Mitigating double-dipping in behavior-agnostic RL

Yaren Aslan[1], Stephan Bongers[1] and Frans A. Oliehoek[1]

[1]Delft University of Technology, Delft, the Netherlands

Abstract. This paper addresses the issue of double-dipping in off-policy evaluation (OPE) in behaviour-agnostic reinforcement learning, where the same dataset is used for both training and estimation, leading to overfitting and inflated performance metrics especially for variance. We introduce SplitDICE, which incorporates sample-splitting and cross-fitting techniques to mitigate double-dipping effects in the DICE family of estimators. Focusing specifically on 2-fold and 5-fold cross-fitting strategies, the original off-policy dataset is partitioned with random-split to get separate training and evaluation datasets. Experimental results demonstrate that SplitDICE, particularly with 5-fold cross-fitting, significantly reduces error, bias, and variance compared to naive DICE implementations, providing a more doubly-robust solution for behavior-agnostic OPE.

Keywords: DICE · off-policy evaluation · behaviour-agnostic · sample-splitting · cross-fitting · double-dipping · overfitting

1 Introduction

In reinforcement learning, for each state of the environment, the agent takes a certain action based on the policy, retrieves a reward based on this action and transitions into a new state. The overarching goal of reinforcement learning methodologies is to acquire an optimal policy which maximizes the long-term cumulative rewards [17]. In the context of policy evaluation, *off-policy evaluation* (OPE) refers to the setting where the agent estimates the value of a target policy by referring only to a dataset of experience previously collected by other policies in the said environment [16]. The objective of OPE is to estimate the expected cumulative (discounted) reward that a new policy (namely, the target policy) would achieve if deployed in the environment. This is important for understanding how well the new policy might perform before actually deploying it [21]. However, it is essential to note that this logged experience is collected by potentially multiple and possibly unknown behavior policies which requires the embodiment of the concept known as *behavior-agnostic*.

Behaviour-agnostic OPE specifically denotes an approach where the learning algorithm does not make any assumptions about the behavior policy that generated the dataset [23]. Recent advancements in addressing the unknown bounds of behaviour-agnostic OPE have led to the development of various estimators collectively referred to as the "DICE" family which stands for *DIstribution Correction Estimation* [12], [22], [26], [23]. These estimators are used to display the ratio between the propensity of the target policy to visit distinct state-action pairs compared to their occurrence likelihood

in the off-policy data. DICE-based estimators exercise “a single marginal ratio to re-weight the rewards for each state-action pair”, consequently achieving a relatively low variance for the estimate values [3].

It is worth highlighting that policy evaluation in general requires tedious consideration of the model complexity to prevent pitfalls such as overfitting or underfitting the data. Overfitting occurs when a modeling approach mirrors every underlying pattern of the data, resulting in high accuracy when applied to the original dataset. However, it fails to generalize well to unseen and foreign datasets. This is mainly caused by data with many features and/or an excessively large neural network. In this context, *double-dipping* refers to the practice of overfitting a model by building (training) and evaluating (estimating) it on the same dataset [2]. This then causes misleadingly high performance metrics with artificially inflated statistical significance since the model is being evaluated on the data it was trained on, leading to “circular logic”.

It is common in many DICE estimators to employ the same dataset to conduct the training process and the estimation of the target policy [23]. This practice is the driving factor for double-dipping. Another pressing issue related to the double-dipping behaviour is that the DICE family employs primal and dual regularization techniques as a regression method to tackle high variance and therefore to avoid overfitting [23]. The main concern regarding this choice is that this introduces a trade-off, as regularization can bias the parameter of interest with the aim of mitigating overfitting.

This paper explores how sample-splitting and cross-fitting techniques, when applied to DICE estimators, can mitigate the impact of double-dipping in behavior-agnostic reinforcement learning and therefore reduce variance in the estimate results. To break it down more clearly, the aim of this work is to adopt commonly used techniques in double/debiased machine learning (DML), namely *sample-splitting* and *cross-fitting*, for the DICE estimators and analyze how effective they are in mitigating the risks associated with double-dipping. DML’s "double" behaviour comes from simultaneously estimating two predictive models: one for the primary target of interest and another for auxiliary outcomes [7]. When compared against naive ML estimators, their fast rates of convergence and robust behaviour with respect to a broader class of probability distributions makes the objective of this study even more striking [7]. In this work, we introduce SplitDICE, which incorporates sample-splitting and cross-fitting into the existing implementation of the DICE estimator.

The paper begins with a background on integrating OPE, DICE, and DML in section 2, followed by a review of related research on double machine learning methods and DICE estimators, identifying improvements and this study’s contributions in section 3. It then outlines the technical methods, including sample-splitting and cross-fitting, in section 4. The experimental setup, including data generation and configurations, is detailed in section 5, and the results are analyzed in section 6, focusing on convergence, error, variance, and bias, with statistical significance noted for relative error. The paper concludes by discussing the results and limitations in section 7, and section 8 highlights the 5-fold cross-fit DICE estimator’s superior performance over the naive BestDICE in reducing error, bias, and variance.

2 Background

This section begins with background on reinforcement learning and policy evaluation, followed by the motivation for off-policy evaluation and the DICE estimators designed for behavior-agnostic settings. In section 2.1 and section 2.2, we introduce equations from Yang et al. [23]. We then discuss the double-dipping problem, the main focus of this study, and the proposed sample-splitting solution. Finally, we introduce SplitDICE, the estimator developed to address this issue, and explain sample-splitting and cross-fitting in section 2.3.

2.1 Policy Evaluation in Reinforcement Learning

In a reinforcement learning (RL) setting, we consider an infinite-horizon Markov Decision Process (MDP) [17]. This process is defined by the tuple $M = \langle S, A, R, T, \mu_0, \gamma \rangle$. This consists of a state space S, action space A, reward function R, transition probability function T, initial state distribution μ_0, and a discount factor $\gamma \in [0, 1)$. In RL, a policy π defines the agent's strategy or behavior in an environment. It is a mapping from states to actions, denoted as $\pi(a_t \mid s_t)$, which specifies the probability distribution over actions A that the agent selects when in state s at step $t \geq 0$. The environment yields a scalar reward $r_t = R(s_t, a_t)$ and then transitions to a new state $s_{t+1} \sim T(s_t, a_t)$.

