UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Bazen Teklehaymanot Tadele

Variance Reduction In Online Controlled Experiments

Master's Thesis (30 ECTS)

Supervisor(s): Carlos Bentes, MSc
Elena Sügis, PhD

Tartu 2025

Variance Reduction In Online Controlled Experiments

Abstract:

Online controlled experiments help determine causal effects by randomly assigning participants to treatment and control groups. However, a high variance within experimentation data can obscure treatment effects. Variance reduction techniques mitigate this noise, enhancing sensitivity, reducing the required sample size, and experiment duration. Traditional variance reduction methods often struggle to reduce variance effectively because they don't fully account for the partial correlations between multiple covariates and business metrics. In this study, we integrate and evaluate established machine learning-based variance reduction techniques within the internal experimentation platform of Bolt Technology OÜ, an Estonian multinational mobility company. We show that these approaches reduce the variance of experiment metrics by up to 51.2% and outperform existing internal methodologies by 6%. Our findings demonstrate that machine learning-based methods can effectively overcome the limitations of traditional methods, offering a more robust variance reduction in controlled experiments in the ride-hailing domain.

Keywords:
Variance reduction, Machine Learning, Online Controlled Experiments

CERCS: P160 - Statistics, operation research, programming, actuarial mathematics.

Hajuvuse vähendamine veebipõhistes kontrollitud katsetes

Lühikokkuvõte:

Veebipõhised kontrollitud katsed aitavad kindlaks teha põhjuslikke seoseid, jagades osalejad juhuslikult test- ja kontrollrühmadesse. Siiski võib katseandmete suur hajuvus katse mõjusid varjutada. Hajuvuse vähendamise tehnikad leevendavad seda probleemi, suurendades katse tundlikkust ning vähendades vajaliku valimi suurust ja katse kestust. Sageli ei suuda traditsioonilised hajuvuse vähendamise meetodid hajuvust tõhusalt vähendada, kuna need ei võta täielikult arvesse osalisi korrelatsioone mitme kovariandi ja ärimõõdikute vahel. Selles uurimistöös rakendame masinõppel põhinevaid hajuvuse vähendamise tehnikaid Eesti rahvusvahelise mobiilsusettevõtte Bolt Technology OÜ sisekasutuses oleval testimisplatvormil. Näitame, et sellised meetodid vähendavad katsemõõdikute hajuvust kuni 51,2% ja töötavad senistest ettevõttes kasutatud meetoditest 6% võrra paremini. Meie tulemused näitavad, et masinõppel põhinevad meetodid suudavad tõhusalt ületada traditsiooniliste meetodite piirangud, vähendades oluliselt hajuvust sõidujagamise valdkonna kontrollitud katsetes.

Võtmesõnad:
Hajuvuse vähendamine, masinõpe, veebipõhised kontrollitud katsetused

CERCS: P160 - Statistika, operatsioonanalüüs, programmeerimine, finants- ja kindlustusmatemaatika

Disclaimer

Specific business metrics and model training input features have been anonymized to protect proprietary company information. Metrics use placeholder names (e.g., business metric-I, business metric-II, etc.), and features are described by their broader categories rather than individual identifiers. Each category encompasses a group of related features.

Contents

List of Figures

List of Tables

1 Introduction

One of the most critical factors for the success of a modern business, such as a ride-hailing service, is the ability to make data-driven decisions. Online Controlled Experiment (OCE) helps to evaluate the impact of modifications to a product or service and make data-driven decisions (Kohavi et al., 2020). OCEs are widely used in the industry by companies like Google, Facebook, Amazon, Bing, Yandex, eBay, Etsy, LinkedIn, Lyft, Uber, Airbnb, Booking.com (Gupta et al., 2019), with thousands of experiments being run annually (Deng et al., 2013). At Bolt, OCEs are embedded into product development workflows to help teams measure the real-world impact of new features and changes before rolling them out broadly.

The sample size is a key factor in OCEs (Kohavi and Longbotham, 2023). It is the number of users participating in the experiment to detect the Minimum Detectable Effect (MDE) with statistical power. The users' enrollment rate determines the sample size in the experiment variants (control and treatments) and the experiment duration (Kohavi and Longbotham, 2023). At big companies, detecting small changes can have significant business implications (Deng et al., 2013). However, small MDEs increase the required sample size, prolong the experiment duration, and slow the business decision-making process. Variance Reduction (VR) techniques mitigate this by reducing noise in OCEs, lowering the required sample sizes, and reducing experiment duration.

This Master's thesis was carried out in collaboration with the Estonian mobility company Bolt. The company provided the required data, infrastructure, and mentorship to explore VR methods. The study evaluates VR techniques that utilize Machine Learning (ML) against the company's current approaches to identify potential improvements.

In this thesis, we aim to address the business question: how can we improve the sensitivity of OCEs conducted at Bolt, reduce sample sizes, shorten experiment durations, and enable faster time to market using VR techniques? To this end, we apply VR approaches that leverage ML.

The following sections of this thesis are structured as follows: Chapter 2 provides an overview of related work in VR, encompassing traditional VR techniques that use historical data Controlled Experiment Using Pre-Experiment Data (CUPED) (Deng et al., 2013) and ML-based approaches Control Using Predictions as Covariate (CUPAC) (Li et al., 2020) and Machine Learning for Variance Reduction in Online Experiments (MLRATE) (Guo et al., 2022). Chapter 3 includes the methodology, data description, implementation of VR techniques, and ML models. Chapter 4 presents the experimental results, evaluation, and a comparison of the different approaches. In Chapter 5, we discuss potential improvements for future research. Finally, Chapter 6 summarizes the thesis's outcomes.

2 Related Work

In this chapter, we start by defining OCE and VR tasks, then give an overview of the classical approaches that solve them, and finally introduce the state-of-the-art methods focusing on CUPAC and MLRATE.

2.1 Online Controlled Experiments

Different stakeholders in a company, such as product managers, data scientists, and engineers, develop a hypothesis about a new feature or service. OCE enables systematic testing of the hypothesis and measuring the impact of the new feature on Overall Evaluation Criterion (OEC), such as user engagement or revenue. Figure 1 shows the general process of OCE. The process starts with a hypothesis, followed by a design phase, where the experiment is designed and the sample size is determined. Next, the experiment is run, enrolled users receive the different variants, and the data is collected. Finally, the data is analyzed to determine the effect of the treatment.

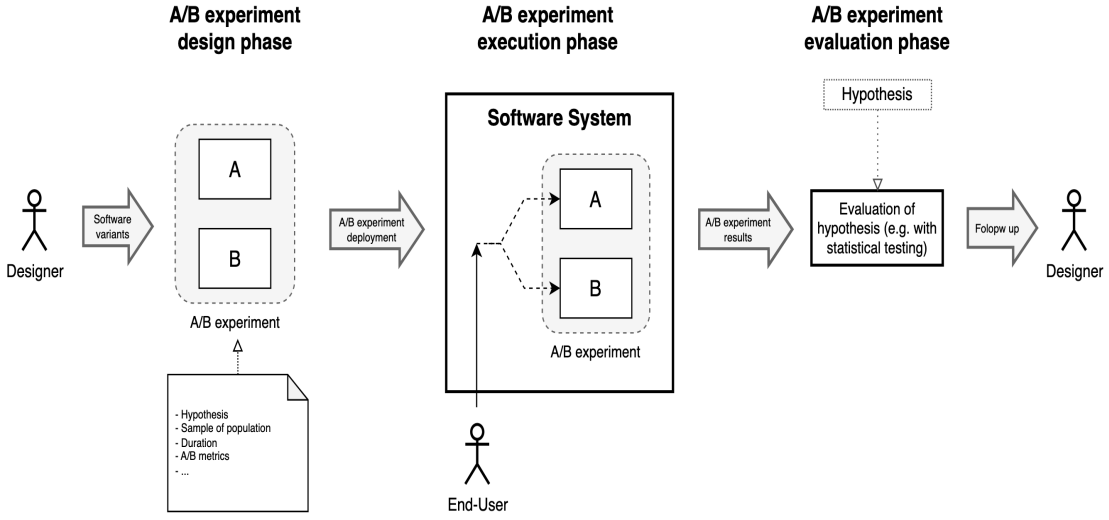

Figure 1. General A/B testing process (Garousi and Mäntylä, 2016).

In most OCE, users are randomly assigned to treatment and control groups, and the outcomes are then compared to understand the change's causal effect on specific Business Metric (BM), also known as OEC (Kohavi et al., 2020). The treatment group has access to the new feature, while the control group doesn't, see Figure 2. These online controlled experiments are also known as A/B tests or controlled experiments.

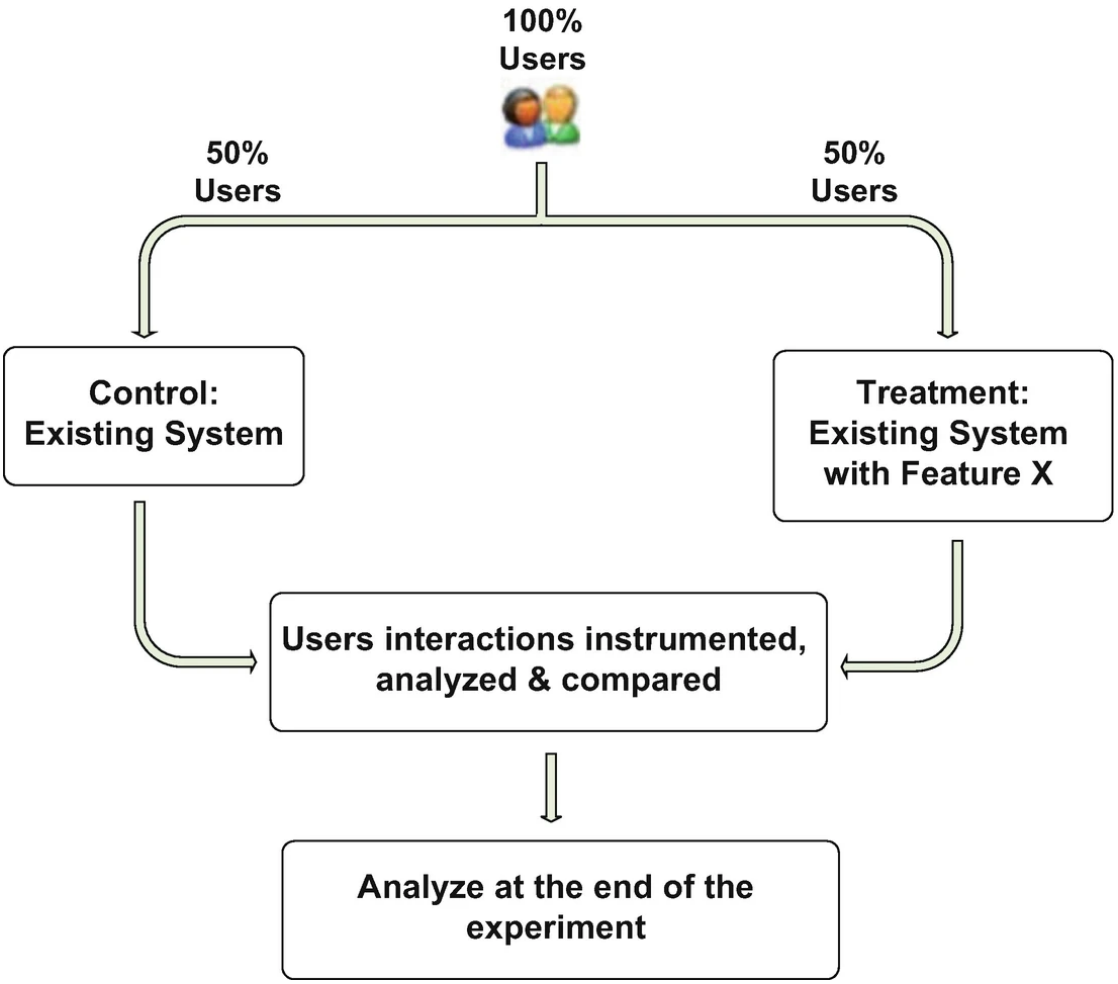

Figure 2. High-level structure of an online experiment (Kohavi and Longbotham, 2023).

