# Sparse Sequential Learning

## Exploring Stochastic Contextual Linear Bandit and Feature Selection Combinations for Fixed Reduced Dimensions

**Vivek Kasyap Pasumarthi**[1]
**Supervisor: Julia Olkhovskaia**[1]
[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfillment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Vivek Kasyap Pasumarthi
Final project course: CSE3000 Research Project
Thesis committee: Julia Olkhovskaia, Luciano Cavalcante Siebert

## Abstract

Stochastic contextual linear bandits are widely used for sequential decision-making across many domains. However, in high-dimensional sparse settings, most candidate features are irrelevant to predicting outcomes, and collecting such data is costly. This study examines various SCLB algorithms combined with feature-selection methods using a fixed reduced dimensionality. With a synthetic dataset exhibiting 90% feature sparsity, we simulated these algorithms and compared their performance to both their base counterparts and to one another. Our experiments demonstrate that integrating feature selection into posterior- and confidence-based bandits can achieve nearly identical regret while requiring only a fraction of the data-collection cost.


## 1 Introduction

The bandit problem describes a sequential decision-making model in which a learner interacts with an environment over multiple rounds. In each round, the learner chooses an action from an action set, and the environment reveals a corresponding reward. The learner's aim is to maximize its cumulative reward by selecting the best possible action per round without any prior knowledge of the actions' reward distributions. In the stochastic contextual linear bandit (SCLB) setting, the learner also observes additional information and computes a feature vector per action. The reward for each action is a linear function of its feature vector and an unknown parameter vector. When sequentially selecting actions and observing their rewards, the learner balances exploration, selecting an action to learn more about the parameter vector, and exploitation, selecting the best possible action at that moment. Some real-life applications include healthcare [7], recommender systems [16], and dialogue systems [12]. *Bouneffouf and Rish* [5] survey other practical uses of contextual bandits.

Stochastic linear bandits with high-dimensional sparse features are a practical model for a variety of domains, such as personalized medicine and online advertising [4]. In many of these applications, there is a wide range of candidate features, but only a few are relevant for predicting outcomes [8]. In addition, a large number of samples is required to make accurate predictions, and acquiring them can be expensive. This challenge motivates the study of SCLBs within the high-dimensional sparse regime. Prior work to exploit sparsity in contextual bandits includes agnostic lasso bandits [9], doubly robust estimators [10], and high-dimensional online methods [4]. However, to the best of our knowledge, there is no explicit effort in considering feature selection techniques during the bandit process. To address this gap, we propose to employ sparse regression to construct a sparse reward estimate and embed feature selection methods within SCLBs to exploit the feature sparsity, with the intent of reducing data costs and speeding up learning.

In this study, we consider different SCLBs and feature selection methods, whose combinations produce the algorithms we want to empirically analyze. Specifically, we aim to answer the question: **How do the base bandits and combinations of SCLBs and feature selection methods compare in a high-dimensional sparse setting when constrained to a fixed number of features?** To answer this, we consider the following sub-questions:

1. How does the average cumulative regret change as the number of selected features and the timing of feature selection vary?
2. How do the algorithms perform relative to the base SCLBs?
3. How do the algorithms perform relative to each other?
4. How do the best average cumulative regrets of the algorithms vary as the feature sparsity changes?

## 2 Background and Methodology

This section formally states the problem at hand, introduces the relevant algorithms, and describes the methodology.

### 2.1 Problem Formalization

A stochastic contextual linear bandit (SCLB) is a sequential decision-making model that runs for $T \in \mathbb{Z}^+$ rounds, where $T$ is called the *horizon*. In each round $t \in \{1, ..., T\}$, the learner observes a set $\mathcal{A}_t$ of $N$ actions, as well as additional information $\mathcal{C}_t$, called the *context*, from the environment. For each action $a_{i,t} \in \mathcal{A}_t$, the learner applies some feature map $\phi(\mathcal{A}_t, \mathcal{C}_t)$ to produce a $d$-dimensional feature vector $x_{i,t} \in \mathbb{R}^d$, where $d$ is the *ambient dimension*. The environment yields the true reward $r_t = x_t^\top \theta^* + \eta_t$, where $x_t$ is the feature vector of the chosen action, $\theta^*$ is an unknown parameter vector and $\eta_t$ is a zero-mean $\sigma$-sub-gaussian noise term. When sequentially choosing actions, the learner maintains an estimate of the parameter vector $\hat{\theta}$, and computes an estimated reward $\hat{r}_{i,t} = \langle x_{i,t}, \hat{\theta}_t \rangle + \eta_t$ for each action. It balances exploration — selecting actions via some policy to improve its estimate of the parameter vector, and exploitation — using the current estimate of the parameter vector to choose the best possible action. Various bandit learners differ in the policy they use to explore and exploit. After observing the true reward $r_t$, the learner updates $\hat{\theta}$ and proceeds to the next round. The learner's objective is to minimize expected cumulative regret after $T$ rounds.

$$\mathbb{E}[R_T] = \sum_{t=1}^{T} \mathbb{E}\left[ \max_{x \in \phi(\mathcal{A}_t, \mathcal{C}_t)} x^\top \theta^* - x_t^\top \theta^* \right]$$

Under the sparse setting, $\theta^*$ is a sparse vector with $s\%$ of the features being non-zero.