We define the value of a policy π by the discounted cumulative reward it receives:

$$\rho(\pi) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 \sim \mu_0, \forall t, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)\right] \quad (1)$$

In RL, the behavior policy denoted as μ refers to the policy that the agent is currently following to interact with the environment. It is a distinct concept from the target policy π which the agent seeks to optimize and improve during its learning (training) process. Therefore, naturally, in the context of policy evaluation, the policy being evaluated is the target policy. In a more concrete way, with policy evaluation, we aim to evaluate how effectively the training of the behavior policy aligns with the target policy. The value of a target policy can be expressed equivalently in two manners using different functions [23]:

$$\begin{aligned} \rho(\pi) &= (1-\gamma) \cdot \mathbb{E}_{a_0 \sim \pi(s_0)} \left[Q^\pi(s_0, a_0)\right] \\ &= \mathbb{E}_{(s,a) \sim d^\pi} \left[R(s, a)\right], \quad s_0 \sim \mu_0 \end{aligned} \quad (2)$$

where Q^π stands for the state-action values and d^π represents the visitations of π which satisfy the following:

$$Q^\pi(s, a) = R(s, a) + \gamma \cdot P^\pi Q^\pi(s, a) \quad (3)$$

$$d^\pi(s, a) = (1-\gamma)\mu_0(s)\pi(a|s) + \gamma \cdot P^*_\pi d^\pi(s, a) \quad (4)$$

As the two linear operators, P^π stands for the policy transition operator whereas P^*_π stands for its transpose which satisfy the following:

$$P^{\pi}Q(s,a) := \mathbb{E}_{s' \sim T(s,a), a' \sim \pi(s')}[Q(s',a')] \tag{5}$$

$$P^{*}_{\pi}d(s,a) := \pi(a|s) \sum_{\substack{s' \sim T(s,a) \\ a' \sim \pi(s')}} T(s|s',a')d(s',a') \tag{6}$$

The function Q^{π} represents the Q-values for policy π, mapping state-action pairs (s,a) to their expected value under π. Additionally, the function d^{π} is the normalized state-action visitation distribution, indicating the probability of encountering (s,a) under π, averaged over time with γ-discounting.

2.2 DICE Estimators for Behaviour-agnostic Off-policy Evaluation

Off-policy evaluation (OPE) is a crucial technique in RL that aims to estimate the value of a policy $\rho(\pi)$ by using only a fixed dataset of experiences, without further interactions with the environment. This aspect is particularly important in settings where interacting with the real environment is costly, risky or even infeasible [21]. Therefore, OPE makes it much more practical to improve policies in a controlled, off-line manner before any real-world deployment, thereby mitigating possible risks and reducing high costs.

In this context, we assume we have access to a finite dataset $D = \{(s_0^{(i)}, s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)})\}_{i=1}^{N}$, where $s_0^{(i)} \sim \mu_0$, $(s^{(i)}, a^{(i)}) \sim d_D$ are samples from some unknown distribution d_D (such as the state-action visitation distribution $d^{\pi}(s,a)$ for some unknown policy π). Estimators using DICE methods employ the following expression to derive an estimate average per-step reward value of the target policy:

$$\rho(\pi) = \mathbb{E}_{(s,a,r)\sim d_D}[\hat{\zeta}(s,a) \cdot r] \text{ where } \hat{\zeta}(s,a) = \frac{d^{\pi}(s,a)}{d_D(s,a)} \tag{7}$$

where $(s,a,r) \sim d_D$ is used as an abbreviated form of $(s,a) \sim d_D, r = R(s,a), s' \sim T(s,a)$. In other words, this simulates sampling from the dataset D when using a finite number of samples. Most importantly, $\hat{\zeta}(s,a)$ stands for the distribution correction ratio. The main objective of the DICE estimators is that they aim to approximate the value of this correction ratio without requiring the knowledge of d^{π} or d_D. This is where the concept of behaviour-agnostic arises from. Note that the estimate calculation in Equation 7 is used as the standard DICE paradigm which corresponds to the dual objective.

This estimation is achieved by solving a linear programming (LP) problem under the constraints imposed by the d^{π} distribution. Specifically, the DICE framework leverages the duality between the Q^{π}-function and the d^{π}-distribution, as outlined in Yang et al.'s work [23]. The estimation process involves expressing the objective as a min-max optimization problem, where the Lagrangian formulation is used to handle the inherent complexities of off-policy evaluation. To make this optimization feasible, particularly when dealing with large or continuous state and action spaces, the Lagrangian is augmented with regularization techniques which is known as the augmented Lagrangian method (ALM) [18]. These regularizations are crucial as they introduce strong convexity into the optimization problem, which stabilizes the solution and helps mitigate

issues like bias in gradient estimation. This approach allows the DICE estimators to approximate the correction ratio $\hat{\zeta}(s, a)$ without requiring explicit knowledge of the d^π or d_D distributions.

2.3 Double-dipping and Sample Splitting with k-fold Cross-fitting

Double-dipping in machine learning refers to overfitting by using the same dataset for both model training and evaluation, leading to poor generalizability [2]. In the context of DICE estimators, this issue arises as the same dataset is used for training and estimating the target policy. In order to introduce more stability into the optimization, DICE employs mechanisms to apply regularization techniques and redundant constraints [23]. Regularization works by adding a penalty term to the loss function used to train the model which increases the loss for larger model coefficients, thereby discouraging the model from fitting too closely to the training data [5]. While regularization techniques help mitigate overfitting, they can increase bias by preventing the model from capturing true patterns.

Therefore, this study explores whether sample-splitting, a fundamental technique from double/de-biased machine learning, can reduce variance while maintaining or reducing bias. This is experimented with sample-splitting and cross-fitting methods being layered on top of the regularization techniques mentioned in the previous section to further improve the reliability and robustness of the estimation process. These methods ensure that the model is not overly dependent on any single partition of the data, leading to more accurate and generalizable estimates. More specifically, sample-splitting involves dividing the dataset into K folds, using one part for training and the other for estimation, then switching roles to utilize the full dataset, a process known as cross-fitting. This approach aims to restore efficiency lost due to data partitioning by averaging estimates across all folds [8]. For further explanation on sample-splitting and cross-fitting applied to DICE estimators, please see section 4.