2.2 Variance Reduction

Variance is a key variable in determining the required sample size of an OCE, and by extension, it also influences the experiment's duration. After developing a hypothesis, the next step in OCE is to determine the required sample size, which is a function of the MDE(δ), statistical power($1 - \beta$), significance level(α), and the variance(σ^2). Both MDE, statistical power, and significance level are fixed values. Conversely, the variance is a random variable that needs to be estimated using historical data.

For comparison of means between two independent groups, a commonly used approximation is (Serdar et al., 2021):

$$N = \frac{(r+1)(Z_{\alpha/2} + Z_{1-\beta})^2 \sigma^2}{rd^2} \tag{1}$$

where:

- N: required sample size for group 1 (if $r = 1$, same applies for group 2)
- $r = \frac{n_1}{n_2}$: ratio of sample sizes between the two groups
- σ^2: pooled variance of the two groups
- d: difference between the two group means (i.e., the MDE)
- $Z_{1-\beta}$: Z-score corresponding to the desired power (e.g., 0.84 for 80% power)
- $Z_{\alpha/2}$: Z-score for the significance level (e.g., 1.96 for $\alpha = 0.05$)

This work focuses on VR methods that leverage ML, namely CUPAC and MLRATE. These methods use ML models to adjust the target metric and estimate the treatment effect. We identify covariates and train different ML models to predict the target metric. Various types of randomization units (Kohavi et al., 2020) exist in an OCE, such as user, session, impression level, etc. In this work, we focus on experiments that have user-level randomization units.

2.2.1 Stratification

Stratification involves dividing participants into homogeneous subgroups called strata based on covariates (e.g., location or user device) (Xie and Aurisset, 2016). Each stratum is then balanced between the control and treatment groups. This helps ensure that both groups are similar with respect to these characteristics, which reduces variance and improves sensitivity. In stratification, the K strata are independent, and there is no correlation in the variability of the different subgroups. Stratified sampling reduces variance by removing the between-strata variance (Xie and Aurisset, 2016). Despite its effectiveness, the practical application of stratification is limited by the need to define strata and the difficulty of balancing them.

2.2.2 Pre-Experiment and Experiment Data

Pre-experiment data refers to historical data collected before the experiment begins. It is commonly used to estimate the variance of the target outcome during sample size calculation. Additionally, it helps identify covariates that are correlated with the target metric and can be used to adjust the outcome variable. Leveraging pre-experiment data for metric adjustment ensures that VR methods are independent of the experiment effect (Deng et al., 2013).

Experiment data refers to the data collected during the experiment and is used to estimate the treatment effect and draw conclusions about the hypothesis. Throughout this thesis, we use the terms experiment data and in-experiment data interchangeably.

2.2.3 CUPED

CUPED utilizes pre-experiment data to reduce the variability of the target metric and achieve better sensitivity (Deng et al., 2013). CUPED uses a single covariate to adjust the target outcome. The choice of covariate is open. Generally, using pre-experiment data of a target metric as a covariate provides the most significant variance reduction (Deng et al., 2013).

Following the formulation from CUPED (Deng et al., 2013), the estimator is expressed as:

$$Y_{cuped} = \bar{Y} - \theta\bar{X} + \theta\mathbb{E}[X] \tag{2}$$

where Y is the target metric, X is the covariate, θ is any constant, and Y_{cuped} is unbiased estimator of $\mathbb{E}[Y]$ because $-\theta\mathbb{E}[\bar{X}] + \theta\mathbb{E}[X] = 0$, and the variance of the unbiased estimator Y_{cuped} is given by:

$$\begin{aligned} \mathrm{var}(Y_{cuped}) &= \mathrm{var}(\overline{Y} - \theta\overline{X}) = \frac{1}{n}(\mathrm{var}(Y - \theta X)) && (3) \\ &= \frac{1}{n}\left(\mathrm{var}(Y) + \theta^2\mathrm{var}(X) - 2\theta\mathrm{cov}(Y, X)\right). && (4) \end{aligned}$$

To minimize $var(Y_{cuped})$, the choice of θ is as follows:

$$\theta = \frac{cov(Y, X)}{var(X)} \tag{5}$$

where $cov(Y, X)$ is the covariance between the target metric and the covariate, and $var(X)$ is the variance of the covariate. The coefficient θ adjusts the target metric based on the relationship between the covariate and the target metric.

With the optimal choice of θ, the variance of the unbiased estimator Y_{cuped} is given by:

$$var(Y_{cuped}) = var(\bar{Y})(1 - \rho^2) \tag{6}$$

Where ρ is the correlation between Y and X, $\rho = corr(Y, X)$.

X is an independent variable that is used to adjust the target metric, and to reduce variance effectively, it is crucial to select the covariate X with:

- High correlation with Y.
- Known $\mathbb{E}[X]$ value.

2.2.4 CUPAC

The CUPAC VR method extends CUPED to identify the partial correlation between multiple covariates and target metric (Li et al., 2020). Similar to CUPED, CUPAC uses pre-experiment data to adjust the target outcome. However, instead of using a single variable as a control variate, a model is trained with data before the experiment, and used to predict the target metric during the experiment. Next, the model prediction is used as a control variate to adjust the target metric.

The formulation of CUPAC is as follows (Li et al., 2020):

$$Y_{cupac} = Y - \theta g(X) + \theta \mathbb{E}[g(X)] \tag{7}$$

where Y is the target outcome, and $g(X)$ is the model prediction.

The variance of the estimator Y_{cupac} is given by:

$$var(Y_{cupac}) = \frac{1}{n}(var(Y) + \theta^2 var(g(X)) - 2\theta cov(Y, g(X)))$$

Similar to CUPED, to minimize $var(Y_{cupac})$ the coefficient θ is computed as:

$$\theta = \frac{cov(Y, g(X))}{var(g(X))}$$

And with the optimal choice of θ, the variance of the unbiased estimator Y_{cupac} is given by:

$$var(Y_{cupac}) = var(Y)(1 - \rho^2) \tag{8}$$

Where ρ is the correlation between Y and $g(X)$, $\rho = corr(Y, g(X))$.

2.2.5 MLRATE

MLRATE uses ML predictors for variance reduction (Guo et al., 2022). MLRATE accounts for the correlation between a target outcome and multiple covariates using ML. The model input features contain a list of covariates X that are correlated with the target metric and the in-experiment metric as a target Y. MLRATE employs cross-fitting to avoid overfitting biases and is robust to poor model predictions(Guo et al., 2022). Supposing the predictions are uncorrelated with the outcomes. In that case, the MLRATE estimator performs no worse than the difference-in-means estimator(Guo et al., 2022). On the other hand, if predictions are highly correlated with outcomes, the rate of variance reduction is high. In experiments conducted at Facebook, the MLRATE estimator has over 70% lower variance compared to the difference in means estimator (Guo et al., 2022).

The linear regression-adjusted estimator of treatment effect for MLRATE(Guo et al., 2022) is expressed as:

$$Y_i = \alpha_0 + \alpha_1 T_i + \alpha_2 g(X_i) + \alpha_3 T_i(g(X_i) - \mathbb{E}[g(X)]) + \epsilon_i \tag{9}$$

where Y is the target outcome, X represents the list of covariates, $g(X)$ is the model prediction. The term $\mathbb{E}[g(X)]$ is the average model prediction, and T_i is a treatment indicator that indicates whether the participant is in the treatment or not. α_0 is the intercept, α_1 is the treatment effect, α_2 is the effect of model prediction $g(X_i)$ on Y_i, α_3 is the interaction effect between treatment T_i and $g(X_i) - \mathbb{E}[g(X)]$, and ϵ_i is the error term.

2.3 Model selection

Different families of ML models can be used for the estimators. The choice of the model depends on the specific task and the data available. In this work, we use a Gradient Boosted Decision Trees (GBDT) model with different hyperparameter tuning and workflows to optimize the model performance.

2.4 Model objective functions

ML-based VR is a regression problem that estimates the treatment effect with the least variance by adjusting the target outcome using pre-experiment covariates. In this section, we define different regression objective functions that can be used to train ML models for VR. The choice of the objective function is crucial, as it directly affects the model's performance.

- Mean Absolute Error (MAE) measures the average absolute difference between the predicted and actual values. It is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{10}$$

  where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of samples.

- Mean Squared Error (MSE) measures the average squared difference between the predicted and actual values. It is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

  where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of samples. MSE is sensitive to outliers and can be affected by extreme values.

- Root Mean Squared Error (RMSE) is the square root of the mean squared error. It is defined as:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{12}$$

$$= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{13}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of samples.

Selecting an appropriate objective function depends significantly on the particular task under consideration. In cases where it remains unclear which objective function would be most suitable, evaluating and comparing different objective functions empirically through controlled experimentation is advisable.

2.5 Limitations

Methodologies such as CUPED only take into account the relationship between one pre-experiment covariate and the target business metric. We address this limitation by applying CUPAC and MLRATE to ride-hailing experiments. We leverage machine learning to identify the partial relationship between multiple pre-experiment covariates and the target business metric. These approaches allow us to reduce variance in OCEs effectively.

3 Methodology

This chapter provides an overview of the end-to-end pipeline, a description of the data utilized for the research, outlining the preprocessing techniques, the models developed, and the specific training methodologies employed.

Our experiments consist of several explorations and sequential steps. The starting point of OCE is the formulation of a hypothesis, followed by the design and setup of the experiment. After setting up the experiment, the variants are deployed, and the experiment starts. After the experiment is started, data is collected for 1 week, and an interim analysis is conducted using the experiment data to validate the variance estimation. Finally, when the experiment concludes, the data is analyzed to estimate the treatment effect and determine whether to accept or reject the hypothesis. The entire process is illustrated in Figure 3.

VR methods such as CUPED, CUPAC, and MLRATE are integrated throughout the pipeline. Historical data are used to estimate variance and determine sample size during setup. At the interim stage, VR methods are applied to re-estimate variance, update the required sample size, and adjust the experiment duration using early experiment data. After the experiment, VR techniques are used to adjust the OEC and estimate the treatment effect.