### 2.2 Related works

**$\varepsilon$-greedy** [3]: this learner does exploration by choosing an action uniformly at random with probability $\varepsilon$, and does exploitation by selecting the best possible action using its

parameter vector estimate with probability $1 - \varepsilon$.

**Explore-Then-Commit (ETC)** [11][8]: this learner does exploration by selecting each action a fixed number of times in the initial rounds and collecting data. Then, it computes a parameter vector estimate $\hat{\theta}$. Finally, it does exploitation by committing to $\hat{\theta}$ when choosing actions in the subsequent rounds.

**LinUCB** [6][1]: this learner is based on the principle of optimism in the face of uncertainty. It maintains a regularized design matrix $V_\tau = I + \sum_{t=1}^{\tau} x_t x_t^\top$, a reward-feature vector $b_\tau = \sum_{t=1}^{\tau} x_t r_t$ and a ridge-regression estimate $\hat{\theta}_t = V_t^{-1} b_t$. In each round, the learner computes an upper confidence bound (UCB) for each action, then chooses the one with the highest value. $\text{UCB}_{i,t} = x_{i,t}^\top \hat{\theta}_{t-1} + \alpha\sqrt{x_{i,t}^\top V_{t-1}^{-1} x_{i,t}}$, where $\alpha$ is an exploration bonus. The first term is the estimated reward which drives exploitation, and the second term is an uncertainty bonus, which drives exploration. By selecting the action with the largest UCB value, the learner balances exploration and exploitation. A closely related variant, OFUL replaces the fixed bonus $\alpha$ with a coefficient $\beta_t = \sqrt{\lambda}S + R\sqrt{d\ln(1/\delta) \; + \; d\ln(tL^2/d\lambda)}$.

**Thompson Sampling (TS)** [2]: this learner maintains a Gaussian posterior $\mathcal{N}(\hat{\mu}, \nu^2 V^{-1})$, where $V = I + \sum_{t=1}^{T} x_t x_t^\top$ is a regularized Gram matrix, $\hat{\mu} = V^{-1} \sum_{t=1}^{T} r_t x_t$ is the current least-squares estimate of the unknown parameter vector, and $\nu^2$ is the noise variance. In each round, it samples $\tilde{\mu}_t$ from the posterior and selects the action $a_t$ with the maximum score, the score being $x_t^\top \tilde{\mu}_t$. By doing so, it gradually transitions from exploration via initial randomized sampling to exploitation due to the concentration of $\tilde{\mu}$ around the true parameter vector $\theta^*$.

**ANOVA F-value** [15]: this feature selection method uses univariate statistical tests to select the top k features. It computes an F-score and a corresponding p-value for each feature. Then, it ranks the features by their F-scores in descending order and returns the indices of the top k highest-scoring features.

**Recursive Feature Elimination (RFE)** [14]: this feature selection method does recursive feature elimination with a lasso estimator. It fits the regression model on the full feature set. It ranks the features by the magnitude of their learned coefficients and removes the lowest ranking feature. This process is repeated for multiple iterations, until only $k$ feature remain. Then, it returns the corresponding indices.

## 2.3 Methodology

**General setup**: We run several simulations for the algorithms, each performing multiple trials. At the start of every trial, we sample a new hidden parameter vector $\theta^*$ with $s\%$ sparsity. In each round $t \leq T$, the environment supplies a new set of $N$ feature vectors. The learner selects an action $a_t$ and uses its corresponding feature vector $x_t$ to observe the corresponding reward $r_t$ from the environment. The learner also maintains a history $H$ of $(x_t, r_t)$ pairs. After the first $p$ rounds, it uses $H_t\{(x_1, r_1), ...((x_p, r_p)\}$ to run a one-time feature selection routine and select the $k$ best features most predictive of the reward. We define $p$ as a warm-up period for the feature selection. For the remaining rounds $t > p$, the learner projects the feature vectors down to those $k$ selected features. At the end of each trial, we compute a cumulative regret graph. Over the course of multiple trials, the graphs are averaged.