3 Related Work

The study that sustains the backbone of this work is that of Yang et al. on off-policy evaluation via the regularized Lagrangian [23]. By showing that the previous DICE formulations are all equivalent to regularized Lagrangians of the same linear program (LP), they specifically investigate the dual form, namely $d{-}LP$, for off-policy evaluation. They then identify a list of choices with regards to translating this formulation into a stable minimax optimization problem. These choices consist of the specification of redundant constrains and the regularization of primal and dual variables. This optimization unifies all the variants of the DICE estimators under one framework by selecting an appropriate regularization configuration. These are listed as DualDICE [12], GenDICE [25], GradientDICE [26], DR-MWQL and MWL [22], LSTDQ [10], Algae $Q{-}LP$ [13] and BestDICE with it being the variant that achieves the best performance out of all. This said, the estimator introduced by our paper builds upon BestDICE as well.

There has also been studies in developing doubly-robust estimators inspired from the methods used in machine learning. In a recent study of Kallus and Uehara, with the objective of achieving true off-policy evaluation in time-invariant Markov processes, they develop a new estimator based on double reinforcement learning [9]. They design an estimator that employs both estimated stationary density ratios and q-functions. This design maintains efficiency even when both components are estimated slowly and ensures consistency when either component is estimated accurately. The strategy used for the double reinforcement learning also sets the initiative behind this study. In their comparative analysis of the estimator architecture, one of the estimators used for comparison is DualDICE which is a part of the DICE family as mentioned before. This variant uses dual regularization parameters similar to BestDICE which makes their work especially valuable to give close attention to the nature of dual regularized DICE estimators. However, their study lacks a comparative analysis of different k-fold cross-fitting proportions, which our paper aims to address.

In Chernozhukov et al.'s study for double machine learning for treatment and causal parameters, they dive into the details of Double ML and the methodology behind it in order to improve the performance of naive ML estimators [6]. Their objective differs from previously mentioned studies since it does not encompass DICE estimators or off-policy evaluation in its scope. By setting up a background for why and how modern supervised statistical/machine learning fails to provide accurate estimators of causal parameters, they provide reasoning behind the poor performance of naive estimators one of which being regularization bias. They come to point out that while regularization helps with stability and convergence, it introduces a bias into the estimator. This is especially relevant to the current study since DICE estimators use techniques of primal and/or dual regularization in order to avoid overfitting. It is essential to note that the main objective of this study is to demonstrate the orthogonalization of double machine learning, driven by the need to counteract bias introduced by non-orthogonal ML estimators. However, our study is specifically motivated by addressing the phenomenon of double-dipping, which naturally focuses on reducing variance. Since within the DICE family, some estimators such as BestDICE have already been tailored to counteract the negative effects of certain regularization choices on bias, our study prioritizes variance as a more valuable performance metric than bias.

Jacob's study on cross-fitting and averaging in machine learning for estimating heterogeneous treatment effects explores the performance of twelve different estimators across four meta-learners [8]. The study aims to uncover correlations between the learning processes of these meta-learners and the specific procedures used in doubly robust machine learning techniques, such as various types of sample-splitting, cross-fitting, and averaging. While Jacob's work is confined to machine learning and does not concern off-policy evaluation, it offers valuable methodological insights for our work, particularly in establishing a robust comparative analysis. Although our study does not employ the same breadth of techniques as Jacob's, we adopt similar cross-fitting proportions, specifically 2-fold and 5-fold cross-fitting, to maintain consistency in our experimental approach. However, it is important to recognize that the results in Jacob's study are more statistically significant than ours, largely due to the fact that the regularization strategies within the DICE framework can potentially reduce the efficiency of

double machine learning techniques. This highlights the unique challenges of integrating behavior-agnostic off-policy evaluation with double machine learning, which differ from those addressed in Jacob's study.

4 Methodology

In the context of DICE estimators, as discussed in section 2, training the estimator involves focusing on the Q-value functions and visitation densities. The parameter of interest, in this case, is the estimated value of the target policy, $\rho(\pi)$.

As suggested by Chernozhukov et al., we assume that a random sample $(W_i)_N^{i=1}$ with W_i is an individual observation drawn from the distribution of W is available for evaluation and training [6]. For two fittings, we build two prediction models based on the roles of the samples, these samples being I and I^c. The true value ν_0 of the nuisance parameter ν is estimated by $\hat{\nu}_0(I^c)$ using the training sample $(W_i)_{i\in I^c}$. The true value θ_0 of the target parameter θ is estimated by the estimator $\check{\theta}_0(I, I^c)$ using the evaluation sample $(W_i)_{i\in I}$. Here, the nuisance parameter refers to the distribution correction ratio and the target parameter refers to the average per-step reward value, as mentioned before in Equation 7. In the aggregation of the results, the calculation of the joint estimate value is given by:

$$\tilde{\theta}_0 = \frac{\check{\theta}_0(I, I^c)}{2} + \frac{\check{\theta}_0(I^c, I)}{2} \tag{8}$$

where I and I^c represent a random 50-50 split of the dataset. Over a batched number of episodes $\{1, \ldots, N\}$, the size of I is n, the size of I^c is also n, and the total sample size is $N = 2n$. We then construct an estimator $\check{\theta}_0(I, I^c)$ that employs the nuisance parameter estimator $\hat{\nu}_0(I^c)$ where I is used for evaluation dataset and I^c is used for training dataset. This can be interpreted as building the prediction model (neural network) of the estimator on the training dataset, I^c. Then, we reverse the roles of I and I^c and construct an estimator $\check{\theta}_0(I^c, I)$ that employs the nuisance parameter estimator $\hat{\nu}_0(I)$ where I^c is used for evaluation dataset and I is used for training dataset. We are now building the prediction model (neural network) of the estimator on the dataset I that was used as evaluation dataset in the first fitting. The results of the two estimators are then aggregated into a final estimate value, $\tilde{\theta}_0$ by taking the average.

For a fold number that is larger than 2, the process involves a K-fold random split of the entire sample with $k = 1, \ldots, K$. Then, for each fold we construct an estimator $\check{\theta}_0(I_k, I_k^c)$ that employs the nuisance parameter estimator $\hat{\nu}_0(I_k^c)$ where I_k is used for evaluation dataset and I_k^c is used for training dataset. Note that each training set $I_k^c = \bigcup_{m\neq k} I_m$ has size $N \cdot \left(\frac{K-1}{K}\right)$ and each evaluation set I_k has size $\frac{N}{K}$ and the total sample size is N. The calculation for the joint estimate value is then given by:

$$\tilde{\theta}_0 = \frac{1}{K}\sum_{k=1}^{K} \check{\theta}_0(I_k, I_k^c) \tag{9}$$

As an example, we present the diagram in Figure 1 for a visual representation of the splitting process that applies 5-fold cross-fitting. Each box colored with blue represents

the evaluation dataset of the corresponding iteration whereas the grey colored boxes are merged altogether to represent the training dataset.