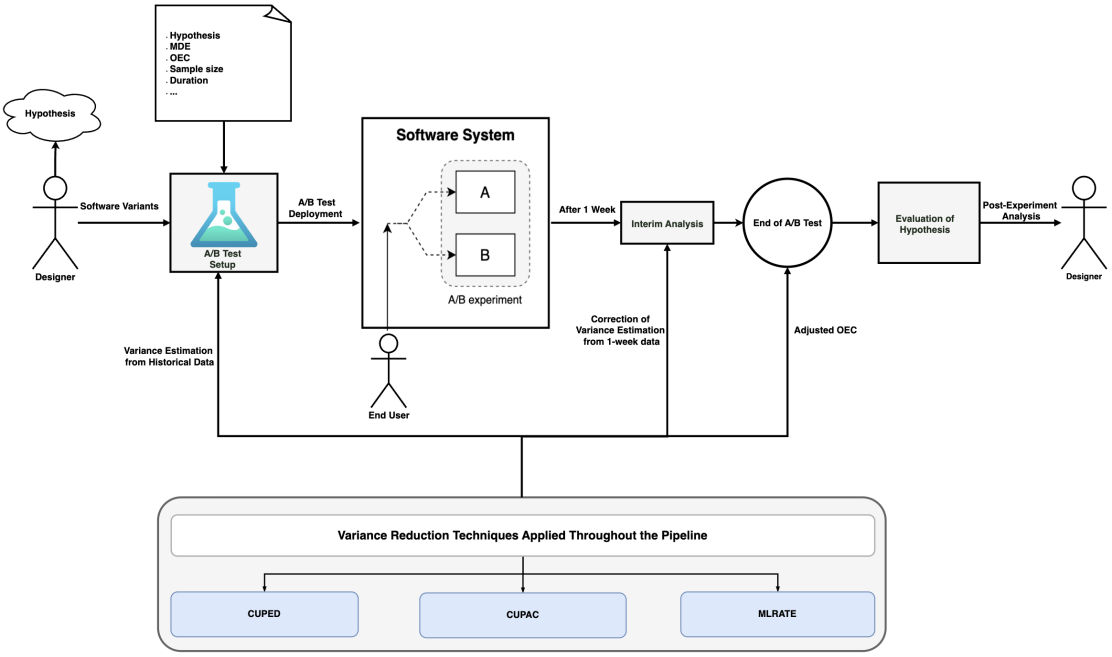

Figure 3. End-to-end workflow of an OCE incorporating VR techniques. The diagram presents the complete experimental lifecycle, from hypothesis formulation and A/B test setup to interim analysis and post-experiment evaluation.

To apply the VR methods on real-world ride-hailing OCEs, first, user-level experimental data is collected by Bolt Technology OÜ. The raw data undergoes feature engineering, which includes cleaning, normalization, and encoding of categorical features. Next, we develop predictive models for the three frequently used BMs. We explore two model pipelines for CUPAC and one for MLRATE. Following model development, three variance reduction methods are applied and compared: CUPED (used as a baseline), CUPAC (with two data/model setups), and MLRATE. Finally, the effectiveness of each variance reduction method is evaluated by measuring the percentage reduction in variance of the target BM.

3.1 Data Description

Real-world data is used to evaluate the effectiveness of the variance reduction methods applied to ride-hailing OCEs. The dataset is provided by Bolt Technology OÜ.

3.1.1 Real-World Data

The dataset contains information from experiments conducted in the ride-hailing business vertical in 2024 in Tallinn, Estonia. The dataset comprises anonymized user-level data

with no Personally Identifiable Information (PII) and experiment metric data.

3.1.2 Data Preprocessing and Feature Selection

For model training and evaluation, we identified the top 3 frequently used metrics in the ride-hailing experiments that use user-level randomization. Next, we use historical data and create a user profile. The user profile comprises 100+ features. Moreover, each business metric's feature set can differ, and we identify the most relevant features for each metric. We preprocess the data by creating a removal criterion as follows:

- **Domain knowledge:** This experiment focuses on the ride-hailing business vertical, so we remove features that are not relevant to this vertical.
- **Missing values:** features with too many missing values.
- **Low variance:** Non categorical features with low variance. We use a threshold of 0.01 to remove features with low variance.
- **Correlation:** Features with a Pearson correlation coefficient above 0.85 were identified as highly correlated (see Figure 4), and the feature that has the least mutual information with the target metric is removed.
- **Mutual information:** features with low mutual information with the target metric. Using a threshold of 0.01.

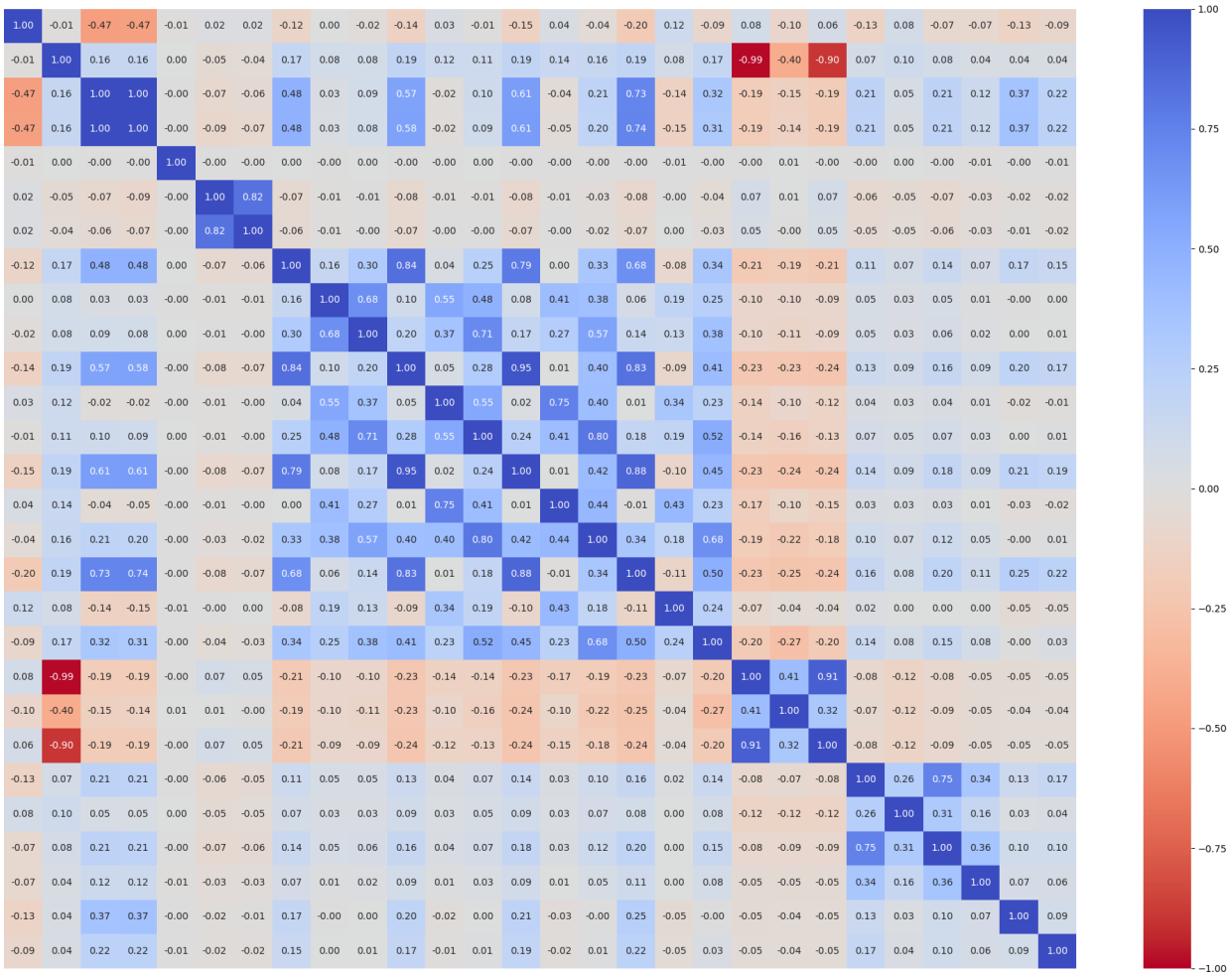

Figure 4. Correlation heatmap of features in the **Ride History** category (see Table 1), with values ranging from -1 (strong negative correlation) to 1 (strong positive correlation). Dark blue indicates a strong positive correlation, white indicates little to no correlation, and red indicates a strong negative correlation.

Additionally, we apply normalization and one-hot encoding on categorical features.

As discussed in the disclaimer section , we refer to the top 3 frequently used business metrics as **BM-I**, **BM-II**, and **BM-III**. Moreover, for the set of features, we prepared 11 categories shown in Table 1. Each category contains a group of features.

No.	Feature Category	No. of Features
1	User Metadata	8
2	Account Lifecycle	4
3	Account Status and Flags	15
4	Communication Preferences	8
5	Ride History	28
6	Order History	7
7	User Spending Stats	14
8	Engagement and Activity	17
9	Segmentation and Cohorts	5
10	Data Change & Ingestion	6
11	Marketing Consent	15

Table 1. Feature categories for grouping a related set of features.

3.2 Gradient Boosted Decision Trees

We use LightGBM (Ke et al., 2017) for our GBDT implementation. To enhance LightGBM model performance, we performed hyperparameter tuning using the Hyperopt library (Bergstra et al., 2013). Table 2 shows the optimal hyperparameter settings for the three business metrics we selected in this study.

Hyperparameter	ML Models		
	Business Metric-I Model	**Business Metric-II Model**	**Business Metric-III Model**
boostingType	gbdt	gbdt	gbdt
lambdaL2	0.02	1.5	1.43
learningRate	0.03	0.28	0.25
numIterations	170	200	280
numLeaves	300	200	150
objective	tweedie	tweedie	tweedie
tweedieVariancePower	1.04	1.77	1.74

Table 2. Hyperparameter settings for business metric level ML models.

- **boostingType**: The type of boosting to use. We use GBDT.
- **lambdaL2**: The L2 regularization.
- **learningRate**: The learning rate for the model.
- **numIterations**: The number of boosting iterations.

- **numLeaves**: The maximum number of leaves in one tree.
- **objective**: The objective for the regression model. The set of possible values are: `regression_l2`, `regression_l1`, `huber`, `fair`, `poisson`, `quantile`, `mape`, `gamma` or `tweedie`.
- **tweedieVariancePower**: The variance of tweedie distribution, and the value must be between 1 and 2.

The hyperparameter tuning process begins with feature selection and engineering on historical OCE data. The resulting dataset is used in a K-fold cross-validation setup ($K = 5$), where each fold serves as the validation set once to ensure robust performance evaluation. For each hyperparameter configuration, the average RMSE across the validation folds is computed and used as the optimization metric. We use MLFlow to track the hyperparameters and evaluation metrics(see Figure 18). The end-to-end tuning workflow is illustrated in Figure 5.