**$\varepsilon$-greedy-FS**: this algorithm uses a contextual version of the $\varepsilon$-greedy learner described by *Auer et al.* [3]. It maintains a least-squares estimate of the parameter vector $\hat{\theta}$ using an online lasso regressor. In each round, the learner chooses an action uniformly at random with probability $\varepsilon$. Otherwise, for each action $a_{i,t}$, it estimates the reward $\hat{r}_{i,t} = x_{i,t}^\top \hat{\theta}_t + \eta_t$ and selects the action with the largest value. On choosing an action, the learner observes the true reward $r_t$ and updates its lasso regressor with the $(x_t, r_t)$ pair. At the end of round $p$, the algorithm runs a one-time feature selection routine to pick the best $k$ features. Then, it retrains its lasso regressor to only operate on these $k$ dimensions. In subsequent rounds, the learner reduces the feature vectors to these $k$ selected features.

**ETC-FS**: this algorithm uses a simplified version of the contextual ETC learner described by *Hao & Lattimore* [8]. The learner explores by selecting each of the $N$ actions exactly $m$ times for the first $mN$ rounds and recording $(x_t, r_t)$ pairs. At the end of round $mN$, it computes a parameter vector estimate $\hat{\theta}$ using a lasso regressor. In subsequent rounds $t > mN$, the learner sequentially selects actions with the highest reward estimates $\hat{r}_{i,t} = x_{i,t}^\top \hat{\theta} + \eta_t$ and commits to $\hat{\theta}$. At the end of round $p$, the algorithm runs a one-time feature selection routine to pick the best $k$ features. Like $\varepsilon$-greedy-FS, it retrains its lasso regressor to only operate on these $k$ dimensions for rounds $\{p+1, ..., T\}$.

**LinUCB-FS**: this algorithm uses the LinUCB learner described by *Chu et al.* [6], with the time-varying bonus $\beta(\delta)$ term from OFUL [1]. It maintains a regularized covariance matrix $V$, as well as $V^{-1}$, updated efficiently via Sherman-Morrison rank-1 formula, as well as response vector $b$ and regression estimate $\hat{\theta} = V^{-1}b$. At each round $t$, it computes $\beta_t = \sqrt{\lambda} + \sqrt{2\ln(1/\delta) \; + \; d\ln(1 + (t-1)/\lambda d)}$ and $\text{UCB}_{i,t} = x_{i,t}^\top \hat{\theta}_{t-1} + \beta_t\sqrt{x_{i,t}^\top V_{t-1}^{-1} x_{i,t}}$, then selects the action with the highest UCB. After the first $p$ rounds, the algorithm runs a one-time feature selection routine to pick the best $k$ features. It projects $V$, $V^{-1}$, b and $\hat{\theta}$ to the $k$-dimensional subspace. All subsequent updates and action selection steps take place in this reduced space.

**TS-FS**: this algorithm uses the TS learner described by *Agarwal & Goyal* [2]. It maintains a regularized covariance matrix $V$, as well as $V^{-1}$, updated efficiently via

Sherman-Morrison rank-1 formula, as well as a response vector $b$ and the posterior mean $\mu = V^{-1}b$. In each round, it draws a sample $\tilde{\theta}$ from $\mathcal{N}(\mu, \nu^2 V^{-1})$. For each action, a score $x_t^\top \tilde{\theta}_t$ is computed and the one with the highest score is chosen. After the first $p$ rounds, the algorithm runs a one-time feature selection routine to select the best $k$ features. Like LinUCB-FS, it projects $V$, $V^{-1}$, b and $\mu$ to the $k$-dimensional subspace. In the subsequent rounds, all posterior updates and action selection steps operate in this reduced feature space.

## 3 Experimental Setup

This section describes the setup used to simulate the algorithms.

Our framework uses a synthetically generated dataset. In this study, we executed multiple simulations, each consisting of 30 independent trials. We set the feature map $\phi(\mathcal{A}_t, \mathcal{C}_t)$ to the identity function, thus using the context itself as the feature vector set. At the start of each trial, we sampled $\theta^* \in \mathbb{R}^{100}$ from a standard normal distribution and imposed $90\%$ sparsity by setting a random subset of its entries to 0. The trials were conducted for 500 rounds. In each round, the environment produced a set of 10 feature vectors, each $\in \mathbb{R}^{100}$ and drawn i.i.d. from a multivariate Gaussian distribution with mean 0 and covariance $I_{100}$. Rewards were generated using a 0.05-sub-gaussian noise term. At the end of each trial, we computed its cumulative regret curve. Finally, for each simulation, we averaged the per-round cumulative regret across the 30 trials to attain a smooth graph.

Prior to our main simulations, we fine-tuned the hyperparameters of the base SCLBs to their best-performing configuration in the target environment by running multiple simulations of each SCLB across a range of hyperparameter configurations and selecting the one that minimizes the average cumulative regret curve. Table 1 shows the hyperparameters considered in these simulations.

| Algorithm | Grid Search Space | Selected |
|---|---|---|
| $\varepsilon$-greedy | $\alpha \in \{0.01, 0.05, 0.1, 0.5\}$, $\varepsilon \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ | $\alpha_1 = 0.1,\ \varepsilon = 0.01$ |
| ETC | $\alpha \in \{0.01, 0.05, 0.1, 0.5\}$, m $\in \{1, 5, ..., 10, 20, 30, 40, 45\}$ | $\alpha = 0.1$ |
| LinUCB | $\delta \in \{0.001, 0.01, 0.1, 0.5, 0.9\}$, $\lambda \in \{0.1, 1, 10, 15, 20, 25\}$ | $\delta = 0.9,\ \lambda = 15$ |
| TS | $\lambda \in \{0.1, 0.5, 1, 5, 10, 20\}$ | $\lambda = 1$ |

Table 1: Hyperparameter grid and selected values for each bandit algorithm.