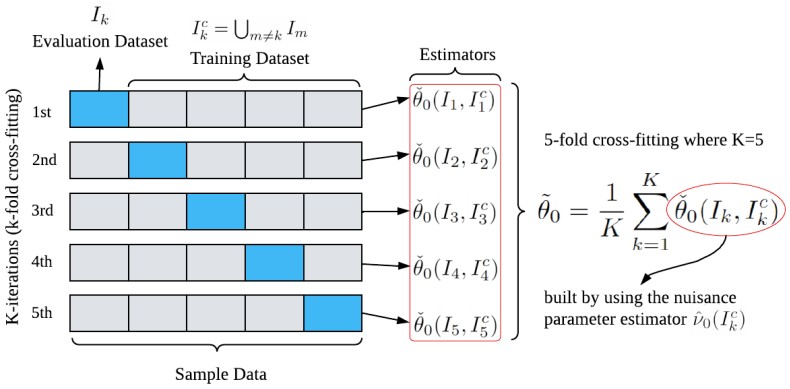


Fig. 1: **Model representation for a 5-fold cross-fit estimator**. The sample data is split into 5 folds. During each iteration of cross-fitting, one fold, I_k, serves as the evaluation dataset, while the remaining folds are merged into a training dataset, I_k^c. In each iteration, a new estimator is built using the nuisance parameter estimator $\hat{\nu}_0(I_k^c)$. The final estimate value $\tilde{\theta}_0$ is obtained after running all the iterations and averaging out the results retrieved from each estimator.

Following the mathematical concepts provided before, we introduce SplitDICE, which implements k-fold cross-fitting for DICE estimators. The three algorithms fundamental to the process of k-fold cross-fitting for the DICE estimators are presented in section A. The core concept that comes preliminary to cross-fitting which is sample-splitting is demonstrated in Algorithm 1, where the dataset is split into training and evaluation sets based on a specified ratio, while preserving the episode structure of the original data. Iterations continue until each subset is used for estimation, with remaining subsets forming the training set. All possible train-eval pairs are generated for cross-fitting. Algorithm 2 and Algorithm 3 detail the training-estimation process and the full run of k-fold cross-fitting respectively, contributing to the calculation of the final joint estimate after running all the K folds.

5 Experimental Setup

For the purposes of this research, Google's DICE-RL codebase was forked for own use [24], it is made publicly available on the GitHub platform [1]. The datasets used for the experiment were generated using OpenAI Gym, which is an open-source Python library and a standard API for reinforcement learning. It provides simulated training environments to both train and test reinforcement learning agents. The environment simulated to create datasets for the experiment is `Frozenlake-v0`, which features discrete action and observation spaces. In the FrozenLake environment, the agent navigates a grid world represented as a frozen lake, encountering holes and frozen surfaces (which result in failure with a reward of 0) and a goal (which results in success with a reward of

+1). This process can be reproduced by creating a behaviour and target dataset where $\alpha = 0.0$ refers to the dataset for the behaviour policy and $\alpha = 1.0$ refers to the dataset for the target policy. The aim is to estimate the average per-step reward value of the target policy (also referred as cumulative normalized expected reward) using the behaviour policy. In addition to the environment specifications, as given in the default experimental setup of Nachum et al., 20 datasets are created with 20 unique seed numbers (from 0 to 19, inclusive) for generalization purposes [12]. The number of trajectories and the maximum length of each trajectory are set to 200 and 100 respectively for all replications.

For all the datasets separately, the estimator is trained for 10,000 steps and the value of the target policy is estimated at intervals of 100 training steps. Initially set as default at 10,0000 training steps with estimations made every 500 steps, this configuration was adjusted due to the Frozenlake environment's faster convergence, necessitating fewer training steps for accurate estimator performance.

For the experiment at hand, we use DICE with no cross-fitting (in other words, Naive DICE), SplitDICE with 2-fold and 5-fold cross-fitting. For all the estimator types, the estimator configurations are kept the same meaning we use the default choice of parameters as provided in the codebase. More specifically, from the unified framework of DICE estimators, the configurations used for the experiment are those belonging to BestDICE due to its high performance and best unbiased estimate achievement as concluded by Yang et al. [23]. More specifically, these configurations are listed as $\alpha_Q = 0, \alpha_\zeta = 1, \alpha_R = 0/1$ with $\zeta \geq 0$ and λ. Here, α_Q stands for the primal regularizer and α_ζ for the dual regularizer, they both influence how much regularization is applied to the primal and dual problem in optimization, respectively. Additionally, α_R stands for inclusion (1) / exclusion (0) of reward, ζ and λ stand for the positivity and normalization constraints.

6 Results

In this section, we present and analyze the experimental results in relation to the objectives of the study. The analysis is divided into three subsections: section 6.1 covers relative error calculations and their statistical significance, section 6.2 examines other performance metrics across all 20 seeds, and section 6.3 discusses convergence to the ground truth, with a focus on variance as the key metric.

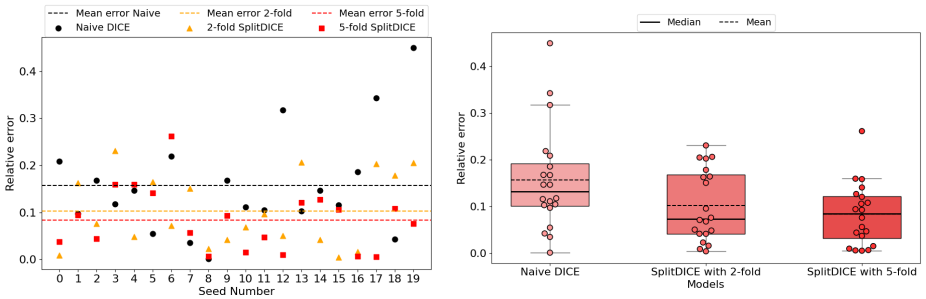
6.1 Error Calculation and Comparison

To evaluate the estimator's error, we use the absolute relative error as a metric. This choice is motivated by the fact that relative error normalizes the absolute error based on the true value, resulting in a dimensionless measure. Such normalization is particularly useful for comparing errors across different scales, which is relevant for this experiment. The datasets, generated with different seeds, may have varying ground truth values for the target policy's average per-step reward. Therefore, the significance of an error lies not just in its magnitude, but in its proportion relative to the true value.

