p_t)] \tag{9}$$

Since the BESS has three discrete actions, we convert the continuous probability output into one of the feasible actions using a probability threshold $\hat{P}$. The predicted action is chosen as

$$\hat{a}_t = \begin{cases} \text{charge}, & p_t > \hat{P}, \\ \text{discharge}, & p_t < 1 - \hat{P}, \\ \text{idle}, & 1 - \hat{P} \le p_t \le \hat{P}. \end{cases} \tag{10}$$

The threshold $\hat{P}$ is treated as a hyperparameter and is tuned on a validation set by maximizing cumulative profit over the validation horizon. When a predicted action is infeasible due to SOC constraints (e.g., charging at full SOC), we project it onto the feasible action set as an idle action. This is the only way this model can obtain the idle state.
This completes the mathematical formulation of Model 1, consisting of the optimization problem that defines optimal actions and the logistic regression classifier that approximates this optimal policy using only the SOC and the previous hour's electricity price.
During the development of the logistic regression model, we initially followed the assignment formulation in which the supervised labels were defined as 1 for charging and 0 for discharging or idle. However, as pointed out by the course instructor, this labeling scheme leads to a highly unbalanced dataset because the idle action occurs much more frequently than the other two. As a result, the logistic regression model almost never predicted the charging action, with estimated probabilities rarely exceeding 0.5. Since techniques for handling class imbalance (such as class weights or resampling) were not covered in the course, the instructor recommended modifying the training data.
Following this instruction, we redefined the classification problem as a strictly binary task that distinguishes only between charging and discharging, and we removed all samples for which the optimal action is idle. The labels are therefore constructed as

$$y_t = \begin{cases} 1, & a_t^{ch} = 1, \\ 0, & a_t^{di} = 1, \end{cases} \tag{11}$$

and all instances with $a_t^{id} = 1$ are excluded from the training set. This modification results in a much more balanced dataset, allowing the logistic regression model to learn a meaningful decision boundary. Although idle actions are excluded during training, they remain necessary when the controller is applied. Idle is therefore reintroduced only through feasibility constraints: if the classifier predicts discharge when the battery is empty, or predicts charge when the battery is full, the predicted action is projected to the idle state. In this way, idle is not explicitly modeled as a class but arises naturally from the physical limitations of the BESS.
To choose the probability threshold $\hat{P}$ that converts the logistic regression output into an action, we tuned this hyperparameter based on the achieved profit rather than standard classification metrics. Since the data are time-dependent, we used a five-fold time-series cross-validation setup. For every candidate value of $\hat{P}$, we computed the cumulative profit on all validation folds and then averaged these values. The threshold that produced the highest average profit was selected ($\hat{P} = 0.37$) as the optimal one. This approach ensures that the classifier is tuned directly for economic performance, which is the actual objective of the BESS controller.

### 3.2.1 Logistic regression results

Overall, the predictive performance of the classifier remains modest: the model achieves an accuracy of approximately 65% on the test set. When the resulting policy is evaluated in the market simulation, it yields a profit of 5,468,221.85 DKK, which corresponds to 9.72% of the

optimal profit of 56,232,234.84 DKK obtained directly from the full optimization model. We evaluated both a ridge-regularized logistic regression model and an unregularized alternative. Since time-series cross-validation produced nearly identical accuracy scores for the two approaches, we selected the unregularized model to maintain interpretability and preserve the natural scale of the coefficients.
The unregularized logistic regression assigns negative coefficients to both predictors, indicating that higher SOC and higher prices reduce the likelihood of charging. The SOC coefficient is -0.96, while the price coefficient is -0.54. The SOC effect is therefore substantially stronger, roughly 1.8 times the influence of price. This focus on the SOC might stem from the inability to choose the idle value when unsure about the charging.

# 4 Reinforcement Learning Control (Model 2)

This chapter presents a reinforcement learning formulation of the battery arbitrage problem, complementing the supervised-learning approach developed in the previous section.