For the RFE feature selection method, we chose $\alpha = 0.1$, with a step size of 1.

Following the hyperparameter tuning, we ran our main algorithm simulations. $\varepsilon$-greedy-FS and ETC-FS both build a new lasso regressor on the reduced feature space. Here, we used regularization $\alpha = 0.1$. For each algorithm, we conducted a grid search over $p$ and $k$, visualizing its performance as a heatmap of average cumulative regret. For benchmarking, we compared each algorithm with its corresponding base SCLB. We also compared the algorithms with each other using their best $(p, k)$ configuration.

## 4 Results and Discussion

The performance of an algorithm is measured by its average cumulative regret curve. The main goal of a learner is to select the best possible action per round. The fewer rounds it takes to do this, the smaller the average cumulative regret at the end of 500 rounds.

### 4.1 Base SCLB vs. SCLB-FS

**$\varepsilon$-greedy and $\varepsilon$-greedy-FS:**

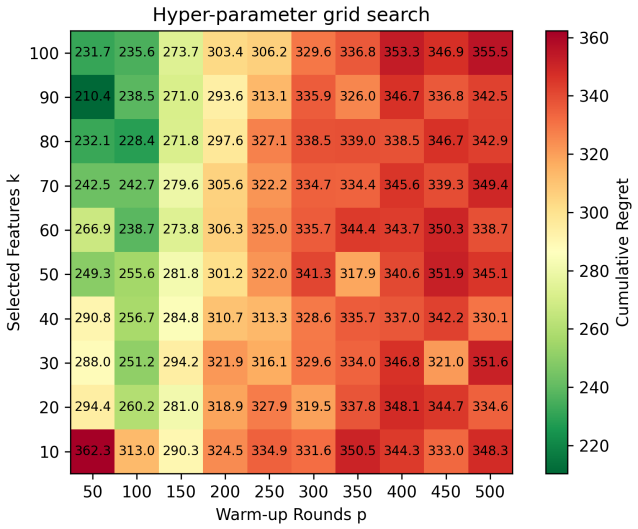


Figure 1: $\varepsilon$-Greedy-FS (ANOVA-F) heatmap

Figure 1, Figure 2 and Figure 3 demonstrate that integrating feature selection into the $\varepsilon$-Greedy bandit yields modest reductions in cumulative regret, while also decreasing the volume of data collected. In (Figure 1), early feature selection (small $p$) improves regret when a relatively large number of features are selected, most likely due to ANOVA-F's univariate scoring omitting some relevant features. By contrast, Figure 2 achieves its lowest regret at low $p$ and $k$ values. Finally, Figure 3 shows both $\varepsilon$-Greedy-FS algorithms to improve on the baseline $\varepsilon$-Greedy performance, highlighting that learning can occur just as effectively while collecting substantially less contextual data each round.

**Explore-Then-Commit and ETC-FS:**

Figure 6 shows that ETC-FS algorithms offer at best, marginal improvement over the base ETC contextual bandit.

