Neural Computing and Applications (2023) 35:17555–17577
https://doi.org/10.1007/s00521-023-08622-w

ORIGINAL ARTICLE

A domain-region based evaluation of ML performance robustness to covariate shift

Firas Bayram[1] · Bestoun S. Ahmed[1,2]

Received: 21 October 2022 / Accepted: 17 April 2023 / Published online: 12 May 2023

Abstract
Most machine learning methods assume that the input data distribution is the same in the training and testing phases. However, in practice, this stationarity is usually not met and the distribution of inputs differs, leading to unexpected performance of the learned model in deployment. The issue in which the training and test data inputs follow different probability distributions while the input–output relationship remains unchanged is referred to as covariate shift. In this paper, the performance of conventional machine learning models was experimentally evaluated in the presence of covariate shift. Furthermore, a region-based evaluation was performed by decomposing the domain of probability density function of the input data to assess the classifier's performance per domain region. Distributional changes were simulated in a two-dimensional classification problem. Subsequently, a higher four-dimensional experiments were conducted. Based on the experimental analysis, the Random Forests algorithm is the most robust classifier in the two-dimensional case, showing the lowest degradation rate for accuracy and F1-score metrics, with a range between 0.1% and 2.08%. Moreover, the results reveal that in higher-dimensional experiments, the performance of the models is predominantly influenced by the complexity of the classification function, leading to degradation rates exceeding 25% in most cases. It is also concluded that the models exhibit high bias toward the region with high density in the input space domain of the training samples.

Keywords Covariate shift · Concept drift · Robust machine learning · Classifier evaluation · Model degradation

1 Introduction

Robustness to changes is one of the most exemplary properties of a high-quality machine learning (ML) model in this ever-shifting world. Maintaining this property will allow users to work with trustworthy ML systems that perform consistently and efficiently when exposed to the deployment environment. The issue arises from the assumption of most ML models that the data points are independently and identically distributed (i.i.d), which presume that the data are sampled from the same probability distribution. However, this assumption is often not satisfied in real-world applications [1]. This violation would cause the performance of the predictive model to degrade substantially in practice. The main reason for this *model degradation* is that probability theory forms a central foundation for many data mining and machine learning techniques [2], as it provides a framework for quantifying uncertainty [3].

Changes in the underlying probability distributions have been extensively studied in the literature [4, 5]. The situation in which there is an inconsistency between the joint probability distributions $P(x, y)$ of the training and test datasets is called *dataset shift* [6]. Researchers have classified the general dataset shift phenomenon into several types based on the probabilistic source of change. These types are *covariate shift*, *concept drift* and *prior probability shift* [7]. Detailed taxonomy and definitions of change types can be found in our recent overview [8]. Covariate shift takes the form of a change between the marginal

✉ Bestoun S. Ahmed
bestoun@kau.se

Firas Bayram
firas.bayram@kau.se

[1] Department of Mathematics and Computer Science, Karlstad University, Karlstad 65188, Sweden

[2] Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, 12135 Prague, Czech Republic

distributions of the training and test inputs $P_{tr}(x)$ and $P_{te}(x)$, while the conditional distribution $P(y \mid x)$ remains unchanged [9, 10], whereas the concurrent occurrence of changes in both marginal and conditional distributions is called concept drift [11]. Another type of dataset shift occurs when the distribution of classes $P(y)$ changes, known as prior probability shift [12]. The focus of this paper is on analyzing the behavior of ML models in covariate shift situations as it is one of the most common cases in practice [13].

Covariate shift could appear in many real-world domains, such as healthcare systems [14], computer vision [15], NLP [16], and social media [17]. In practice, many sources cause covariate shift situations. The main reason is that the training input is not a representative sample of the entire population [18]. In this case, the training samples do not represent the overall problem due to the biased data collection process toward a specific subpopulation, known as *sample selection bias* [19], for example, bias toward specific demographic groups. Therefore, the model will be implemented on an unseen distribution in deployment, which, as a consequence, would affect the performance of the model.