The results shown via Figure 2a demonstrate the relative error for each scenario at a given seed number starting from 0 to 19 (inclusive). The comparisons are made between the three values plotted per seed since datasets with different seeds have different

ground truths. The error is calculated by comparing the value of ground truth (retrieved from the target policy) against the final estimated value for average per-step reward (retrieved from the trained behaviour policy).

For a clear comparison of distribution of points, Figure 2b provides a categorical whisker diagram for these error values. It indicates that DICE estimators used with 2-fold and 5-fold cross-fitting exhibit lower rates of error mostly in the spread of the central portion of the data. As seen from shorter whisker lengths, the variance in error values exhibits notable reduction with SplitDICE, particularly when applied with a 5-fold cross-fitting. Although 2-fold SplitDICE is the only model without any outliers, the error values are concentrated towards the edges of the box on the 1st and 3rd quartiles, indicating a skewness in the distribution. Another important finding is that the median line overlaps the mean for 5-fold cross-fitting, this implies that the data is not skewed heavily in one direction and that there are no significant outliers pulling the mean away from the median. This is noteworthy to mention since it confirms that higher-fold cross-fitting provides better estimates due to more thorough validation, demonstrating its validity [6].



(a) The results are categorized by the considered estimator models, with the mean relative error displayed for each as a general summary of all the data points.

(b) The results are categorized by the considered estimator models, with the (non-outlier) data points displayed as a randomized swarm to avoid overlaps.

Fig. 2: **Scatter (a) and box-whisker (b) plots showing the relative error between the final estimated average per-step reward value and the ground truth.** The results are obtained for all the 20 seeds from 0 to 19, inclusive.

Before moving on with any statistical analysis on significance, we conducted Anderson -Darling and Shapiro-Wilk tests to determine whether the data is normally distributed and thereby decide whether to continue with parametric or non-parametric approaches. Since the results of the test determined at least one of the datasets to be not normally-distributed, non-parametric approaches were considered and therefore Kruskal-Wallis test was determined suitable. This analysis aim to compare different estimators' performance and determine whether there exists statistically significant disparities in the results of the relative error values. The results for Kruskal-Wallis test-statistic and p-value are reported as 5.73377 and 0.05688 respectively, with the alpha value set to 0.1 (i.e. significance level of 10%). A higher test-statistic suggests a greater likelihood of significant differences among the groups. The obtained p-value indicates

that there is a 5.688% chance of reaching the observed test-statistic value if the null hypothesis were true (i.e. there are no significant differences between groups). However, given that the p-value is less than the predetermined significance level, the null hypothesis is rejected. Therefore, we infer that at least one of the estimator categories exhibits a mean that is significantly different from the others.

However, since Kruskal-Wallis is an extension of Mann-Whitney U test for three or more categorical and independent groups, we also conducted Mann-Whitney U test for post-hoc pairwise comparisons to identify which group(s) contributes to the significance. The results of this test are reported via Table 1. From all the pairwise comparisons, the only significant result comes from Naive versus 5-fold. With the adjusted p-value surpassing the significance level and a considerably high U-value, the difference in the median values of the two estimator models in terms of rank sums shows a significant disparity.

Table 1: Mann-Whitney U Test Results for Relative Error Comparison

Group 1	Group 2	Test-statistic (U-value)	p-value	Adj. p-value after Holm correction	Reject
2-fold SplitDICE	5-fold SplitDICE	230.0	0.42488	0.42488	False
2-fold SplitDICE	Naive DICE	258.0	0.11986	0.23971	False
5-fold SplitDICE	Naive DICE	286.0	0.02073	0.06220	True

6.2 Evaluation of Performance Metrics

Additionally, other metrics such as mean-squared error, bias and variance have been calculated from the average per-step reward values of 2-fold and 5-fold cross-fit estimators,as well as a naive single estimator, after training all for 10,000 steps. The formulas for these metrics are detailed in section B. The resulting values have been provided in Table 2.

We can say that the results advocate for the advantages of using cross-fitting methods over the naive estimator. The naive estimator has the highest MSE, marking less accuracy in prediction. On the contrary, 2-fold cross-fitting significantly reduces the MSE value reflecting a nearly 55% improvement. As we move on to 5-fold cross-fitting, we see even a further reduction in the MSE performance with approximately a 70% improvement over the naive estimator and a 25% improvement over 2-fold. Additionally, bias is substantially lower with cross-fitting methods compared to the naive estimator, although surprisingly enough 5-fold demonstrates a slight increase in bias compared to the 2-fold approach. As expected so, variance also show improvements with cross-fitting; again with 5-fold cross-fitting achieves the lowest in both, establishing more consistent and precise results of estimation.

Table 2: Performance measures for the considered estimators

Scenarios	MSE	Bias	Variance
Naive estimator	7.773960e-06	0.000309	0.035533
2-fold cross-fit	3.424816e-06	0.000144	0.015937
5-fold cross-fit	2.487964e-06	0.000172	0.011163

6.3 Convergence of Average Per-Step Reward over Training

Plots provided by Figure 4 illustrate the average per-step reward over 10,000 training steps, with results generalized using the median and error bars at 25^{th} and 75^{th} percentiles across 20 datasets with distinct seeds. The average reward of the target policy serves as the ground truth, and results are aggregated by averaging over all seeds.As it can be seen clearly, all three estimators show convergence to the true value of the target policy. The convergence has a more fluid movement for SplitDICE however whereas for the naive implementation, there seems to be more extremities and peaks throughout the training process. The deviation in the convergence trend of Naive DICE shown via Figure 4a is smoothed out when applied cross-fitting. Another noticeable remark is that both folds of SplitDICE start with a relatively high peak at the start of the training process. Although variance is generally high, this is not the case for Naive DICE. This could be attributed to sampling variability meaning the training set created as an outcome of random split of the original dataset may not generalize well to the overall dataset thereby leading to estimations that are far from the ground truth until the estimator adjusts to the true distribution during the later stages of training. This is also an apparent occurrence between the two different folds since 2-fold SplitDICE shown via Figure 4b reaches a higher initial peak (between 0.05 and 0.06) compared to the 5-fold SplitDICE shown via Figure 4c (between 0.04 and 0.05). This would also indicate that by splitting of the data into higher number of folds, the training set is more likely to be a better representative of the original dataset's variability.