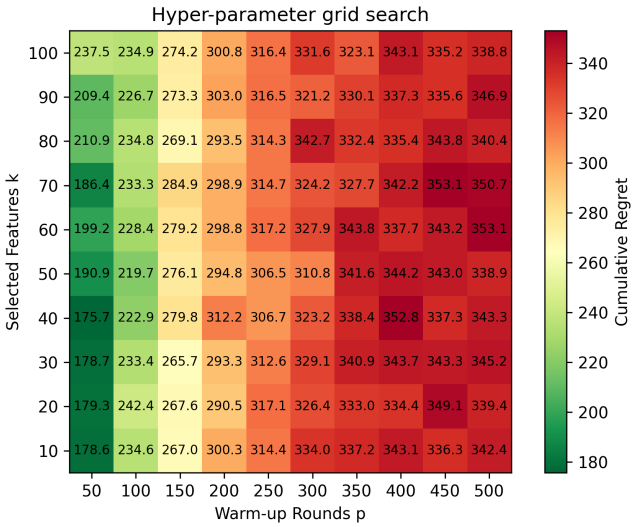


Figure 2: $\varepsilon$-Greedy-FS (RFE) heatmap

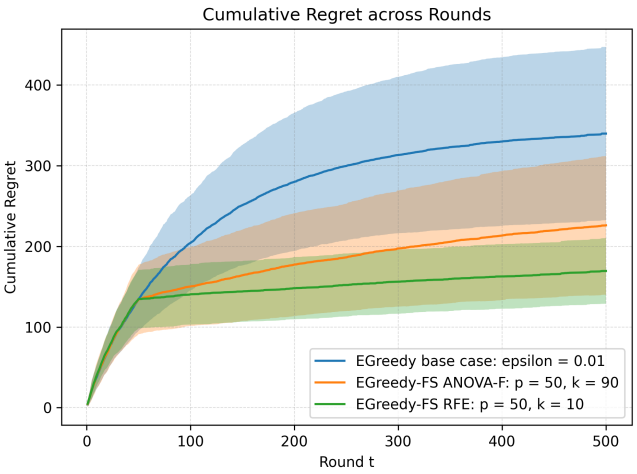


Figure 3: $\varepsilon$-Greedy vs $\varepsilon$-Greedy-FS

In Figure 4, reducing $p$ and $k$ increases the cumulative regret, while in Figure 5, performance essentially remains unchanged. The important thing to highlight is that the exploration phase is indifferent to feature selection, since the action selection policy does not rely on the selected features. Column $p = 100$ in Figure 4 and Figure 5 represents any column $p < mN$. Because this approach relies on collecting data and training a regression estimate of $\hat{\theta}$, feature selection during exploitation also yields no benefits. Nevertheless, the experiment demonstrates that a reduced feature space can perform equally well, although this space is not discovered until after the algorithm is run. A modified learner that re-enters exploration at a later stage could exploit this insight.

**LinUCB and LinUCB-FS:**

Figure 9 shows that the LinUCB bandit benefits from feature selection by delivering equal or improved performance

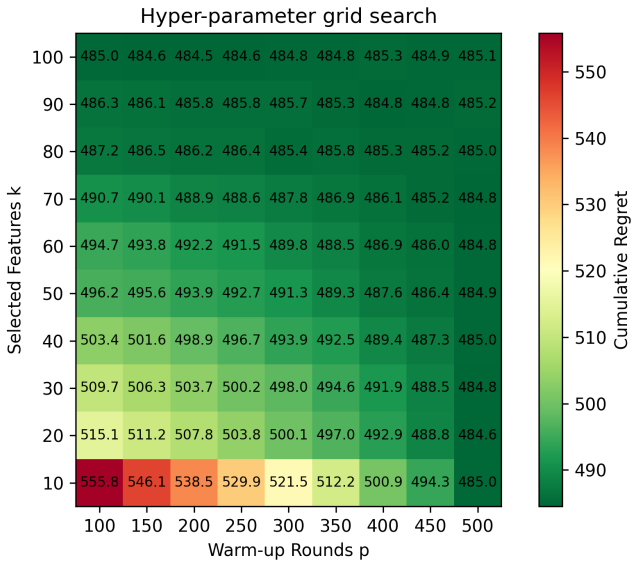


Figure 4: ETC-FS (ANOVA-F) heatmap

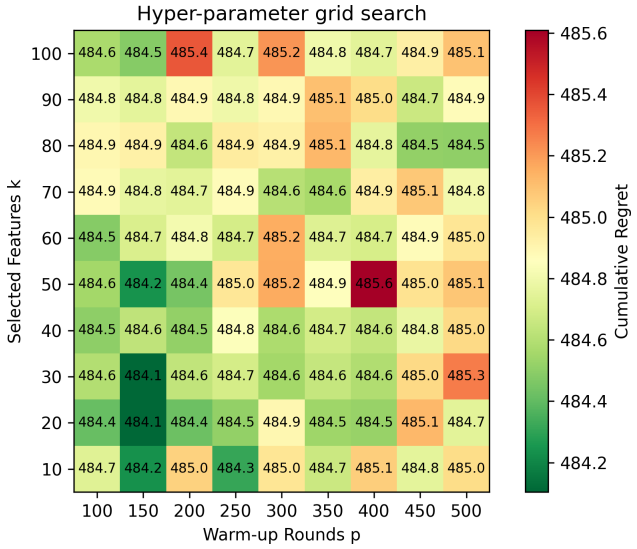


Figure 5: ETC-FS (RFE) heatmap

while using fewer features per update, thereby reducing both computational and data-collection burdens. Figure 7 exhibits roughly consistent performance at moderate to large $p$ and $k$ values. Figure 8 yields substantial regret reduction even at minimal $p$ and $k$ values, demonstrating that early removal of extraneous features accelerates convergence to the optimal arm and highlighting that it is important to not omit the features that determine the reward.