Analyzing the out-of-sample performance of ML models before deployment is not trivial because of the ubiquitousness of the system's evolution in real-life applications. The classical way is to perform out-of-sample validation using held-out datasets through cross-validation or bootstrap techniques. These techniques assume stationarity in the data distribution of training and test data. Therefore, they cannot fully estimate the model's success rate in the deployment environment where a distributional shift is likely to occur [20]. Another approach to validate the ML models' performance is analyzing the accuracy and bias variability across data sub-populations. Exploring such variability would help diagnose the performance and potentially develop methods to mitigate its effects.

In this paper, the robustness of several common machine learning algorithms on synthetic data is evaluated in the presence of covariate shift in binary classification problems. An overall performance assessment and a region-based evaluation of the model's robustness are performed by decomposing the probability density function (pdf) of the input space domain of the test samples according to the input density distribution ratio between the test and training data samples. To this end, the contributions of this paper are summarized as follows:

1. An evaluation framework is presented to assess the robustness of common ML classifiers in several drift scenarios.
2. Comprehensive comparisons and experiments are conducted to measure model degradation rates under covariate shift in different classification problem settings.
3. Decomposition of the input space based on the ratio of test-to-training input densities is proposed to evaluate the models' performance per domain-region.

For the sake of visualization, exhaustive comparison experiments are conducted on two-dimensional artificial classification problems with covariate shifts formulated by simulating a broad spectrum of distribution drifts before evaluating the model robustness using additional higher-dimensional datasets. The experiments in this paper are designed to address the following primary research questions:

RQ1: What are the main factors that affect the robustness of performance in covariate shift problems and lead to model degradation?

RQ2: Which ML classification algorithms tend to be more robust (vulnerable) to specific problem settings?

RQ3: How does the performance of the classifiers vary in the input space domain when decomposed by regions based on the test-to-training density ratio?

The remainder of this paper is organized as follows. Section 2 gives a general background on the covariate shift problem and provides an overview of the related work. Next, Sect. 3 contains a detailed explanation of the settings specified for the experiments. The methodology for the performance evaluation of the ML models is detailed in Sect. 4. Section 5 presents the results of the different experiments. Threats to validity are discussed in Sect. 6. Finally, Sect. 7 concludes the work and sets out future directions for further extended use of this paper.

2 Background and related work

This section provides an overview of the covariate shift problem and reviews the related work to evaluate the robustness of the ML model's performance under covariate shift.

2.1 Problem formulation and notation

In binary classification problems, the goal is to obtain a prediction function $f : X \longmapsto Y$ that has been trained on training dataset $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^{N}$ of size N drawn i.i.d from a joint distribution $p_{tr}(x, y)$, where $x_i \in \mathbb{R}^d$ is a d-dimensional data instance, or covariates vector $\mathbf{x}$, and $y_i \in \{-1, +1\}$ is the class label. The classifier's performance is evaluated on a test dataset $\mathcal{D}_{te}$ drawn from a joint distribution $p_{te}(x, y)$. The classical method to learn the classifier is through solving the following empirical risk minimization (ERM) problem:

$$\min_{\theta\in\Theta} R(f) = \min_{\theta\in\Theta} \mathbb{E}_{p_{te}}[\ell(f_\theta(x), y)] \approx \min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell(f_\theta(x_i), y_i), \quad (1)$$

where θ denotes the parameter vector, $\mathbb{E}_{p_{te}(x,y)}$ is the expectation over the test distribution $p_{te}(x, y)$, and ℓ is a selected loss function. In stationary distributions, i.e., when $p_{te}(x) = p_{tr}(x)$, ERM provides a *consistent* estimator [21]. While adaptation techniques are required for non-stationary distributions, such as covariate shift situations.

2.2 Covariate shift adaptation

The minimization problem in Eq. 1 works well under the assumption that $p_{te}(x, y)$ is the same as $p_{tr}(x, y)$, which usually does not hold in practice. Thus, an adjustment to ERM introduced in Eq. 1 should be performed for dataset shift situations, that is, when $p_{te}(x, y) \neq p_{tr}(x, y)$. A probabilistic source for the dataset shift is a change in the input data distributions, i.e., when $p_{te}(x) \neq p_{tr}(x)$, which is referred to as *distribution shift* [22], also known as *data drift* [23]. As depicted in Fig. 1, covariate shift is a subset of the generic distribution shift situation when the conditional probability distribution that represents the input–output rule is the same between the training and the test data [24]:

$$p_{tr}(y \mid x) = p_{te}(y \mid x),\ p_{tr}(x) \neq p_{te}(x). \quad (2)$$

One of the most common techniques to address the covariate shift problem is to employ an importance weighting function defined as: $w(x) = \frac{p_{te}(x)}{p_{tr}(x)}$ which estimates the test-to-training density ratio [25]. Subsequently, the risk minimization problem in Eq. 1 is adjusted to the importance-weighted risk:

$$\min_{\theta\in\Theta} R(f) = \min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} w(x_i)\ell(f_\theta(x_i), y_i). \quad (3)$$