In order to truly understand the effects of the double-dipping behaviour, we analyze the variance in the average per-step reward. We hypothesize that SplitDICE will exhibit lower variance compared to Naive DICE, indicating that the double machine learning strategies employed in SplitDICE provides better stability. As shown in Figure 3, SplitDICE, particularly in the 5-fold configuration, demonstrates a tighter concentration of data points around the median, whereas Naive DICE shows greater variability and less consistency. This observation indicates not only that there is a descending trend in variance from Naive to 5-fold SplitDICE but also that SplitDICE (more significantly for the 5-fold version) densely clusters data points around the desired range achieving a more stable and focused distribution. Additionally, for SplitDICE the gap between the mean and the median is much smaller suggesting that distribution of reward values is more symmetric and centered around the true value.

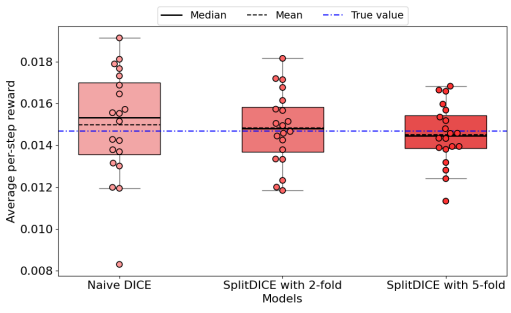
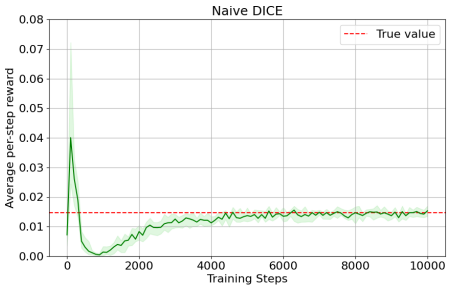
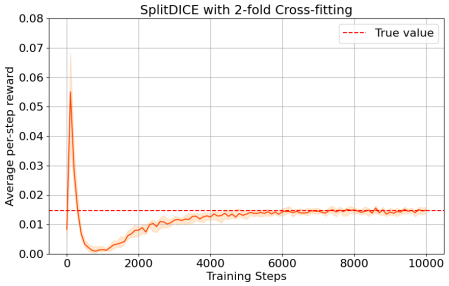


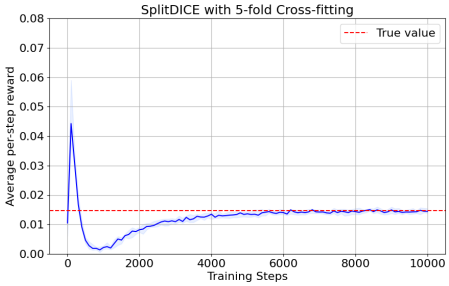
Fig. 3: **Box-whisker plot showing the estimated average per-step reward value (calculated at step=10,000) for each seed.** The results are categorized by the considered estimator models, with the (non-outlier) data points displayed as a randomized swarm to avoid overlaps. Additionally, the mean ground truth value is displayed as a reference criterion for comparison.



(a) The sample data is used both as a training and an evaluation dataset.



(b) The sample data is randomly divided into training and evaluation subsets. Each subset is used for fitting in turn, and the final estimate is the average of both fittings.



(c) The sample data is randomly divided into five subsets. Each subset is used once for estimation while the remaining four are used for training. The final estimate is the average of the five fittings.

Fig. 4: **Convergence to the true value (average per-step reward of the target policy) over the entire training process of Naive DICE (a), 2-fold SplitDICE (b) and 5-fold SplitDICE (c)**. The estimation of the average per-step reward value is done at every 100-step intervals of the entire training process, which totals 10,000 steps overall.

7 Discussion

The analysis yielded several insights. While variance is crucial for assessing double-dipping risks, examining how the cross-fit strategy impacts bias was also valuable due to the trade-off between them. Although the differences in bias were smaller than in variance, there was a clear downward trend from Naive DICE to SplitDICE. The whisker plots in Figure 2b and Figure 3 revealed that while the estimators' error values had similar interquartile range widths, their overall value distributions differed. In contrast, for the average per-step reward values, the patterns differed: the estimates for each model were similarly close to the true value, but their interquartile ranges varied much more significantly. In summary, this proves that variance in error was (nearly) unaffected changing to SplitDICE but the variance in the average per-step reward values decreased noticeably.

Additionally, before conducting the statistical analysis for relative error comparison, we first assessed the normality of the datasets, choosing between the Shapiro-Wilk [19] and Kolmogorov-Smirnov (K-S) [4] tests. The K-S test has limitations, including high

sensitivity to extreme values and invalid critical regions when parameters are estimated, making it less suitable for this experiment [15]. Given that the Shapiro-Wilk test is more appropriate for smaller sample sizes [11], we used both the Anderson-Darling (a refinement of the K-S) [20] and Shapiro-Wilk tests. The Anderson-Darling test showed non-normality for both 2-fold and Naive datasets, while the Shapiro-Wilk indicated only 2-fold was non-normal (with Naive's p-value close to significance). These results led us to use non-parametric methods for further analysis, specifically the Kruskal-Wallis test. To account for potential false negatives, we also considered parametric methods under the assumption of normality, using one-way ANOVA with Tukey's HSD for post-hoc analysis [14]. The statistical analysis showed a significant difference between the Naive estimator and 5-fold SplitDICE at a 5% significance level, compared to 10% with non-parametric methods.

We now discuss the study's limitations, starting with the choice of environment: `Frozenlake-v0`, which has a discrete action and observation space. This environment was chosen for its simplicity and manageable computational demands. More complex environments potentially offering richer datasets with discrete action space and continuous observation space, such as `Reacher-v2` and `Cartpole-v0`, were excluded due to their higher computational requirements, as they need more training steps to converge to the true value, as seen in previous DICE studies [23]. This choice ensured the experiments could be completed within a reasonable time frame and with available resources. Furthermore, the dataset was split by shuffling episodes from the off-policy dataset to achieve a fully random split, reducing bias from the original data order. To address the fluctuations due to shuffling variability, results were averaged across multiple seeds, as suggested in prior DualDICE studies [12]. Although the episodes were shuffled, the seed values for random number generation were systematically set from 0 to 19 for consistency and reproducibility. Using randomized seeds could further enhance reliability by minimizing the influence of randomness.