**Thompson Sampling and TS-FS:**

Figure 10, Figure 11, and Figure 12 mirror the LinUCB findings under a Bayesian sampling regime. In Figure 10, higher values of $p$ and $k$ are required to match the perfor-

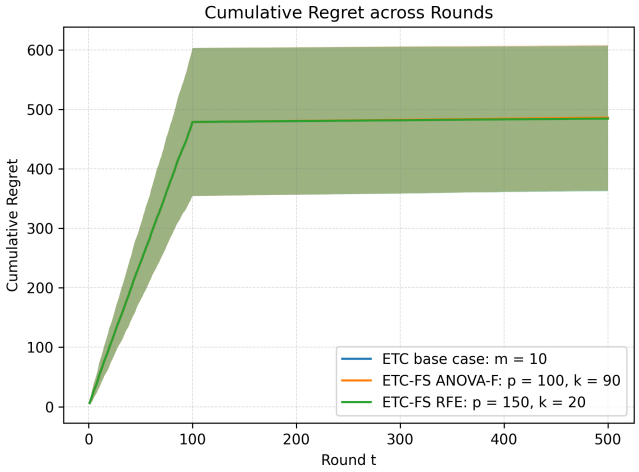


Figure 6: ETC vs ETC-FS

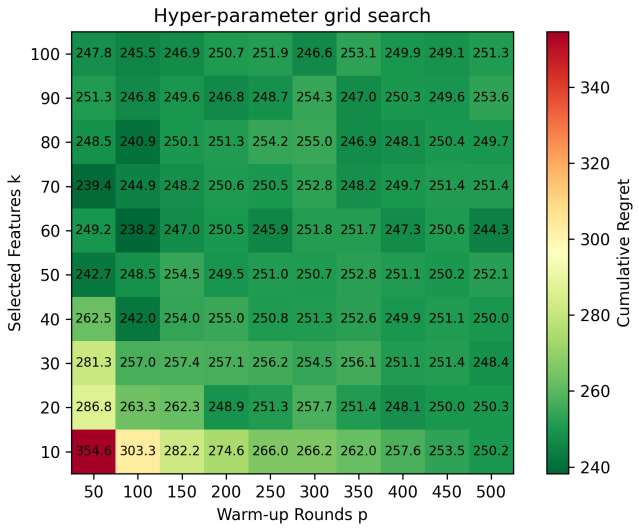


Figure 7: LinUCB-FS (ANOVA-F) heatmap

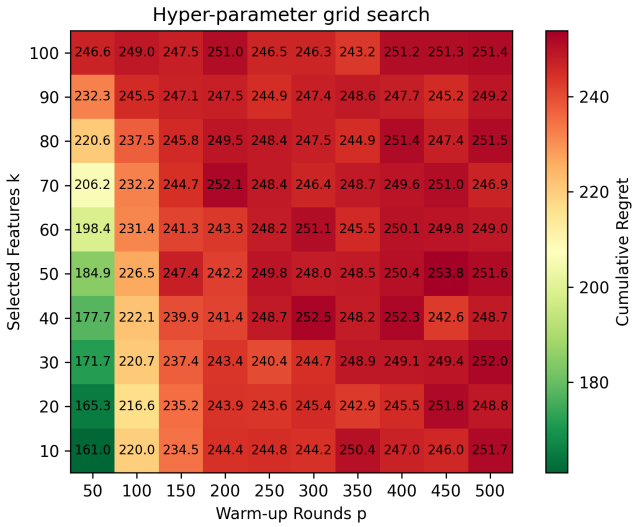


Figure 8: LinUCB-FS (RFE) heatmap

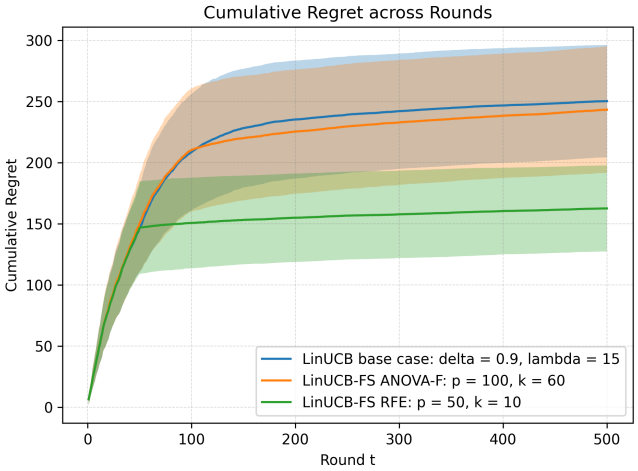


Figure 9: LinUCB vs LinUCB-FS

mance of the base TS contextual bandit, but less feature data is required. Figure 11 achieves pronounced regret reduction at low $p$ and $k$ values, likely due to the posterior's early concentration around $\theta^*$ when sampling on the feature subset. Finally, in Figure 12, TS-FS with RFE delivers the greatest regret improvement among all while using only a fraction of the original feature space, saving on data-collection costs.