There are many popular algorithms to compute the importance weights $w(x_i)$, such as Kernel Mean Matching (KMM) [26], Kullback-Leibler Importance Estimation Procedure (KLIEP) [27] and least squares importance fitting (LSIF) [28]. It was shown that the importance weighting procedure can provide consistent learning under covariate shift, and thus increase the robustness of the performance in changing distributions [29]. However, the main drawback of the procedure is the high computational cost [30].

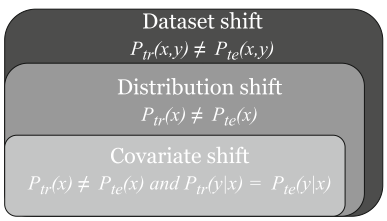


Fig. 1 Visual representation of the relationship between the types of distributional change

2.3 Measuring robustness to distribution shift

Much research is devoted to evaluating ML models' performance under distribution shifts. Such an evaluation would be useful for determining the model's performance after deployment, where the model might perform unexpectedly on unseen data points. A study has investigated the relationship between robustness and complexity of classifiers and concluded that complex classifiers remain more robust to changes than simple classifiers [31]. Similarly, in [32], the authors have tested the robustness of common classifiers in distributional shift situations. The drift was simulated by changing a particular attribute and assessing the influence on the information gain. The authors have concluded that ML models that use more attributes, such as K-Nearest-Neighbors (KNN), in making predictions tend to be more robust than those which rely on fewer attributes, such as Naive Bayes and Logistic Regression.

Several recent studies have investigated the variability in the model's performance across data regions and sub-populations. MANDOLINE framework [33] was proposed to estimate the model's performance under distribution shift. The framework uses a *labeled* validation set from the source distribution and an *unlabeled* set from the target distribution. Users can use their prior knowledge to *group* the data along the possible axes of distribution shift. Then, the reweighted performance estimates are computed. Another framework was proposed to proactively evaluate the model's performance on the *worst-case* distribution [22]. Users choose two sets of variables, *immutable* variables whose distribution should remain unchanged and *mutable* variables whose distribution can be changed. The method identifies the sub-populations with the worst-case risk. Similarly, Sagawa et al. [34] developed a training procedure that uses prior knowledge to form groups in training data and minimizes worst-case loss over these data groups.

In contrast to previous studies, another line of research follows a different approach by predicting the model's performance in distributional shift situations using the regression function. The Average Thresholded Confidence (ATC) method [35] was proposed to obtain a threshold on a model confidence score that enables the prediction of out-of-distribution model's accuracy. In another recent study, Guillory et al. [36] proposed the so-called difference of

confidences (DoC) method that predicts the model's performance under distribution shift. DoC is used to directly estimate the classifier's accuracy gap between the training and the target distributions. When lacking true labels for test sets, the authors [37] have derived distribution statistics that can benefit from the Automatic model Evaluation (AutoEval) problem and estimate the classifier's accuracy.

Evaluating the model's robustness has also been investigated in domain-specific problems. In image processing, the model's performance has been evaluated for many prominent image classification benchmark datasets, such as CIFAR and ImageNet [38–40], MNIST [41]. The problems are constructed by inducing a wide range of distribution shifts. In natural language processing (NLP), Miller et al. [42] evaluated the model's robustness to distribution shift using the popular Stanford Question Answering Dataset (SQuAD) [43]. The authors noted that training models on more out-of-distribution data did not lead to improved robustness for ML models. Despite the rich literature present in the area, the evaluation of the performance of the ML model per region decomposed by test-to-training density ratio has not been yet investigated and is provided by this paper.