8 Conclusion

This study investigated how incorporating sample-splitting and cross-fitting techniques can mitigate the issue of "double-dipping" in behavior-agnostic reinforcement learning. These techniques were integrated into the DICE estimators, resulting in a new approach called SplitDICE. The study involved partitioning the data into subsets and employing cross-fitting through multiple training iterations to ensure each subset was used to estimate the average per-step reward of the target policy. Using the modified DICE-RL codebase from Google Research, this study compared three models: Naive DICE (BestDICE), 2-fold SplitDICE, and 5-fold SplitDICE. Results revealed that 5-fold SplitDICE achieved lower relative error rates compared to Naive DICE at 10% significance, and exhibited a consistent reduction in MSE, bias, and variance across 20 seeds. Additionally, the variance, calculated after 10,000 steps, decreased progressively from Naive DICE to 2-fold SplitDICE and further to 5-fold SplitDICE, reflecting a pattern of increasing stability in convergence. Future work could involve testing these methods in more complex environments with larger trajectories and applying sample-splitting and cross-fitting to other DICE variants such as DualDICE and GradientDICE to explore their potential for scalability.

Acknowledgments. S. Bongers and F. A. Oliehoek are supported by the Mercury Machine Learning Lab, a collaboration between TU Delft, UvA, and booking.com.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aslan, Y.: Use of sample-splitting and cross-fitting techniques to mitigate the risks of double dipping in behaviour-agnostic reinforcement learning. https://github.com/compScienceYaren/dice_rl (2024)
2. Ball, T.M., Squeglia, L.M., Tapert, S.F., Paulus, M.P.: Double dipping in machine learning: Problems and solutions. Biological Psychiatry: Cognitive Neuroscience and Neuroimaging **5**(3), 261–263 (Mar 2020). https://doi.org/10.1016/j.bpsc.2019.09.003
3. Cen, Z., Liu, Z., Wang, Z., Yao, Y., Lam, H., Zhao, D.: Learning from sparse offline datasets via conservative density estimation (arXiv:2401.08819) (Mar 2024). https://doi.org/10.48550/arXiv.2401.08819, http://arxiv.org/abs/2401.08819, arXiv:2401.08819 [cs]
4. Chakravarti, I.M., Laha, R.G., Roy, J.: Handbook of Methods of Applied Statistics, Volume I. John Wiley and Sons, Hoboken (1967)
5. Cheng, R., Verma, A., Orosz, G., Chaudhuri, S., Yue, Y., Burdick, J.W.: Control regularization for reduced variance reinforcement learning (arXiv:1905.05380) (May 2019). https://doi.org/10.48550/arXiv.1905.05380, http://arxiv.org/abs/1905.05380, arXiv:1905.05380 [cs, stat]
6. Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W.: Double machine learning for treatment and causal parameters (Jul 2016)
7. Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., Robins, J.: Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal **21**(1), C1–C68 (2018). https://doi.org/10.1111/ectj.12097
8. Jacob, D.: Cross-fitting and averaging for machine learning estimation of heterogeneous treatment effects (arXiv:2007.02852) (Aug 2020). https://doi.org/10.48550/arXiv.2007.02852, http://arxiv.org/abs/2007.02852, arXiv:2007.02852 [stat]
9. Kallus, N., Uehara, M.: Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning. Operations Research **70** (Feb 2022). https://doi.org/10.1287/opre.2021.2249
10. Lagoudakis, M., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research **4**, 1107–1149 (Jan 2003). https://doi.org/10.1162/1532443041827907
11. Mishra, P., Pandey, C.M., Singh, U., Gupta, A., Sahu, C., Keshri, A.: Descriptive statistics and normality tests for statistical data. Annals of Cardiac Anaesthesia **22**(1), 67–72 (2019). https://doi.org/10.4103/aca.ACA_157_18
12. Nachum, O., Chow, Y., Dai, B., Li, L.: Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections (arXiv:1906.04733) (Nov 2019). https://doi.org/10.48550/arXiv.1906.04733, http://arxiv.org/abs/1906.04733, arXiv:1906.04733 [cs, stat]
13. Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., Schuurmans, D.: Algaedice: Policy gradient from arbitrary experience (arXiv:1912.02074) (Dec 2019). https://doi.org/10.48550/arXiv.1912.02074, http://arxiv.org/abs/1912.02074, arXiv:1912.02074 [cs]
14. Nanda, A., Mahapatra, A., Mohapatra, B., mahapatra, a.: Multiple comparison test by tukey's honestly significant difference (hsd): Do the confident level control type i error. International Journal of Applied Mathematics and Statistics **6**, 59–65 (Jan 2021). https://doi.org/10.22271/maths.2021.v6.i1a.636

15. Peng, G.: Po 04 testing normality of data using sas ® (2004), https://www-semanticscholar-org.tudelft.idm.oclc.org/paper/PO-04-Testing-Normality-of-Data-using-SAS-%C2%AE-Peng/38c74f7d7f480df462c986905d3c1a941bc26dc8
16. Precup, D., Sutton, R., Singh, S.: Eligibility traces for off-policy policy evaluation. Computer Science Department Faculty Publication Series pp. 759–766 (Jun 2000)
17. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., Hoboken, NJ, USA (1994). https://doi.org/10.1002/9780470316887, https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316887, first published: 15 April 1994
18. Rockafellar, R.T.: Augmented lagrange multiplier functions and duality in nonconvex programming. SIAM Journal on Control **12**(2), 268–285 (May 1974). https://doi.org/10.1137/0312021
19. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**(3/4), 591–611 (1965). https://doi.org/10.2307/2333709
20. Stephens, M.A.: Edf statistics for goodness of fit and some comparisons. Journal of the American Statistical Association **69**(347), 730–737 (1974). https://doi.org/10.2307/2286009
21. Thomas, P., Theocharous, G., Ghavamzadeh, M.: High-confidence off-policy evaluation. Proceedings of the AAAI Conference on Artificial Intelligence **29**(11) (Feb 2015). https://doi.org/10.1609/aaai.v29i1.9541, https://ojs.aaai.org/index.php/AAAI/article/view/9541
22. Uehara, M., Huang, J., Jiang, N.: Minimax weight and q-function learning for off-policy evaluation (arXiv:1910.12809) (Oct 2020). https://doi.org/10.48550/arXiv.1910.12809, http://arxiv.org/abs/1910.12809, arXiv:1910.12809 [cs, stat]
23. Yang, M., Nachum, O., Dai, B., Li, L., Schuurmans, D.: Off-Policy Evaluation via the Regularized Lagrangian (Jul 2020)
24. Yang, S., Chen, R., Yang, Y., Hawkins, P., Brevdo, E., Qianli, S.Z.: Dice: The distribution correction estimation library. https://github.com/google-research/dice_rl (2023)
25. Zhang, R., Dai, B., Li, L., Schuurmans, D.: Gendice: Generalized offline estimation of stationary values (arXiv:2002.09072) (Feb 2020). https://doi.org/10.48550/arXiv.2002.09072, http://arxiv.org/abs/2002.09072, arXiv:2002.09072 [cs, stat]
26. Zhang, S., Liu, B., Whiteson, S.: Gradientdice: Rethinking generalized offline estimation of stationary values (arXiv:2001.11113) (Nov 2020). https://doi.org/10.48550/arXiv.2001.11113, http://arxiv.org/abs/2001.11113, arXiv:2001.11113 [cs, stat]