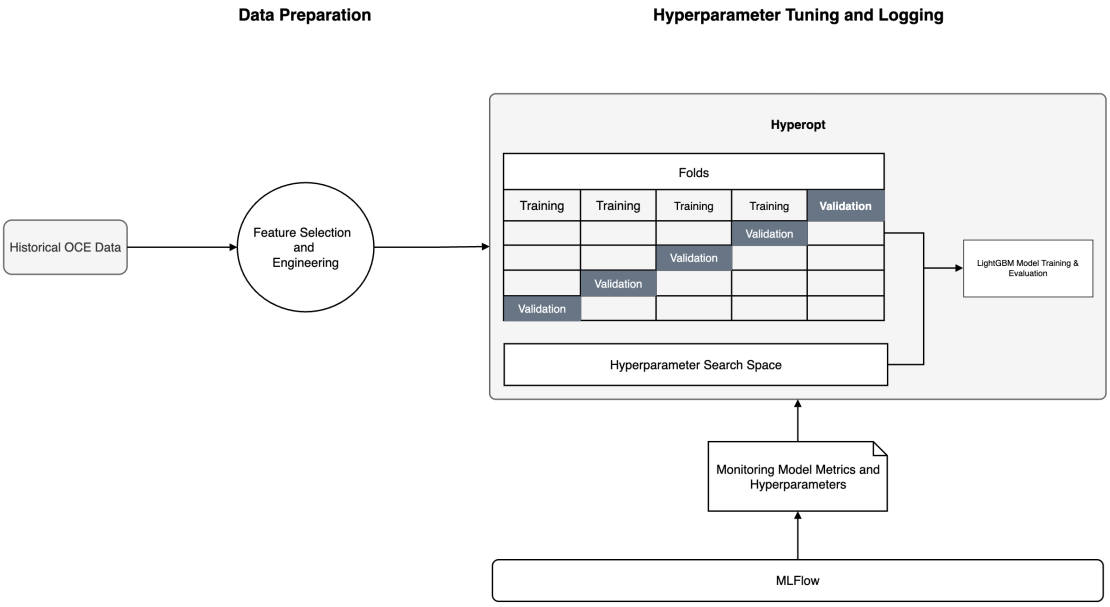

Figure 5. Hyperparameter tuning for the three business metrics.

3.3 Linear Regression

For MLRATE, the model predictions are first prepared using the GBDT model. Next, we use Ordinary Least Squares (OLS) to estimate the treatment effect in an experiment, compute the prediction (α_2) and treatment interaction coefficients (α_3) as discussed in section 3.10.

3.4 Model Pipeline

We develop one model for each BM per OCE. We used a different set of features for each BM model. The model development pipeline begins with data extraction from OCE, separated into pre-experiment and experiment datasets. Following this, a feature engineering step transforms and prepares the data. The full dataset is then partitioned into training and holdout sets. The holdout set is reserved for final performance evaluation. Within the training set, K-fold cross-validation ($K = 5$) is used to assess model generalization ability. For each fold, a different subset of the data is used for validation while the remainder is used for training. The average performance across folds is measured using RMSE. Once cross-validation is complete, the model is evaluated on the holdout set to estimate real-world performance. Both average cross-validation results and holdout performance are tracked using MLFlow, which manages model metrics and artifacts. The final model is then prepared using the full dataset. Figure 6 shows the end-to-end model pipeline.

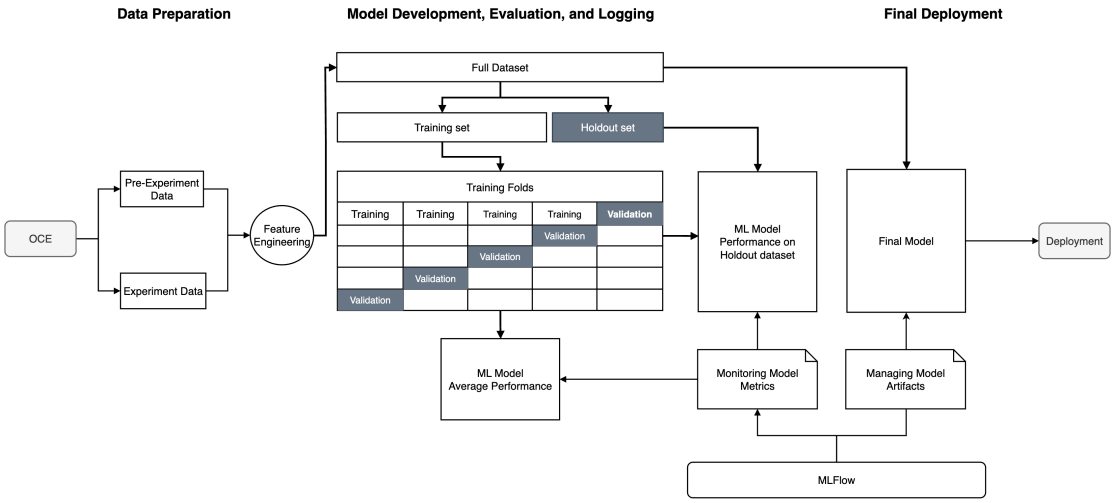

Figure 6. A high-level overview of the model pipeline consisting of data preparation, feature selection, model development, and model evaluation.

It is important to note that this pipeline applies to the GBDT model and not the OLS model.

3.5 Tools and Technologies

All models were implemented using Python version 3.11.9. GBDT was implemented using LightGBM, scikit-learn (Pedregosa et al., 2011) is used in data processing, while OLS regression was carried out using statsmodels (Seabold and Perktold, 2010). For

distributed data processing, we used PySpark (The Apache Software Foundation, 2025; Zaharia et al., 2010), Databricks, and SynapseML (Microsoft, 2024). Hyperparameter optimization was performed using Hyperopt (Bergstra et al., 2013). Data visualization tasks were accomplished using matplotlib (Hunter, 2007) and seaborn (Waskom, 2021). Diagrams illustrating the model pipelines are prepared using DrawIO (Ltd., 2025). For monitoring, logging, and managing the ML pipeline, we used MLflow (Chen et al., 2020).

All training and experimental procedures were conducted within a Databricks environment, configured with 8 worker nodes. Each worker node provided 8 CPU cores and 61 GB of memory. The exact number of worker nodes used varied according to the computational requirements of each experiment and was managed by the cluster autoscaling functionality.

ChatGPT-4 (OpenAI, 2025) was used as a coding assistant. Grammarly (Grammarly Inc., 2025) was utilized to correct grammatical errors and improve vocabulary usage.

3.6 Baseline

In this research, our primary focuses are CUPAC and MLRATE, but to analyze their performance, we use the existing data pipeline using CUPED as a baseline.

3.7 CUPED

We use CUPED as a reliable baseline to compare the performance of CUPAC and MLRATE. CUPED is a traditional method that uses historical data to adjust the target metric.

As discussed in section 2.2.3, the best explainer of a business metric Y is the pre-experiment value of the same metric.

$$X = Y_{pre-experiment} \tag{14}$$

Historical Average (HA) of the BM is used as a control variate in this study.

3.8 CUPAC

As discussed in section 2.2.4, CUPAC uses the output of a model prediction as a control variate (Li et al., 2020). After feature selection, preprocessing, and model development, we apply CUPAC with the GBDT model in ride-hailing experiments and compare it with the baseline method.

Algorithm 1: CUPAC: Variance Reduction using Model Prediction

Input: Covariate vectors X_{T-2}, X_{T-1}; metric value Y_{T-1}; target metric Y_T
Result: Adjusted target metric $\tilde{Y}_T$

1 Construct training dataset $\mathcal{D} = \{(X_{T-2}^{(i)}, Y_{T-1}^{(i)})\}_{i=1}^{n}$
2 Train predictive model $g(X)$ on dataset $\mathcal{D}$
3 Generate $g(X_{T-1})$ predicting Y_T given X_{T-1}: $\hat{Y}_T = g(X_{T-1})$
4 Compute $\overline{g(X_{T-1})}$ as $\overline{g(X_{T-1})} = \frac{1}{n}\sum_{i=1}^{n} g(X_{T-1}^{(i)})$
5 Compute θ as $\theta = \frac{cov(Y_T, \hat{Y}_T)}{var(\hat{Y}_T)}$
6 Compute adjusted metric $\tilde{Y}_T = Y_T - \theta\hat{Y}_T + \theta\overline{g(X_{T-1})}$

In algorithm 1, $T-2$ and $T-1$ represent the pre-experiment period.

Within the CUPAC framework, we develop CUPAC-I and CUPAC-II as distinct variants to explore different modeling approaches independently.

3.8.1 CUPAC Setup I (CUPAC-I)

We create three checkpoints: $T-2$, $T-1$, and T. Both $T-2$ and $T-1$ indicate a time before starting the experiment, and T indicates a time after starting the experiment.

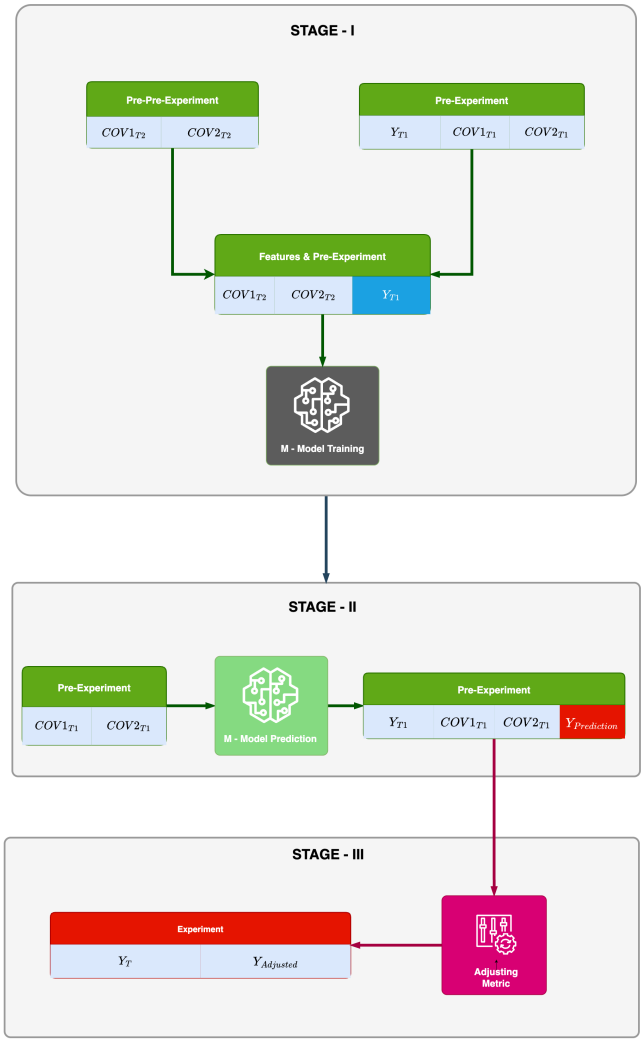

Figure 7. CUPAC-I variance reduction using a list of covariates as model input

- **Stage I:** We prepare the model training dataset by combining the list of covariates at T-2 with the metric value at T-1. Then, the dataset is used to train and evaluate a model.
- **Stage II:** We generate a model prediction using the model trained at Stage I and the list of covariates at T-1.
- **Stage III:** Finally, the target BM is adjusted using the model prediction from Stage II.

3.8.2 CUPAC Setup II (CUPAC-II)

From CUPED, we know that the best explainer of a given metric is the pre-experiment value of the same metric. This method is similar to the first approach; the key difference is that the historical metric value is also included as a model input feature.