3 Experimental settings

In this section, the specifications of the experiments performed using synthetic binary classification problems are illustrated. The experiments are simulated in a two-dimensional input space and a higher input space of four-dimensions. The experimental settings were made more diverse by selecting two- and four-dimensional space settings and omitting the three-dimensional case. And hence to gain a better insight into the performance of the different ML models in varied scenarios.

For the experiment design part, the training dataset is considered to be sampled from a standard Gaussian distribution $X \sim \mathcal{N}_d(0,1)$ throughout all experimental setups. Since any normally distributed variable X, with a particular population mean μ and standard deviation σ, can easily be transformed into a standard Gaussian distribution by applying equation $z = \frac{X-\mu}{\sigma}$. Therefore, the training input density function is as follows:

$$p_{tr}(x) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}X^T X\right), \tag{4}$$

where d is the dimension of the input space.

To generate the test data, several affine transformations are applied to the density of the training data. This method has been widely adopted in the literature to simulate the covariate shift problem in the dataset [10, 21]. The data points are considered to be sampled from a Gaussian distribution $X \sim \mathcal{N}_d(\mu, \Sigma)$ for the test data. Therefore, the test input density function is given by:

$$p_{te}(x) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)\right), \tag{5}$$

where $\mu \in \mathbb{R}^d$ is the mean and $\Sigma \in \mathcal{S}^d_{++}$ is the $d \times d$ symmetric positive-definite covariance matrix whose (i, j)th entry is $Cov[X_i, X_j]$. The statistics μ and Σ represent the parameters of the affine transformations that simulate the drift in the experiments. Specifically, the mean μ simulates a drift induced by translation, and Σ simulates a drift induced by scaling through the variance elements $var(x_i) = \sigma^2_{ii}$, or rotation through the correlation coefficient ρ in the covariance elements $cov(x_i, x_j) = \sigma^2_{ij} = \rho\sigma_i\sigma_j$. For each drift type, the experiments are run in a two-dimensional input space and a higher four-dimensional input space.

Regarding the definition of class posterior probability functions, the formulations that are commonly used in covariate shift research have been followed [44, 45]. In particular, for the two-dimensional datasets settings, two different class posterior probability functions have been defined for classifying the points. The first function is defined as follows:

$$\begin{aligned} F_1 :& p(y = +1 | X = (x_1, x_2)) \\ &= \frac{1}{2}(1 + \tanh(\min(0, x_1) + 4x_2)). \end{aligned} \tag{6}$$

The second class posterior probability function that was designed is more complex to learn and is defined as follows:

$$\begin{aligned} F_2 :& p(y = +1 | X = (x_1, x_2)) \\ &= \frac{1}{2}(1 + \sin(\min(0, x_1) + 2x_2)). \end{aligned} \tag{7}$$

Similarly, for the higher-dimensional datasets settings, Two different class posterior probability functions have been defined as follows:

$$\begin{aligned} F_3 :& p(y = +1 | X = (x_1, x_2, x_3, x_4)) \\ &= \frac{1}{2}(1 + \tanh(\min(0, x_1) - x_2 + 2x_3 + 2x_4)), \end{aligned} \tag{8}$$

$$\begin{aligned} F_4 :& p(y = +1 | X = (x_1, x_2, x_3, x_4)) \\ &= \frac{1}{2}(1 + \sin(\min(0, x_1) + 4x_2 - 3x_3 + 2x_4)), \end{aligned} \tag{9}$$

where $p(y = -1 \mid X) = 1 - p(y = +1 \mid X)$ and the optimal decision boundary is the set of points that satisfy $p(y = -1 \mid X) = p(y = +1 \mid X) = \frac{1}{2}$. For all experiments, training data points of size $N_{tr} = 20000$ are sampled from the probability density function (pdf) defined in Eq. 4, test

data points sampled from the same distribution of size $N_{ts} = 20000$, and another test data points of size $N_{td} = 20000$ sampled from the drifted distribution defined in Eq. 5. Parameter settings and specifications of the two-dimensional experiments are summarized in Table 5 in Appendix 1, and the four-dimensional experiments in Table 6 in Appendix 1. A more detailed description of each experiment is provided in the following subsections.