A Algorihms

Algorithm 1 Sample-splitting of the dataset with fold number K

procedure RANDOM_SPLIT(dataset $\mathcal{D}$, training ratio α)
  Retrieve total number of samples N from $\mathcal{D}$
  Calculate number of evaluation samples $n_1 \leftarrow \alpha \times N$
  Calculate number of training samples $n_2 \leftarrow N - n_1$
  Retrieve all episodes from the dataset as a list
  Shuffle the episodes at random ▷ to account for a fully random split
  Initialize an empty list, $\mathcal{F}$ ▷ to store pairs of train-eval datasets
  Calculate the size of each fold (subset) $f \leftarrow \frac{N}{K}$
  for $k \leftarrow 0$ to $K-1$ **do**
    Determine start index $s \leftarrow k \times f$
    Determine end index $e \leftarrow (k+1) \times f$ **if** $k < K-1$ **else** N
    Initialize a new off-policy dataset $\mathcal{D}_{eval}$ with capacity n_1 ▷ evaluation sample, I_k
    Initialize a new off-policy dataset $\mathcal{D}_{train}$ with capacity n_2 ▷ training sample, I_k^c
    Add episodes from $[s:e]$ to $\mathcal{D}_{eval}$
    Add remaining episodes to $\mathcal{D}_{train}$
    Append the pair $(\mathcal{D}_{eval}, \mathcal{D}_{train})$ to $\mathcal{F}$
  end for
  return $\mathcal{F}$
end procedure

Algorithm 2 Run training and estimation per each fold with fold number K

procedure RUN_TRAINING_ESTIMATION(estimator $\check{\theta}_0$, training dataset $\mathcal{D}_{train}$, evaluation dataset $\mathcal{D}_{eval}$, joint estimates $\mathcal{J}$, fold index k_{index})
  Retrieve the target dataset, $\mathcal{D}_{target}$
  Initialize an empty list, $\mathcal{R}$ ▷ to store estimate values received over training
  for each step from 0 to 10,000 **do**
    Retrieve a batch of transitions $\mathcal{T}$ from $\mathcal{D}_{train}$
    Retrieve a batch of initial steps $\mathcal{S}$ from $\mathcal{D}_{train}$ and preprocess initial steps batch
    Perform a training step for $\check{\theta}_0$ using $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{D}_{target}$
    if step is a multiple of 100 or step is the last step **then**
      Estimate average per-step reward r using $\mathcal{D}_{eval}$ and $\mathcal{D}_{target}$ ▷ via Equation 7
      Append estimate r to $\mathcal{R}$
      if step is the last step **then**
        Update $\mathcal{J}$ by calling CALCULATE_JOINT_ESTIMATE($\mathcal{R}$, $\mathcal{J}$, k_{index})
      end if
    end if
  end for
  return $\mathcal{R}$
end procedure

procedure CALCULATE_JOINT_ESTIMATE(running estimates $\mathcal{R}$, joint estimates $\mathcal{J}$, fold index k_{index})
  Append $\mathcal{R}$ to $\mathcal{J}$
  if k_{index} equals K **then**
    Compute the mean $\tilde{\theta}_0$ of all the folds $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$ in $\mathcal{J}$ ▷ via Equation 9
    Log the results for $\tilde{\theta}_0$
  end if
  return $\mathcal{J}$
end procedure

Algorithm 3 k-fold cross-fitting with fold number K

procedure RUN_CROSS_FITTING
    Load the original dataset $\mathcal{D}$ from the directory
    Retrieve a list of all the fold pairs $\mathcal{F}$ by calling RANDOM_SPLIT($\mathcal{D}$, $\frac{1}{k}$)
    Initialize an empty list, $\mathcal{J}$ ▷ to store estimate values received per fold
    for $k \leftarrow 0$ to $K-1$ **do**
        Split into training dataset $\mathcal{D}_{eval}$ and evaluation dataset $\mathcal{D}_{train}$ using $\mathcal{F}[i]$
        Build a DICE estimator $\check{\theta}_0(\mathcal{D}_{eval}, \mathcal{D}_{train})$ ▷ using $\hat{\mu}_0(\mathcal{D}_{train})$
        Update $\mathcal{J}$ by calling RUN_TRAINING_ESTIMATION($\check{\theta}_0$, $\mathcal{D}_{train}$, $\mathcal{D}_{eval}$, $\mathcal{J}$, $k+1$)
    end for
    return $\mathcal{J}$
end procedure

B Performance Metrics

As suggested by Jacob in his research, following equations are used as performance metrics for the study at hand [8]. Keep in mind the formula for variance calculation is adjusted to account for the subtle fluctuations in the ground truth values such that the the outcome is relative to the truth.

$$\text{MSE} = \frac{1}{S}\sum_{s=0}^{S}\left[\tilde{\theta}_0^s - \theta_0^s\right]^2 \tag{10}$$

$$\text{Bias} = \left|\frac{1}{S}\sum_{s=0}^{S}\tilde{\theta}_0^s - \theta_0^s\right| \tag{11}$$

$$\text{Var} = \frac{1}{S}\sum_{s=0}^{S}\left[\frac{\tilde{\theta}_0^s}{\theta_0^s} - \overline{\frac{\tilde{\theta}_0^s}{\theta_0^s}}\right]^2 \tag{12}$$

where θ_0^s refers to ground truth obtained from the target policy and $\tilde{\theta}_0^s$ refers to the estimate value obtained from the model estimators, in the case of cross-fit estimators, this value is the average over K folds. s here is for the seed number since per each seed, the calculated final estimate value and the ground truth might differ.