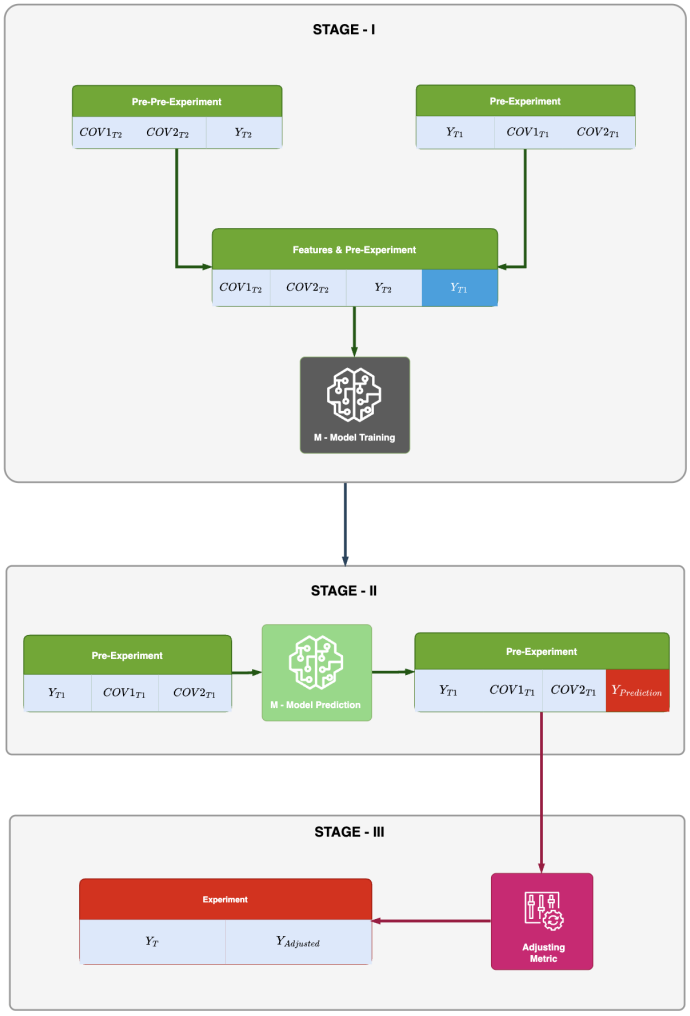

Figure 8. CUPAC-II variance reduction using a list of covariates and pre-experiment metric as model input feature

- **Stage I:** We prepare the model training dataset by combining the list of covariates and metric values at T-2 as features and the metric value at T-1 as a target. Then, the dataset is used to train and evaluate a model.
- **Stage II:** generate model predictions using the model trained at Stage I. Here, the model input features are the list of covariates and metric values at T-1.

- **Stage III:** Finally, the target primary metric is adjusted using the model prediction from Stage II.

3.9 MLRATE

In our implementation of MLRATE, we used pre-experiment features as model input, and the BM values during the experiment as a prediction target. Cross-fitting is applied to avoid overfitting and introducing bias (Guo et al., 2022).

We divide the MLRATE process into four stages:

- **Stage I:** We prepare the dataset by combining the list of features before the experiment starts, and the target metric value after starting the experiment.

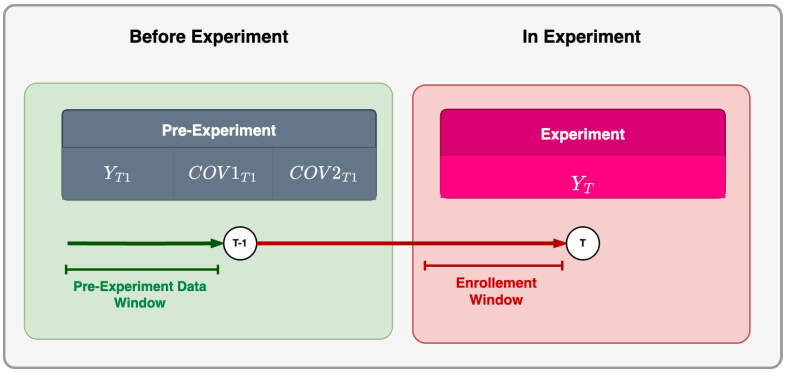

Figure 9. **Stage I:** Dataset preparation by combining pre-experiment features with experiment target metric values.

- **Stage II:** create N partitions of the data prepared at Stage I. Then, we train N models. The number of partitions used to train one model is $N-1$, then the model is used to predict the remaining partition. Figure 10 shows two models, M_1 and M_2, trained on different partitions, $Partition_1$ and $Partition_2$, respectively.

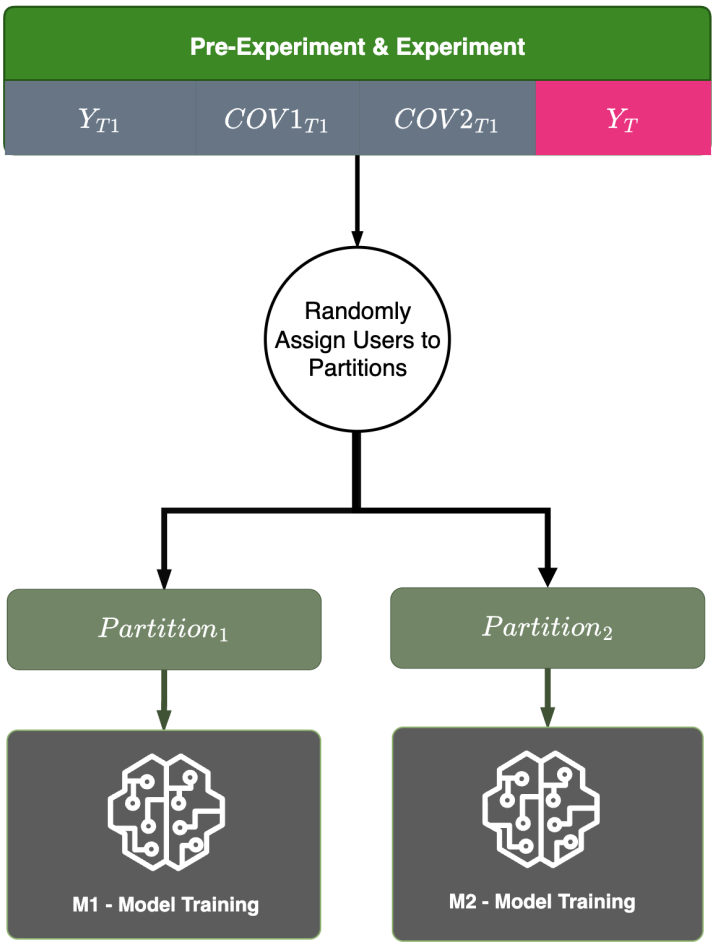

Figure 10. **Stage II:** The dataset is randomly split in to $N = 2$ partitions. Subsequently, two models are trained on the different partitions.

To avoid overfitting, we apply cross-fitting in our model training and prediction process. Users are randomly assigned to one of N partitions using a pseudo-random hash of an anonymous user ID (Kohavi et al., 2009, 2013). We use MD5 (Rivest, 1992) to generate the hash of the user ID. Next, we apply a modulo operation. The randomization approach ensures balanced representation of user characteristics (e.g., device types) across partitions.

Algorithm 2: Cross-Fitting: Model training

Input: Number of partitions N, user ID list $\mathcal{U}$

Divide the range $[0, 10^{10})$ into N equal intervals $\{I_1, I_2, \ldots, I_N\}$

for *each user* $u \in \mathcal{U}$ **do**

  Compute hash: $h_u = \text{MD5}(u) \bmod 10^{10}$

  Assign user u to partition P_i such that $h_u \in I_i$

for *each partition* $i = 1$ *to* N **do**

  Train model M_i on users not in partition P_i

  Store interval I_i as metadata for model M_i

- **Stage III:** At this stage, the model makes predictions on the partition not used in training. For example, model M_1 and M_2 are trained on $Partition_1$ and $Partition_2$, respectively. Therefore, M_1 is used to predict $Partition_2$, and M_2 is used to predict $Partition_1$ (see Figure 11).

Algorithm 3: Cross-Fitting: Model prediction

Input: Number of partitions N, user ID list $\mathcal{U}$

Divide the range $[0, 10^{10})$ into N equal intervals $\{I_1, I_2, \ldots, I_N\}$

Load models $\{M_1, M_2, \ldots, M_N\}$ for each interval I

for *each user* $u \in \mathcal{U}$ **do**

  Compute $h_u = \text{MD5}(u) \bmod 10^{10}$

  Identify partition P_i such that $h_u \in I_i$

  Predict $g(X_u) = M_i(X_u)$

After applying the prediction to the different partitions, the partitions are combined to form the final dataset of features and model predictions.

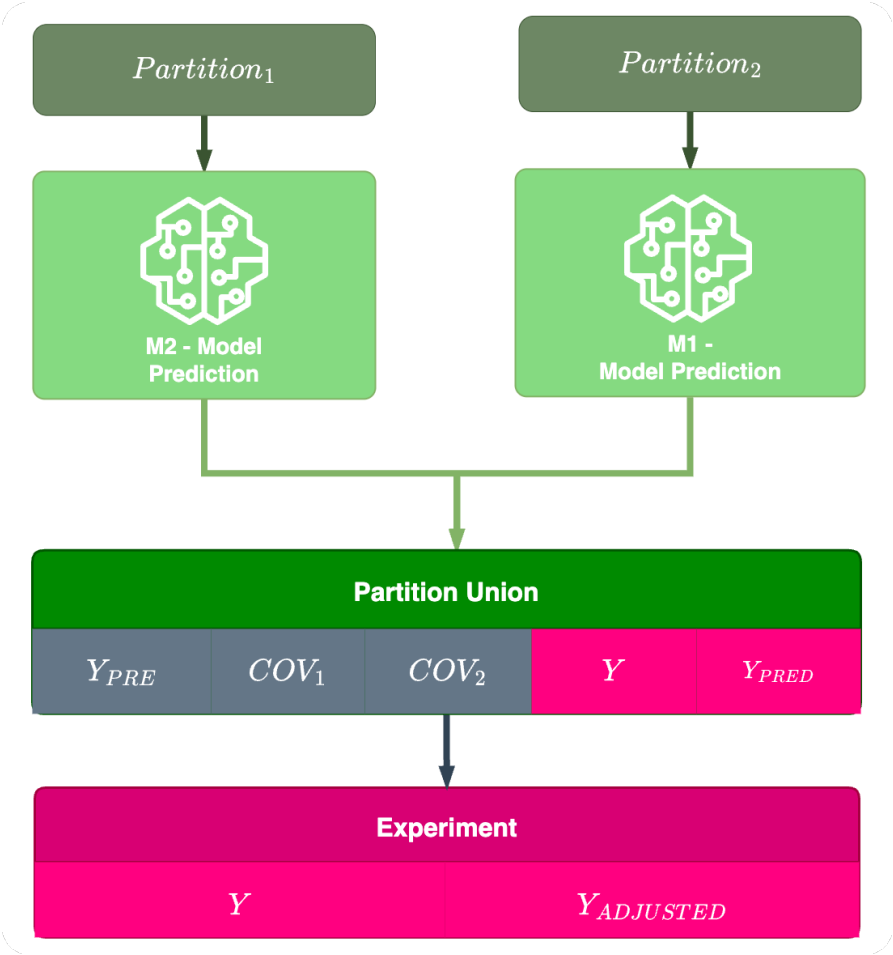

Figure 11. **Stage III:** Model M_1 trained on $Partition_1$ is used to make prediction on $Partition_2$, and model M_2 trained on $Partition_2$ is used to make prediction on $Partition_1$.

- **Stage IV:** Finally, the BMs are adjusted using the combined dataset in a linear regression step. We fit an OLS model using the model prediction and treatment indicator features as discussed in section 2.2.5 to estimate the treatment effect.