3.1 Drift simulated by translation

The first set of experiments was created by shifting the mean of the data μ and fixing the covariance matrix Σ. For this type of drift, two settings were created. One setting is characterized by the one-axis translation simulating a *local concept drift* case [46], and the other by the two-axis translation simulating a *global concept drift* case [47]. The details of the experiments are given as follows:

(a) *One-axis Translation:* For the two-dimensional input space, the translation vector $[3 \quad 0]^T$ is used to shift the original mean $[0 \quad 0]^T$. Using the aforementioned drift parameters, two experiments have been created, one experiment whose class posterior probability function defined in Eq. 6, it is referred to as **Exp1.1**, and visualized in Fig. 2a, and the other one using the function defined in Eq. 7, it is referred to as **Exp1.2**, and visualized in Fig. 2b.

(b) *Two-axis Translation:* For the two-dimensional space, the original mean is shifted by the translation vector $[3 \quad 1]^T$. **Exp1.3** refers to the experiment whose class posterior probability function defined in Eq. 6, and **Exp1.4** using the function defined in Eq. 7. **Exp1.3** and **Exp1.4** are visualized in Fig. 2c and d, respectively. Similarly for the four-dimensional data, the original mean is shifted by the translation vector $[0 \quad -2 \quad -1 \quad 1]^T$. The experiment whose class posterior probability function is defined in Eq. 8 is denoted as **Exp2.1**, while **Exp2.2** denotes the experiment whose class posterior probability function is defined in Eq. 9.

3.2 Drift simulated by scaling

On the contrary of the drift explained in the previous Sect. 3.1, the data have been transformed by scaling the covariance matrix Σ and fixing the mean μ of the data. In this set of experiments, two settings have been created, one setting for one-axis scaling simulating a local concept drift situation and another one for two-axis scaling simulating a global concept drift situation. To scale the covariance matrix Σ using the scalars $s_i > 0$ for $i = 1, \ldots, d$ that

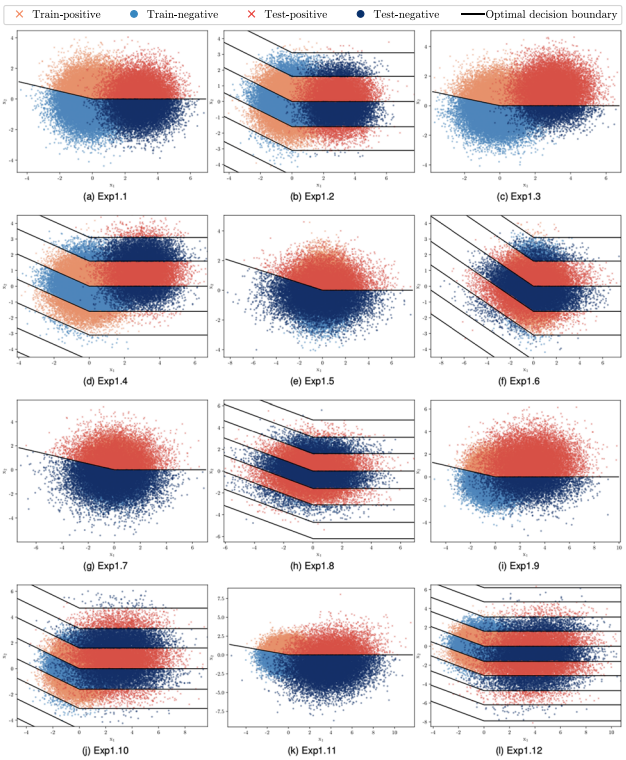


Fig. 2 Training and test data points and the optimal decision boundary of the experiments

represent the scaling factors for each axis direction, the scaling matrix can be written in matrix form as:

$$S_d = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_d \end{bmatrix}, \quad s_i > 0. \tag{10}$$