### 4.2 Best of SCLB-FS

Figure 13 presents the best-performing version of each algorithm. Here, ETC-FS suffers a significantly higher average cumulative regret after 500 rounds, although its slope appears flat. This elevated regret is likely influenced by the choice of $m$. Additionally, feature selection plays a relatively insignificant role in this algorithm. $\varepsilon$-greedy-FS's regret eventually exceeds that of LinUCB-FS, due to additional exploration error. Meanwhile, LinUCB-FS and TS-FS perform similarly to their base counterparts. All these algorithms use RFE for feature selection, highlighting the importance of identifying the reward-predicting features to these algorithms. The benefit, however, is a reduced feature count and, consequently, lower data-collection costs.

## 5 Responsible Research

This section outlines our practices for data privacy, reproducibility, research integrity, and ethical considerations guiding the research.

### Data and Privacy

All experiments use synthetically generated data; no personal data or sensitive information was collected, stored, or shared. Consequently, traditional privacy risks, ethical review requirements, and data sharing policies such as F.A.I.R. do not apply.

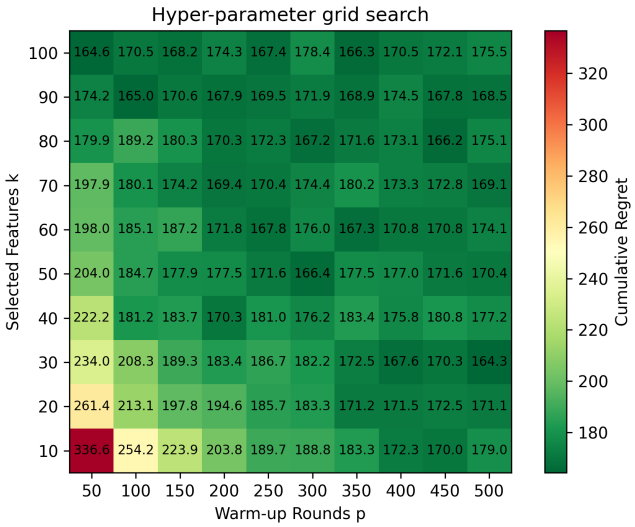


Figure 10: TS-FS (ANOVA-F) heatmap

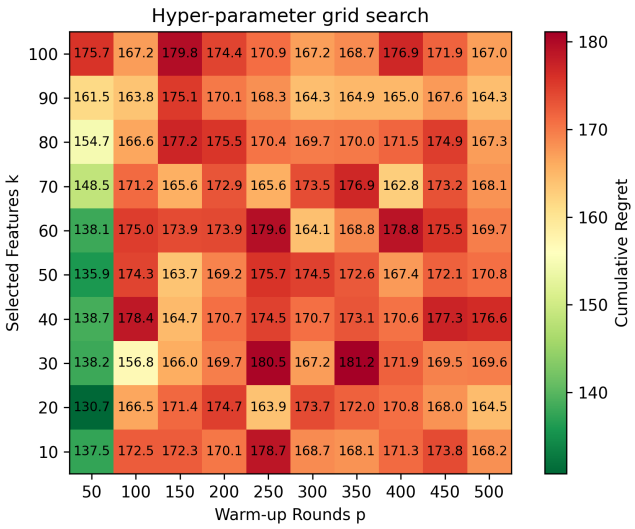


Figure 11: TS-FS (RFE) heatmap

## Reproducibility

To mitigate the effects of randomness in our simulations, we perform multiple independent trials and report averaged metrics. Complete experiment configurations and analysis code are publicly available in the GitHub repository, ensuring full transparency and enabling easy replication of the results. One point to note is that the stochastic nature of the algorithms can result in slight variations in values. However, the overall trends remain consistent.

## Research Integrity

This research is conducted with commitment to research integrity. Every consulted source is listed in the bibliography, and all directly used materials are properly cited. We do not rely on external datasets or licensed components. All our methods, analyses, and findings are reported accurately