3.10 Enrollment Rate

The enrollment rate is the ratio of users who are enrolled in the experiment after the experiment starts. The enrollment rate is a key factor in determining the experiment duration for a given sample size. We compute the daily enrollment rate as follows:

$$E = \frac{U_e}{U} \tag{15}$$

Where U_e is the number of users enrolled in the experiment in a day, and U is the total number of users required for the experiment to reach the desired sample size.

The enrollment rate changes with time. We analyze the enrollment rate of experiments to determine the impact of the variance reduction methods on the experiment duration and the business.

4 Experiment Results

In this chapter, we describe our experiments and the evaluation procedure for ML models, which consists of computing regression metrics such as MSE, RMSE, MAE, and graphical distribution analysis. Moreover, we describe the effectiveness of variance reduction methods on the different BMs. We also analyze the enrollment rate of experiments to determine the impact of the variance reduction methods on the experiment duration.

4.1 Main Results

We conducted our experiments using the dataset described in Section 3.1.1. The results of the variance reduction methods are summarized in Table 3. The table shows the rate of variance reduction achieved by each method across three different business metrics. The MLRATE method achieved the highest variance reduction across all metrics, followed closely by CUPAC-II. The CUPED method showed the least variance reduction among all methods.

Method	**Variance reduction (%)**		
	BM-I	**BM-II**	**BM-III**
CUPED	44.6	44.2	32.4
CUPAC-I	49.3	49.2	27.5
CUPAC-II	50.1	50.3	28.5
MLRATE	**51.2**	**50.4**	**32.8**

Table 3. Percentage of variance reduction achieved by different methods across three BMs. The highest variance reduction for each metric is shown in bold.

All methods perform poorly for BM-III, compared to BM-I and BM-II. Upon further investigation, we found that the model's prediction power for BM-III is lower than BM-I and BM-II (see Figure 15), and it is caused by the limited pre-experiment data.

4.2 Gradient Boosted Decision Trees

We evaluated the importance of the feature for the Gradient Boosted Decision Trees models using two approaches. First, we used the standard feature importance method from the SynapseML Python library (Microsoft, 2024), which computes importance based on the total gain from splits involving each feature (see Figure 12).

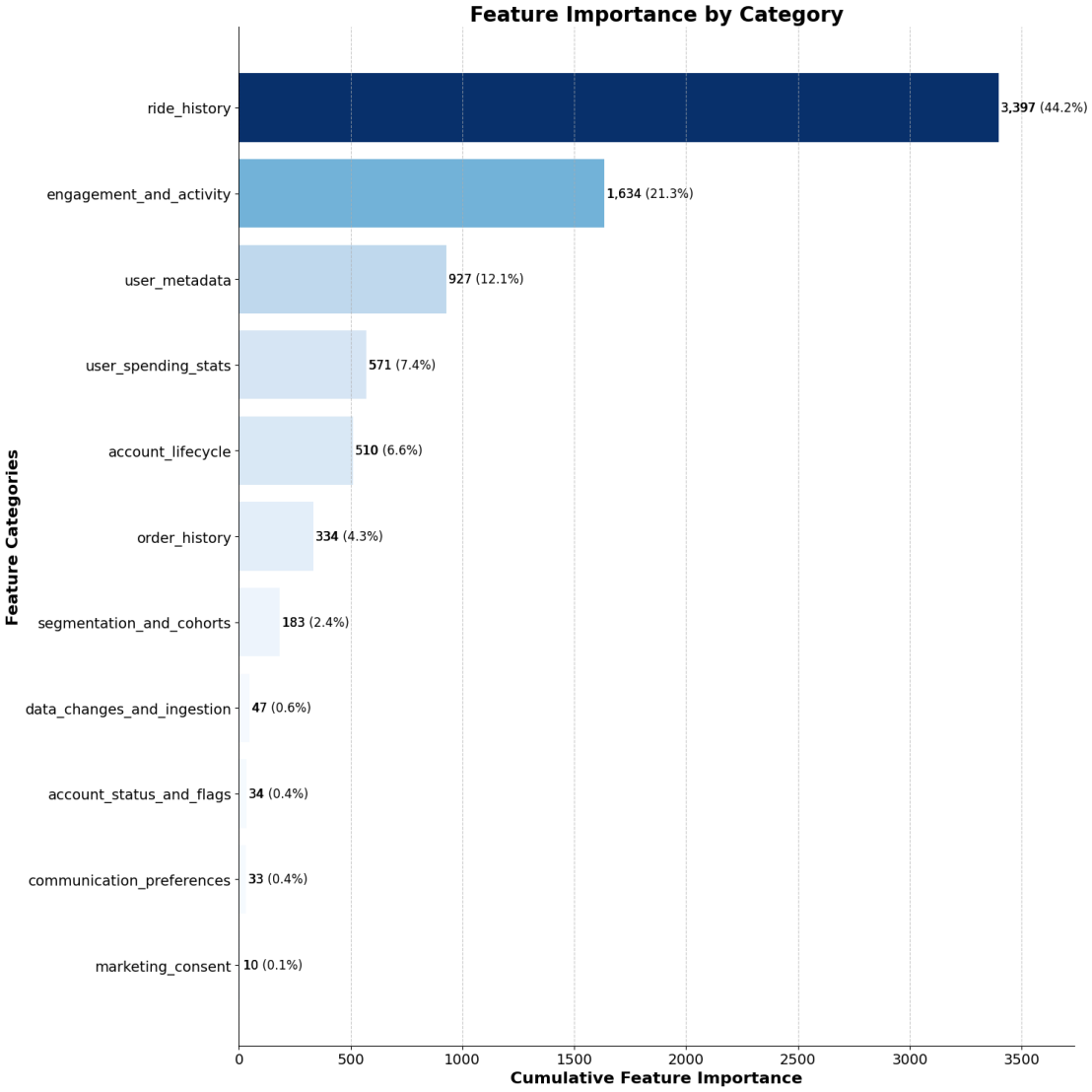

Figure 12. LightGBM cumulative feature importance by category using SynapseML's native function. Feature importance values are aggregated by category, summing the importance scores of all features within each category.

Additionally, we applied the SHAP method (Lundberg and Lee, 2017) to provide a more interpretable measure of feature impact, as illustrated in Figure 13. Both approaches indicate that features related to ride history are the most influential across all three BMs.

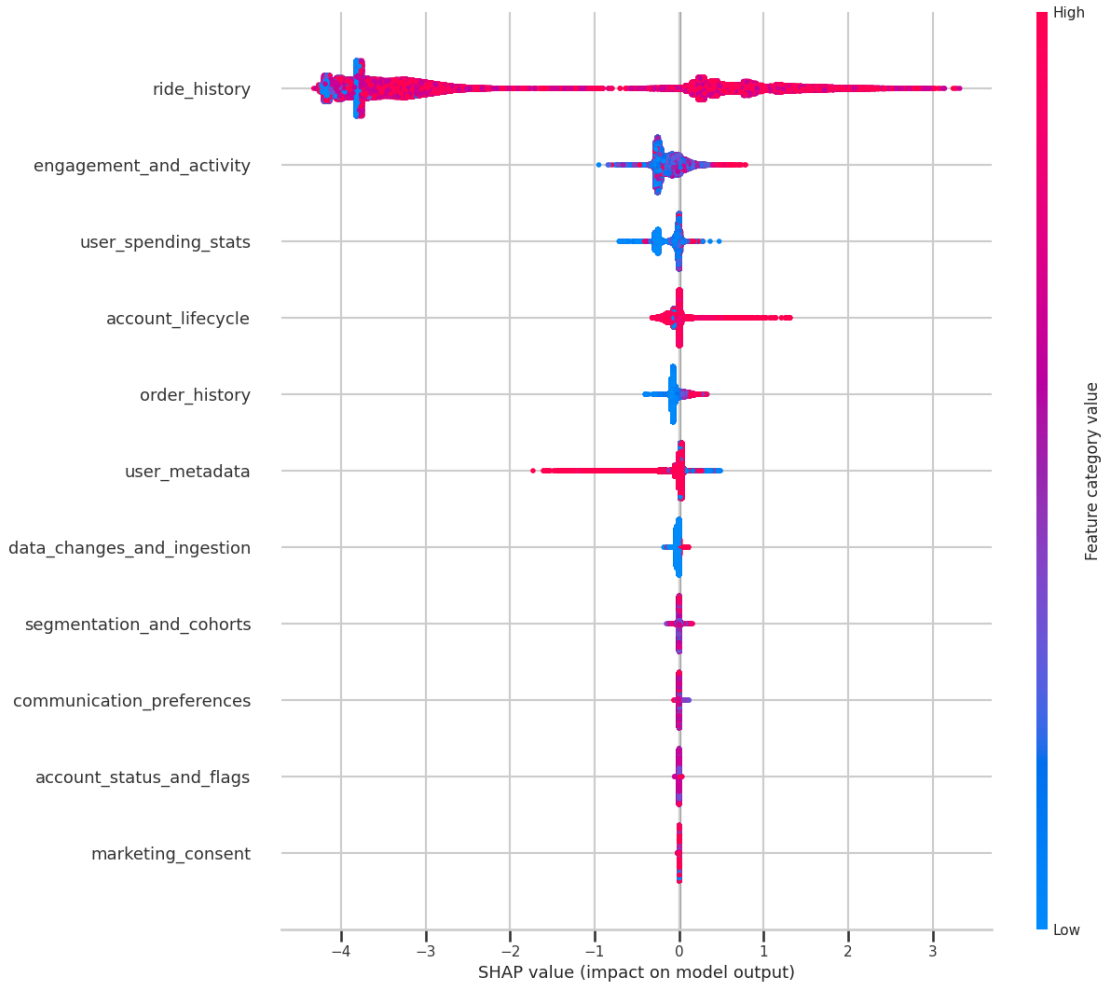

Figure 13. Category-level SHAP summary plot for a LightGBM model. SHAP values for individual features are aggregated by category to reflect group-level importance. Color indicates the mean value of features within each category (red = high, blue = low).

For each BM, we monitor model evaluation metrics, including cross-validation average metrics and performance on the holdout dataset. Table 4 shows the performance of models for the different VR methods as measured using MAE and RMSE. As we can see, the MLRATE models perform best on the holdout dataset, which shows the effectiveness of the cross-fitting technique. The second-best models are the CUPAC-II models, which show that including the historical average of BM as a feature improves model performance.

Method	**Model Target BM**	**CV Average Metrics**		**Holdout Metrics**	
		MAE	RMSE	MAE	RMSE
CUPAC-I	BM-I	2.91	5.24	3.87	7.28
	BM-II	1.87	3.13	2.74	4.98
	BM-III	17.82	33.35	27.22	46.32
CUPAC-II	BM-I	2.06	3.12	2.49	6.14
	BM-II	1.65	2.64	2.26	4.41
	BM-III	18.88	33.90	27.06	45.13
MLRATE	BM-I	1.58	2.53	2.07	4.54
	BM-II	2.17	3.81	2.19	3.95
	BM-III	14.41	27.46	17.89	32.43

Table 4. CV and Holdout Metrics of GBDT model used for VR

To graphically analyze the distribution similarity between BM values and model predictions, we plot the histogram of the BM values and predictions in the holdout dataset. Figure 14 shows the histogram of BM values in the holdout dataset.