## 4.1 Markov Decision Process Formulation

The battery arbitrage problem introduced in Chapter 2 is modeled as a finite Markov decision process (MDP)

$$M = (S, A, \{p_{sa}\}, \gamma, R), \tag{12}$$

where $S$ is the state space, $A$ the set of feasible actions, $\{p_{sa}\}$ the transition probabilities, $\gamma \in [0, 1]$ the discount factor, and $R$ the reward function

### State Space

The state at time $t$ is defined as

$$s_t = (E_t,\, b_{t-1}), \tag{13}$$

where $E_t \in \{0, 100, 200, 300, 400\}$ denotes the discretized SOC, and $b_{t-1} \in \{1, \dots, K\}$ the price bin of the previous hour, as the current and future prices are unknown. Let $(q_0, \dots, q_K)$ denote the quantile edges of the training prices $\lambda_t$. Each price is mapped to a bin via

$$b_t = \min\{k : \lambda_t \leq q_k\}. \tag{14}$$

Thus, the finite state space is

$$S = \{0, 100, 200, 300, 400\} \times \{1, \dots, K\}. \tag{15}$$

### Action Space

The feasible action set $A(s)$ depends on the SOC level:

$$A(E_t) = \begin{cases} \{\text{charge}, \text{idle}\}, & E_t = 0, \\ \{\text{charge}, \text{discharge}, \text{idle}\}, & 0 < E_t < 400, \\ \{\text{discharge}, \text{idle}\}, & E_t = 400. \end{cases} \tag{16}$$

The SOC transition is deterministic:

$$E_{t+1} = E_t + \Delta(a_t), \tag{17}$$

where $\Delta(a_t) \in \{+100, 0, -100\}$ for charge, idle, and discharge, respectively.

### Price Transition Probabilities

The uncertainty comes only from the electricity price. The empirical Markov transition probabilities are estimated from training data as

$$P(b_t = j \mid b_{t-1} = i) = \frac{\sum_{t=2}^{T_{\text{train}}} \mathbf{1}\{b_{t-1} = i,\ b_t = j\}}{\sum_{t=2}^{T_{\text{train}}} \mathbf{1}\{b_{t-1} = i\}}. \tag{18}$$

The full transition kernel factorizes as

$$p_{(E,i),a}(E', j) = \mathbf{1}\{E' = E + \Delta(a)\}\ P(b_t = j \mid b_{t-1} = i). \tag{19}$$

### Reward Function

Because the current price bin $b_t$ is not available at decision time, the model uses its conditional expectation, which represents the best available estimate of $\lambda_t$ when the decision is made:

$$\hat{\lambda}(i) = \sum_{j=1}^{K} P(b_t = j \mid b_{t-1} = i)\, \bar{\lambda}_j, \tag{20}$$

where $\bar{\lambda}_j$ is the mean price in bin $j$. The arbitrage reward is defined as

$$R(s, a) = \begin{cases} -\hat{\lambda}(i)\, P_{\max}, & a = \text{charge}, \\ \ \ \hat{\lambda}(i)\, P_{\max}, & a = \text{discharge}, \\ 0, & a = \text{idle}. \end{cases} \tag{21}$$

### Value Iteration

The optimal value function satisfies the Bellman optimality equation

$$V^*(s) = \max_{a \in A(s)} \left[ R(s,a) + \gamma \sum_{s' \in S} p_{sa}(s') \, V^*(s') \right]. \tag{22}$$

Value iteration updates are computed as

$$V_{k+1}(s) = \max_{a \in A(s)} \left[ R(s,a) + \gamma \sum_{s' \in S} p_{sa}(s') \, V_k(s') \right]. \tag{23}$$