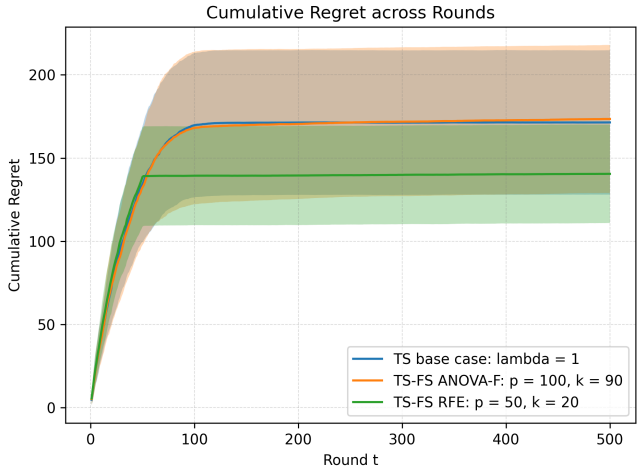


Figure 12: TS vs TS-FS

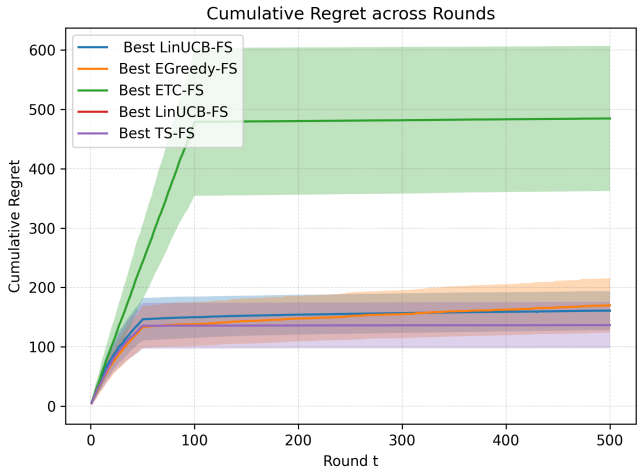


Figure 13: SCLB-FS comparisons

to ensure clarity and accountability. LLMs, specifically ChatGPT [13], have been utilized responsibly in this research for summarizing concepts, debugging code, and rephrasing this paper in a more formal style.

## Ethical Implications

Our work is limited to synthetic-data scenarios and is not yet intended for real-world use. While integrating feature selection into bandit algorithms shows promise, real-world applications, especially in sensitive domains, will require careful tuning, fairness evaluation, and extensive testing to guard against bias or harm. We emphasize methodological transparency and advise against premature use without rigorous validation.

# 6 Conclusion and Future Work

## 6.1 Conclusion

In this work, we empirically evaluated the $\varepsilon$-Greedy-FS, ETC-FS, LinUCB-FS, and TS-FS algorithms, each paired

with ANOVA-F and RFE feature-selection routines. The aim of this paper was to empirically analyze the different algorithms by evaluating their performances relative to their base counterparts and with each other. Across all algorithms, RFE consistently outperformed ANOVA-F, demonstrating the importance of selecting the reward-predicting features for these algorithms. Embedding feature selection into posterior or confidence-based bandits can yield nearly identical regret performance at a fraction of the data-collection cost.

### 6.2 Future Work

This study can be extended in several directions.

1. **Real-world datasets**: Evaluate the integration of feature selection into SCLBs on real-world data. Our experiments were limited to a synthetic dataset; future work can test diverse, real-world datasets and more realistic conditions.
2. **Additional feature-selection methods**: Due to time constraints, we only considered ANOVA-F and RFE. Future research could explore other feature-selection algorithms and hybrid approaches.
3. **Varied sparsity levels**: We evaluated only 90% sparsity; subsequent studies should analyze performance across a spectrum of sparsity levels.

## 7 Acknowledgment

I would like to express my sincere gratitude to my Supervisor Julia Olkhovskaia for her invaluable guidance, thoughtful advice, and consistent feedback. Her support played a crucial role throughout the course of this project. I would also like to thank my research peers, whose contributions to the base code and their readiness to grant help whenever needed were instrumental in the completion of this project.

## References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, volume 24, pages 2312–2320, 2011.

[2] Shilpa Agarwal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2–3):235–256, 2002.

[4] Hamsa Bastani and Mohsen Bayati. Online decision making with high-dimensional covariates. *Operations Research*, 68(1):276–294, 2020.

[5] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. arXiv preprint arXiv:1904.10040 [cs], 2019.

[6] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[7] Audrey Durand, Charis Achilleos, Demetris Iacovides, Katerina Strati, Georgios D. Mitsis, and Joelle Pineau. Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine Learning for Healthcare Conference*, pages 67–82, 2018.

[8] Botao Hao, Tor Lattimore, and Mengdi Wang. High-dimensional sparse linear bandits. arXiv preprint arXiv:2011.04020, 2021.

[9] Min hwan Oh, Garud Iyengar, and Assaf Zeevi. Sparsity-agnostic lasso bandit, 2021.

[10] Gi-Soo Kim and Myunghee Cho Paik. Doubly-robust lasso bandit, 2020.

[11] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, Cambridge, UK, 2020.

[12] Bing Liu, Tong Yu, Ian Lane, and Ole J. Mengshoel. Customized nonlinear bandits for online response selection in neural conversation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5245–5252, 2018.

[13] OpenAI. ChatGPT: Large language model. https://chat.openai.com/, 2023. Accessed: 2025-06-22.

[14] scikit-learn developers. *RFE — Recursive Feature Elimination*. scikit-learn, 2025. Accessed: 2025-06-22.

[15] scikit-learn developers. *SelectKBest — Feature selection: univariate statistical tests*. scikit-learn, 2025. Accessed: 2025-06-22.

[16] Qian Zhou, XiaoFang Zhang, Jin Xu, and Bin Liang. Large-scale bandit approaches for recommender systems. In *International Conference on Neural Information Processing*, pages 811–821. Springer, 2017.