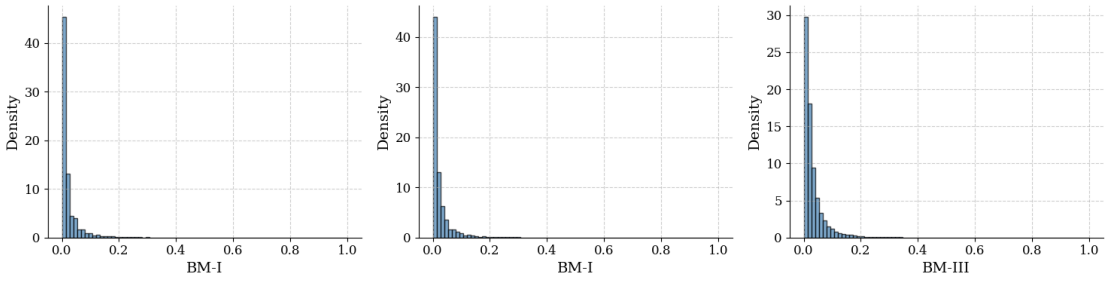

Figure 14. Histogram of BM values for the three business metrics in the holdout dataset.

The predicted BM values from the GBDT models for CUPAC-I, CUPAC-II, and MLRATE methods are shown in Figure 15. The histogram shows high distribution similarity

for BM-I and BM-II. However, the predicted distribution for BM-III shows a difference from the true BM-III distribution, which indicates the impact of limited data on the model's prediction power.

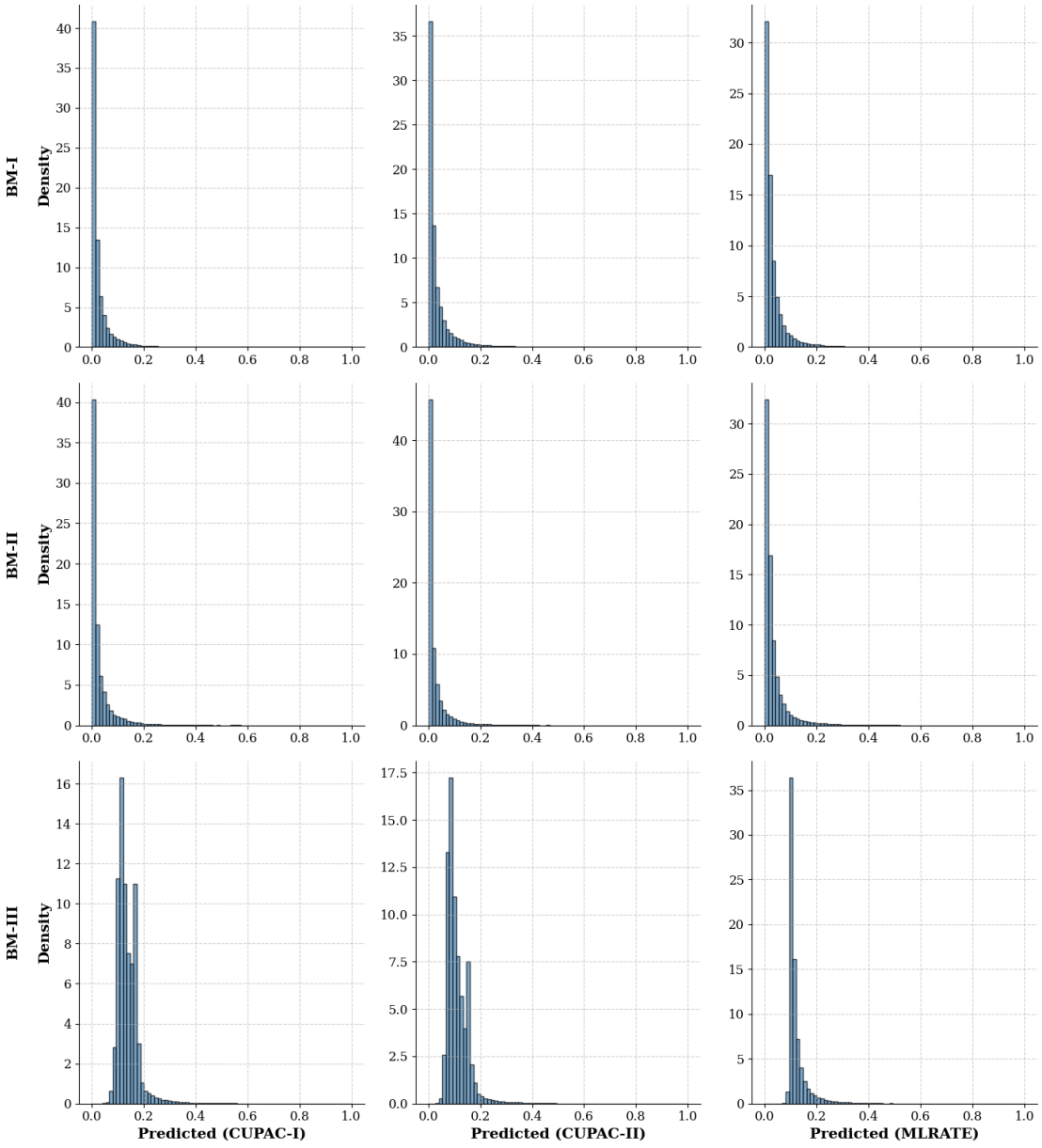

Figure 15. Histogram of normalized predicted BM values of ML models used for CUPAC-I, CUPAC-II, and MLRATE.

4.3 Enrollment Rate

To assess how variance reduction methods affect experiment duration, we examine the enrollment rate progression in OCEs. Typically, over 50% of users are enrolled within the first 7 days, with power users joining early in the experiment. As enrollment progresses, the pool of eligible users diminishes, resulting in a declining enrollment rate.

In short-duration experiments with relatively small sample size requirements, reaching the first 5% of the target sample is comparable to the time needed to get the final 5% (see Figure 16).

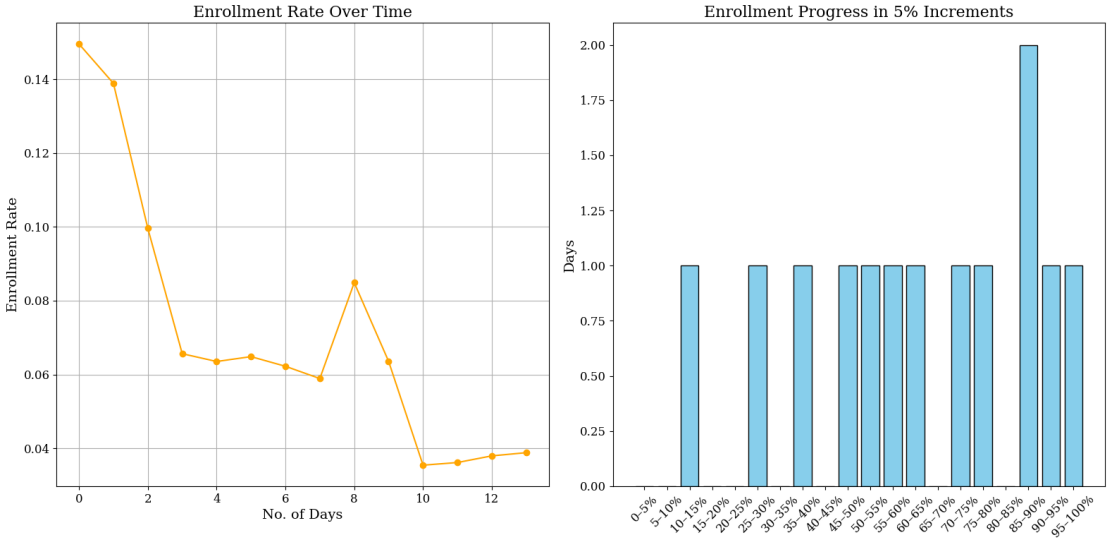

Figure 16. Enrollment rate of experiments lasting less than fifteen days. The sample size is calculated without VR.

In contrast, experiments that require larger sample sizes show a more noticeable slowdown in enrollment. The initial 5% of users may be enrolled in just one day, while the last 5% can take as long as 10 days (see Figure 17).

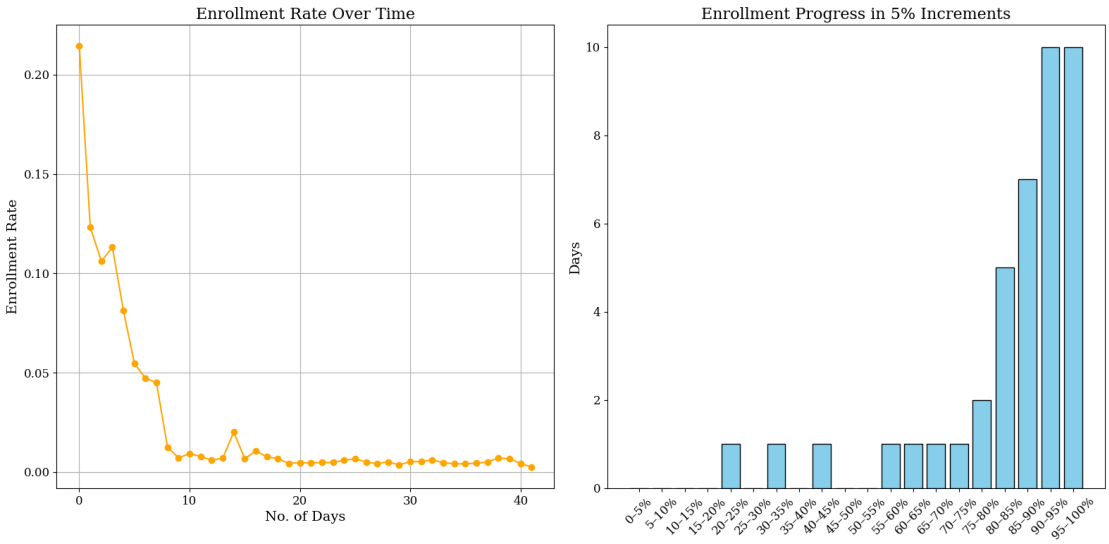

Figure 17. Enrollment rate of experiments lasting more than fifteen days. The sample size is calculated without VR.

The thesis does not report the average reduction in sample size. However, in our experiments, the variance reduction methods CUPAC and MLRATE decreased the required sample size by as much as 5% compared to the baseline method, CUPED. The practical impact of this reduction varies depending on the original duration of the experiment. We expect no significant time reduction for experiments running for less than two weeks (Figure 16), but a reduction of multiple days for longer experiments (Figure 17).

5 Discussion

Our experimental results demonstrate that VR methods utilizing ML algorithms can effectively capture partial correlations between covariates and business metrics and improve the rate of variance reduction in ride-hailing experiments. Specifically, MLRATE consistently outperformed baseline methods, including CUPED, CUPAC-I, and CUPAC-II, across key BMs, achieving approximately 51% variance reduction from the original metric variance.

OCEs measure different aspects of a business through various business metrics. Our study found that MLRATE performed best across all evaluated metrics. The CUPED baseline outperformed CUPAC-I and CUPAC-II methods in a specific metric where there was limited data available for training and prediction. Additionally, the performance of CUPAC on BM-III illustrates that the predictive power of a model can significantly influence the effectiveness of VR methods. Therefore, it is crucial to monitor model performance when applying VR methods that utilize ML.