Convergence is reached when

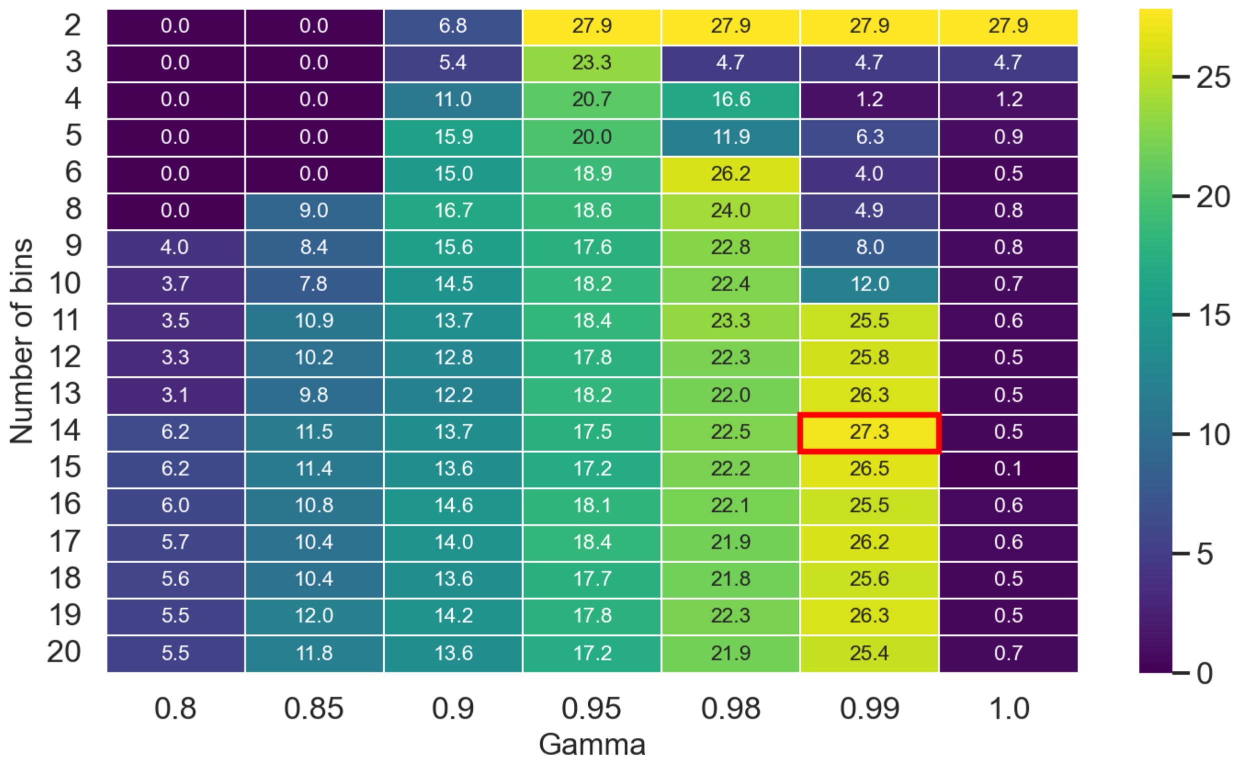
$$\max_{s \in S} |V_{k+1}(s) - V_k(s)| < \varepsilon, \tag{24}$$

with $\varepsilon = 10^{-6}$. The optimal deterministic policy is obtained as

$$\pi^*(s) = \arg\max_{a \in A(s)} \left[ R(s,a) + \gamma \sum_{s'} p_{sa}(s') \, V^*(s') \right]. \tag{25}$$

## 4.2 Hyperparameter Tuning

The discretization of electricity prices into $K$ bins and the choice of discount factor $\gamma$ are not fixed by the assignment. Both parameters directly influence the quality of the MDP approximation: too few bins oversimplify price dynamics, while too many lead to sparse transition counts. Likewise, the discount factor controls how strongly future rewards influence the policy. To select suitable values, a grid search was performed over $K \in \{2, \ldots, 20\}$ and $\gamma \in \{0.8, 0.85, 0.9, 0.95, 0.98, 0.99, 1.0\}$. For each parameter pair, an MDP was constructed from the training data, an optimal policy was obtained via value iteration, and the resulting model was evaluated on the validation set. The objective was to maximize cumulative validation profit. Figure 2 shows the validation profit for all tested configurations.
Performance is poor for very small $K$ when combined with low $\gamma$, but improves as the price discretization becomes more granular. The only exception is $K = 2$, which shows high validation profits for large $\gamma$. However, this behaviour is misleading: with only two bins, the price process collapses into a trivial low vs. high split. Therefore, for a more meaningful structure, higher values of $K$ are required. Among these, the discount factor has a strong effect: values between 0.95 and 0.99 consistently perform best, while $\gamma = 1$ becomes too conservative by overvaluing distant, uncertain rewards. The best reliable configuration is obtained at $K = 14$ bins and $\gamma = 0.99$, yielding a validation profit of 27.3 million DKK. This setting is therefore used to retrain the final RL model on the combined train+validation set before evaluating on the test set.