We show that the relationship between the enrollment rate and time is not constant. Typically, enrollment rates are high at the start of the experiment and decrease over time, implying that for long-duration OCEs, enrolling the final 5% of the required sample size demands substantially more time than enrolling the initial 5%. Our experiments show an improvement in sample size and experiment duration compared to the CUPED baseline method.

Employing variance reduction techniques that incorporate ML at scale requires additional ML pipeline efforts and increased computational demands. In contrast, the baseline method is more straightforward and less resource-intensive but shows performance limitations. It is important to note that all approaches depend on the availability of historical data.

Our thesis only covers OCEs conducted in one city with user-level randomization, and further work is needed to evaluate these approaches at different randomization levels and multiple cities. Evaluations at device or session levels and other experimental designs, such as switchback tests, represent important areas for future research. Moreover, our experiment was conducted with a fixed set of features, and further feature engineering and hyperparameter tuning could yield even better model prediction performance and reduce experiment duration.

In summary, while adding complexity, adopting ML-based VR methods such as MLRATE offers compelling business advantages by reducing experiment duration, which facilitates the decision-making process.

6 Conclusion

In this thesis, we developed a data processing pipeline for preparing OCE data and adjusting primary experiment metrics. We trained and evaluated LightGBM models using covariates as inputs and the target metric as the prediction target. An evaluation framework was implemented to compare the effectiveness of VR methods. Our results show that the proposed pipeline machine learning-based VR method outperforms the baseline CUPED. However, this improvement introduces additional complexity to the experimentation workflow. Moreover, all approaches, CUPED, CUPAC, and MLRATE, depend on pre-experiment data availability. We believe that with further feature engineering and optimization, the variance reduction rate can be improved further, making the added complexity worthwhile.

7 Acknowledgements

The work was completed as part of the Industrial Masters Program with Bolt Technology OÜ. The author expresses gratitude for the help and support received from the company, particularly the Experimentation team at Bolt. Additionally, the author sincerely thanks his company supervisor, Carlos Bentes, and academic supervisor, Dr. Elena Sügis. Their invaluable guidance, support, and expertise were instrumental in completing this thesis.

References

Vahid Garousi and Mika V. Mäntylä. A systematic literature review of literature reviews in software testing. *Information and Software Technology*, 80:195–216, December 2016. ISSN 0950-5849. doi: 10.1016/j.infsof.2016.09.002. URL https://www.sciencedirect.com/science/article/pii/S0950584916301446.

Ron Kohavi and Roger Longbotham. Online Controlled Experiments and A/B Tests. In Dinh Phung, Geoffrey I. Webb, and Claude Sammut, editors, *Encyclopedia of Machine Learning and Data Science*, pages 1–13. Springer US, New York, NY, 2023. ISBN 978-1-4899-7502-7. doi: 10.1007/978-1-4899-7502-7_891-2. URL https://link.springer.com/10.1007/978-1-4899-7502-7_891-2.

Ron Kohavi, Diane Tang, and Ya Xu. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, 03 2020. doi: 10.1017/9781108653985.

Somit Gupta, Ronny Kohavi, Diane Tang, Ya Xu, Reid Andersen, Eytan Bakshy, Niall Cardin, Sumitha Chandran, Nanyu Chen, Dominic Coey, Mike Curtis, Alex Deng, Weitao Duan, Peter Forbes, Brian Frasca, Tommy Guy, Guido W. Imbens, Guillaume Saint Jacques, Pranav Kantawala, Ilya Katsev, Moshe Katzwer, Mikael Konutgan, Elena Kunakova, Minyong Lee, MJ Lee, Joseph Liu, James McQueen, Amir Najmi, Brent Smith, Vivek Trehan, Lukas Vermeer, Toby Walker, Jeffrey Wong, and Igor Yashkov. Top challenges from the first practical online controlled experiments summit. *SIGKDD Explorations*, 21(1), 2019. URL https://bit.ly/ControlledExperimentsSummit1.

Alex Deng, Ya Xu, Ron Kohavi, and Toby Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 123–132, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318693. doi: 10.1145/2433396.2433413. URL https://doi.org/10.1145/2433396.2433413.

Jun Li, Yao Tang, and James Bauman. Improving experimental power through control using predictions as covariate (cupac). *DoorDash Engineering Blog*, 2020. URL https://careersatdoordash.com/blog/improving-experimental-power-through-control-using-predictions-as-covariate-cupac/.

Yongyi Guo, Dominic Coey, Mikael Konutgan, Wenting Li, Chris Schoener, and Matt Goldman. Machine learning for variance reduction in online experiments, 2022. URL https://arxiv.org/abs/2106.07263.

Ceyhan Ceran Serdar, Murat Cihan, Doğan Yücel, and Muhittin A Serdar. Sample size, power and effect size revisited: simplified and practical approaches in pre-clinical, clinical and laboratory studies. *Biochemia Medica*, 31(1):010502, February 2021. ISSN 1330-0962. doi: 10.11613/BM.2021.010502. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7745163/.

Huizhi Xie and Juliette Aurisset. Improving the sensitivity of online controlled experiments: Case studies at netflix. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 645–654, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939733. URL https://doi.org/10.1145/2939672.2939733.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/bergstra13.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

The Apache Software Foundation. Pyspark: Python api for apache spark. https://spark.apache.org/docs/latest/api/python/, 2025. PySpark is the Python API for Apache Spark.

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. HotCloud'10, page 10, USA, 2010. USENIX Association.

Microsoft. Synapseml: Large-scale machine learning on apache spark. GitHub repository, 2024. URL https://github.com/microsoft/SynapseML.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

JGraph Ltd. draw.io is a javascript, client-side editor for general diagramming. https://www.draw.io/, 2025. draw.io is a JavaScript, client-side editor for general diagramming.

Andrew Chen, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, and Corey Zumar. Developments in mlflow: A system to accelerate the machine learning lifecycle. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, DEEM '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380232. doi: 10.1145/3399579.3399867. URL https://doi.org/10.1145/3399579.3399867.

OpenAI. Chatgpt: Ai large language model by openai. https://chat.openai.com, 2025. Large language model.

Grammarly Inc. Grammarly, 2025. URL https://www.grammarly.com. Free AI writing assistant.

Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal Henne. Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery*, 18:140–181, 02 2009. doi: 10.1007/s10618-008-0114-1.

Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 1168–1176, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2488217. URL https://doi.org/10.1145/2487575.2488217.

Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992. URL https://www.rfc-editor.org/info/rfc1321.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.,

2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Appendix

I. Acronyms

BM Business Metric.

CUPAC Control Using Predictions as Covariate.

CUPED Controlled Experiment Using Pre-Experiment Data.

GBDT Gradient Boosted Decision Trees.

HA Historical Average.

MDE Minimum Detectable Effect.

ML Machine Learning.

MLRATE Machine Learning for Variance Reduction in Online Experiments.

OCE Online Controlled Experiment.

OEC Overall Evaluation Criterion.

OLS Ordinary Least Squares.

PII Personally Identifiable Information.

VR Variance Reduction.

III. Visualizations

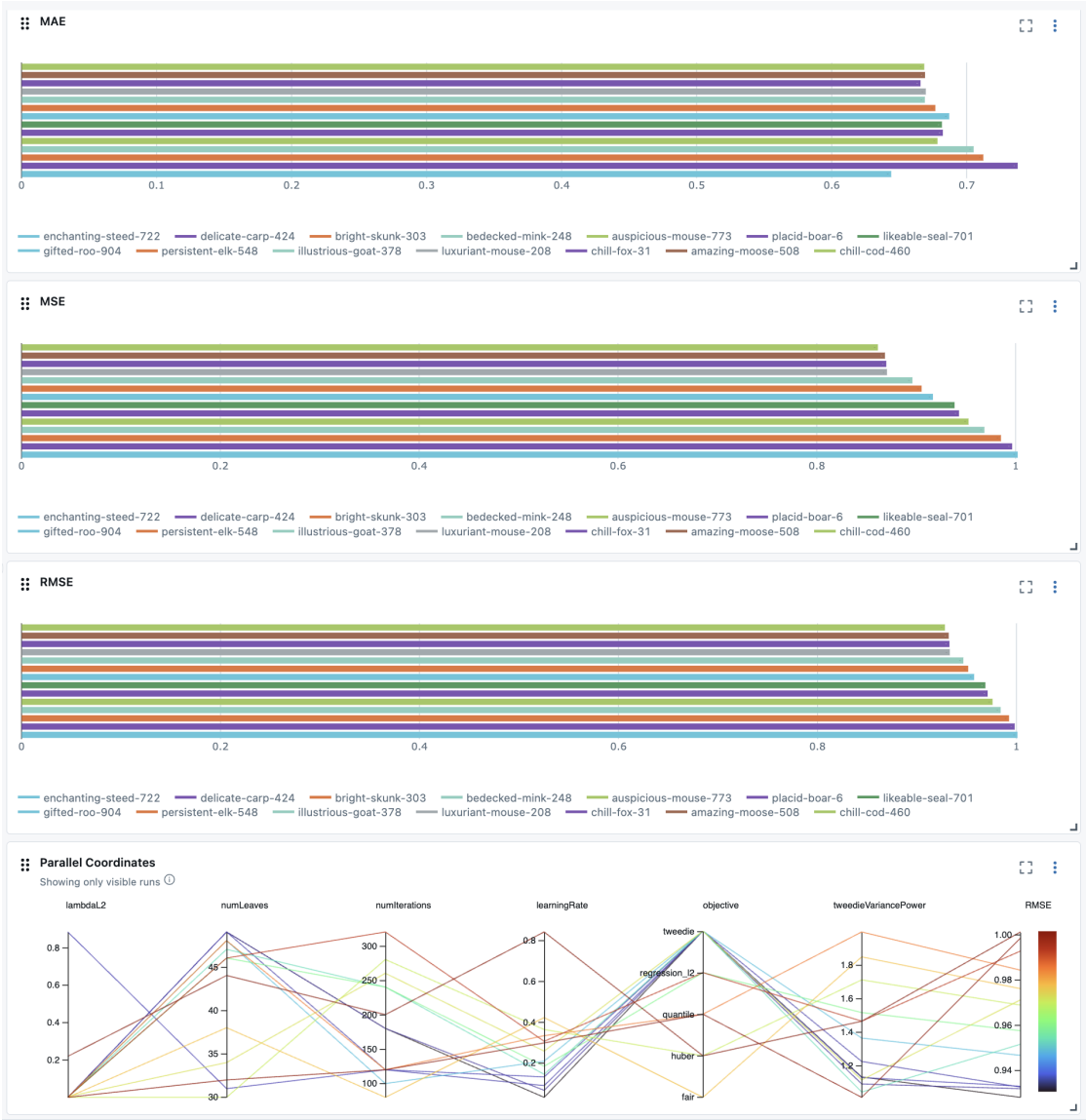

Figure 18. MLFlow monitoring model metrics during hyperparameter tuning.

IV. License

Non-exclusive license to reproduce thesis and make thesis public

I, **Bazen Teklehaymanot Tadele**,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to

   reproduce for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Variance Reduction In Online Controlled Experiments,

   supervised by Carlos Bentes and Dr. Elena Sügis.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe on other persons' intellectual property rights or rights arising from the personal data protection legislation.

Bazen Teklehaymanot Tadele
15/05/2025