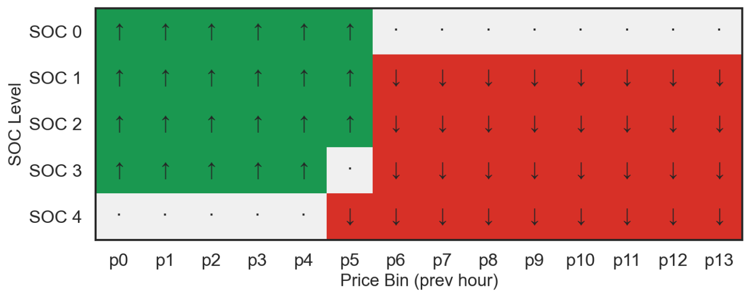


| Number of bins | 0.8 | 0.85 | 0.9 | 0.95 | 0.98 | 0.99 | 1.0 |
|---|---|---|---|---|---|---|---|
| 2 | 0.0 | 0.0 | 6.8 | 27.9 | 27.9 | 27.9 | 27.9 |
| 3 | 0.0 | 0.0 | 5.4 | 23.3 | 4.7 | 4.7 | 4.7 |
| 4 | 0.0 | 0.0 | 11.0 | 20.7 | 16.6 | 1.2 | 1.2 |
| 5 | 0.0 | 0.0 | 15.9 | 20.0 | 11.9 | 6.3 | 0.9 |
| 6 | 0.0 | 0.0 | 15.0 | 18.9 | 26.2 | 4.0 | 0.5 |
| 8 | 0.0 | 9.0 | 16.7 | 18.6 | 24.0 | 4.9 | 0.8 |
| 9 | 4.0 | 8.4 | 15.6 | 17.6 | 22.8 | 8.0 | 0.8 |
| 10 | 3.7 | 7.8 | 14.5 | 18.2 | 22.4 | 12.0 | 0.7 |
| 11 | 3.5 | 10.9 | 13.7 | 18.4 | 23.3 | 25.5 | 0.6 |
| 12 | 3.3 | 10.2 | 12.8 | 17.8 | 22.3 | 25.8 | 0.5 |
| 13 | 3.1 | 9.8 | 12.2 | 18.2 | 22.0 | 26.3 | 0.5 |
| 14 | 6.2 | 11.5 | 13.7 | 17.5 | 22.5 | 27.3 | 0.5 |
| 15 | 6.2 | 11.4 | 13.6 | 17.2 | 22.2 | 26.5 | 0.1 |
| 16 | 6.0 | 10.8 | 14.6 | 18.1 | 22.1 | 25.5 | 0.6 |
| 17 | 5.7 | 10.4 | 14.0 | 18.4 | 21.9 | 26.2 | 0.6 |
| 18 | 5.6 | 10.4 | 13.6 | 17.7 | 21.8 | 25.6 | 0.5 |
| 19 | 5.5 | 12.0 | 14.2 | 17.8 | 22.3 | 26.3 | 0.5 |
| 20 | 5.5 | 11.8 | 13.6 | 17.2 | 21.9 | 25.4 | 0.7 |

Figure 2: Validation profit (M DKK) for different combinations of $K$ and $\gamma$.

## 4.3 Final Policy and Interpretation

Figure 3 shows the optimal policy obtained from value iteration using the best hyperparameter configuration ($K = 14$ bins and $\gamma = 0.99$). The resulting strategy follows a clear arbitrage pattern: For low SOC and price values, the battery charges. The higher the SOC, the lower the price bin has to be for the policy to be charge. Similarly, at high price levels, the battery discharges even at lower SOC levels. Near the physical limits, the actions get restricted and, therefore, at low prices and high SOC and at high prices and low SOC the action becomes idle.



| SOC Level | p0 | p1 | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p9 | p10 | p11 | p12 | p13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOC 0 | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | · | · | · | · | · | · | · | · |
| SOC 1 | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| SOC 2 | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| SOC 3 | ↑ | ↑ | ↑ | ↑ | ↑ | · | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| SOC 4 | · | · | · | · | · | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |

Figure 3: Optimal RL policy obtained from value iteration. Green cells correspond to charging, red to discharging, and gray to idle actions.

The transition region around price bin 5 is particularly interesting: while the SOC levels not affected by the physical limits (SOC 1 and SOC 2) switch from charging to discharging, SOC 3 functions as a narrow idle band before committing to discharge in price bin 5. This arises from the interaction between expected price transitions, the long-term value of preserving flexibility, and the discretized structure of the state space. Overall, the policy aligns with expected threshold behavior for arbitrage tasks, and despite using 14 price bins, reveals a clear threshold structure where most decisions hinge on low vs high price regimes. The transitional behavior around bin 5, where SOC 3 idles while others switch, suggests that only one intermediate bin is needed to preserve flexibility. This supports collapsing the discretization to just three bins: low, transitional, and high without significant loss of strategic nuance.

## 5 Model Evaluation and Comparison

| **Model** | **Total Profit (DKK)** | **Optimality Gap (% of max)** |
|---|---:|---:|
| Theoretical maximum | 56,232,235 | 100.00% |
| Logistic Regression (LR) | 5,468,222 | 9.72% |
| Reinforcement Learning (RL) | 25,628,184 | 45.58% |

Table 2: Summary of cumulative profits and efficiency relative to the theoretical max.

In this section, we evaluate the performance of the two proposed control strategies: Logistic Regression (Model 1) and Reinforcement Learning (Model 2). The models are evaluated on the test set using real-world electricity prices ($\lambda_t^{\text{test}}$). We compare their cumulative profits against the theoretical maximum and analyze the behavioral differences in their charging and discharging cycles.

### 5.1 Performance Comparison

Table 2 summarizes the total financial performance of both models compared to the Theoretical Maximum over the test period. The theoretical maximum achieved a total profit of approximately 56.2 million DKK. The Logistic Regression (LR) model captured 9.72% of this potential. However, the Reinforcement Learning (RL) model significantly outperformed the supervised approach, achieving a total profit of 25.63 million DKK, corresponding to an optimality gap of 45.58%.

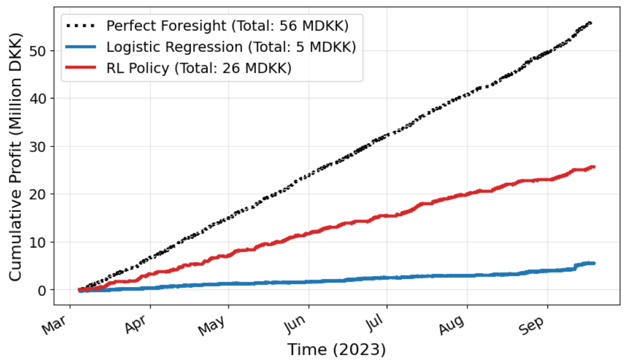


Figure 4: Cumulative profit comparison: The black dotted line represents the theoretical maximum. The blue and red lines represent the Logistic Regression and RL policies, respectively.

This substantial difference highlights the limitations of the supervised classification approach used in Model 1. By treating each time step as an independent classification problem (Charge vs. Discharge) based on a probability threshold, the LR model fails to account for the future value of stored energy. Conversely, the RL agent, through the Value Iteration algorithm and a discount factor of $\gamma = 0.98$, successfully learned to weigh immediate rewards against future potential, resulting in a far more efficient arbitrage strategy.

Figure 4 illustrates the cumulative profit accumulation over time. While the LR model (blue line) grows linearly but slowly, the RL model (red line) follows a steeper trajectory, capturing nearly half of the available market value. The gap between the RL model and the theoretical maximum (black dotted line) can be attributed to the discretizations of the state space (price bins) and the lagged information structure, which prevents the agent from reacting to price spikes with perfect precision.

## 5.2 Behavioral Analysis and Dynamics

The Logistic Regression model exhibits a highly volatile, control strategy. Because the classifier makes decisions based on an immediate probability threshold without an explicit planning horizon, it tends to cycle rapidly between charging and discharging. The model rarely utilizes intermediate SOC levels, instead rushing to fully charge ($E_{\max}$) or fully discharge (0) immediately upon detecting a price signal. Crucially, because the "Idle" class was removed from the training phase to balance the dataset, the LR model lacks the capacity to wait. It effectively only idles when forced by physical constraints (e.g., attempting to discharge an already empty battery). This behavior leads to suboptimal timing, where the battery is often depleted before the true peak prices of the day occur.
In contrast, the Reinforcement Learning agent demonstrates a more strategic and patient behavior. The policy does not oscillate constantly between extremes; instead, it frequently chooses to remain idle even when the battery is not full or empty. This indicates that the Value Iteration algorithm successfully learned to weigh immediate profits against the value of future flexibility. By holding a charge during moderately high prices and discharging only during extreme price spikes, the RL agent maximizes the spread between buy and sell prices. This strategic smoothes out the operation cycles compared to the erratic LR model is the primary driver of its superior financial performance.

# 6 Conclusion

This assignment investigated two data-driven approaches for Battery Energy Storage System (BESS) a supervised Logistic Regression classifier (Model 1) and a Reinforcement Learning agent (Model 2).
The results demonstrate a clear performance hierarchy. The Logistic Regression model, trained on optimal labels, struggled to capture significant market value (9.72% of optimal). Its myopic decision-making, limited by a binary classification threshold and a lack of temporal planning, resulted in an aggressive cycling strategy that often mistimed the market.
In contrast, the Reinforcement Learning model achieved a capture rate of 45.58%, generating over 25.62 million DKK. By formulating the problem as an MDP, the RL agent successfully learned a stationary policy that accounts for the stochastic nature of price transitions and the value of future flexibility. The behavioral analysis highlighted that the RL agent learned to be idle when price signals were weak whereas the LR agent had to be reactive.

# A Appendix

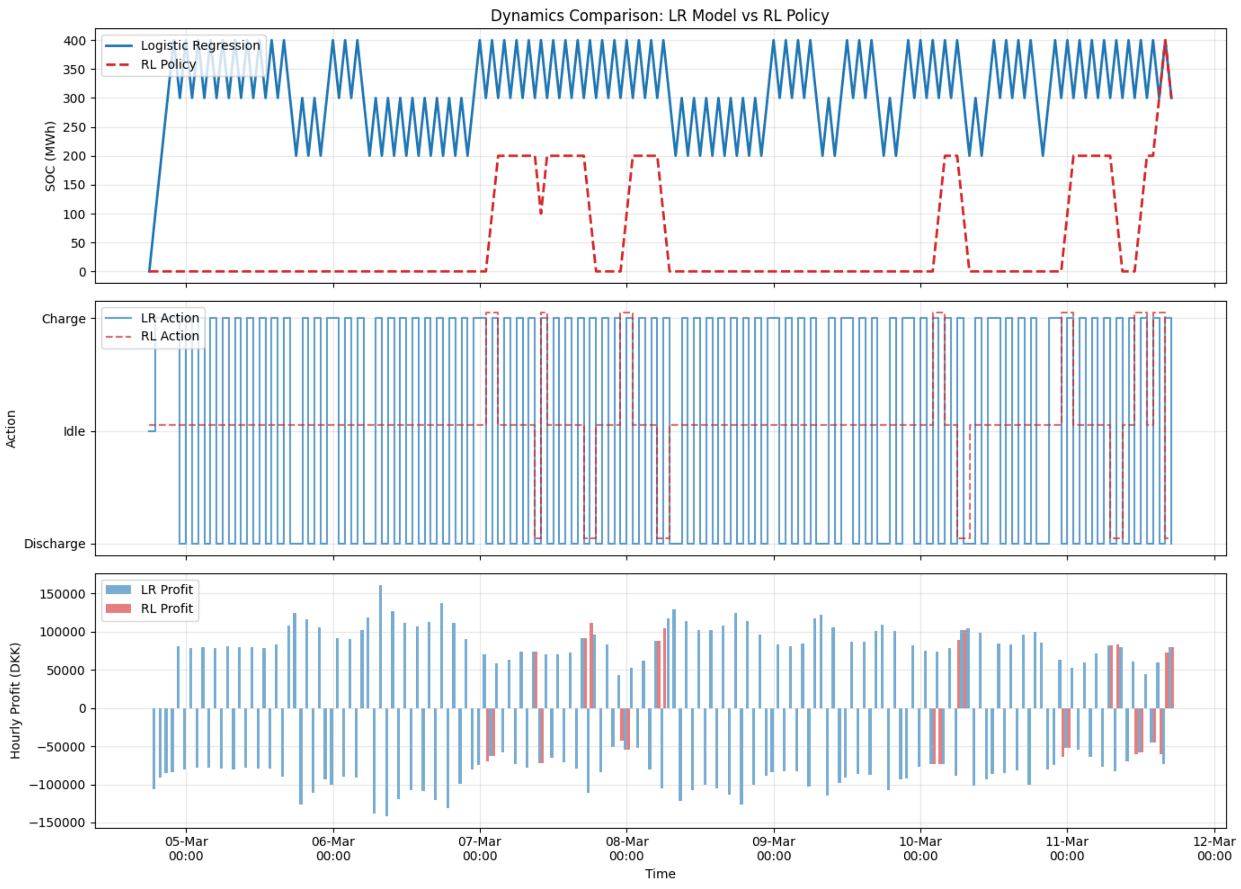


Figure 5: Dynamics comparison between Logistic Regression (blue) and RL Policy (red dashed) over a one-week period. Top: State of Charge (SOC). Middle: Actions (Charge, Idle, Discharge). Bottom: Hourly Profit.