Therefore, the scaled covariance matrix Σ' can be found by computing the product $\Sigma' = SS\Sigma$, the scaled covariance matrix is cataloged as:

$$\begin{aligned} \Sigma' &= \begin{bmatrix} s_1^2 & 0 & \cdots & 0 \\ 0 & s_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_d^2 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \sigma_{12}^2 & \cdots & \sigma_{1d}^2 \\ \sigma_{21}^2 & \sigma_2^2 & \cdots & \sigma_{2d}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1}^2 & \sigma_{d2}^2 & \cdots & \sigma_d^2 \end{bmatrix} \\ &= \begin{bmatrix} (s_1\sigma_1)^2 & (s_1\sigma_{12})^2 & \cdots & (s_1\sigma_{1d})^2 \\ (s_2\sigma_{21})^2 & (s_2\sigma_2)^2 & \cdots & (s_2\sigma_{2d})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (s_d\sigma_{d1})^2 & (s_d\sigma_{d2})^2 & \cdots & (s_d\sigma_d)^2 \end{bmatrix}, \end{aligned} \tag{11}$$

where s_i is the scaling factor of dimension i and σ_{ij} is the covariance between dimensions i and j. The details of the experiments are given as follows:

(a) *One-axis scaling:* For the two-dimensional input space, the scaling matrix $S = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ is used to transform the original covariance matrix $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Applying Eq. 11, the scaled covariance matrix is $\Sigma' = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$. Using the scaled covariance matrix Σ', two experiments have been created: **Exp1.5**, visualized in Fig. 2e, and **Exp1.6**, visualized in Fig. 2f, whose class posterior probability function defined in Eqs. 6 and 7, respectively.

(b) *Two-axis scaling:* For the two-dimensional space, the original covariance matrix is transformed by the scaling matrix $S = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$, resulting in a scaled covariance matrix $\Sigma' = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$. **Exp1.7**, visualized in Fig. 2g, and **Exp1.8**, visualized in Fig. 2h, refer to the experiments whose class posterior probability function defined in Eqs. 6 and 7, respectively. Similarly for the four-dimensional data, the original covariance matrix is transformed by the scaling matrix: $S = \begin{bmatrix} \sqrt{3} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{3} \end{bmatrix}$. By applying Eq. 10, the scaled covariance matrix is found: $\Sigma' = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$. **Exp2.3** and **Exp2.4** denote the experiments whose class posterior probability function defined in Eqs. 8 and 9, respectively.

3.3 Drift simulated by translation and scaling

In this set of experiments, a combination of linear transformations of the data points is used by translating and scaling the dataset to simulate the drift. For two-dimensional data, translation is formed using the translation vector $[3 \quad 1]^T$, while scaling is formed using the scaling matrix $S = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$, resulting in a scaled covariance matrix $\Sigma' = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$.

Exp1.9, visualized in Fig. 2i, and **Exp1.10**, visualized in Fig. 2j, refer to the experiments whose class posterior probability function defined in Eqs. 6 and 7, respectively.

3.4 Drift simulated by translation, scaling and rotation

In this set of experiments, the drift is simulated by applying three different affine transformations to the dataset by translating, scaling, and rotating the data points. The covariance matrix after scaling and rotating can be calculated by means of the following matrix multiplication:

$$\Sigma'' = R\Sigma' R^T, \quad \Sigma' = SS\Sigma, \tag{12}$$

where R is the rotation matrix. Note that the order of the transformation methods affects the end results. In these experiments, the assumption is made that the data are being scaled and rotated.

For the rotation of the data points, a rotation matrix R is defined to rotate the data points through a desired angle θ about their origin in space. In a two-dimensional space, the general definition of the rotation matrix R is given by the following equation:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{13}$$

For the four-dimensional space, a basic rotation matrix R is given by the following equation:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{14}$$

where θ is the rotation angle.

In our experimental settings, for the two-dimensional case, a translation vector $[4 \quad -1]^T$ is used to shift the original mean μ. For scaling and rotation, the scaling matrix $S = \begin{bmatrix} 2 & 0 \\ 0 & \sqrt{3} \end{bmatrix}$ is used to scale and rotate the data by 45°. This parameter setting would result in the following scaled and rotated covariance matrix by applying Eq. 12 and the rotation matrix defined in Eq. 13: $\Sigma'' = \begin{bmatrix} 3.5 & 0.5 \\ 0.5 & 3.5 \end{bmatrix}$.

Exp1.11, visualized in Fig. 2k, and **Exp1.12**, visualized in Fig. 2l, refer to the experiments whose class posterior probability function defined in Eqs. 6 and 7, respectively.

For four-dimensional data, the original mean μ is shifted through the translation vector $[0 \quad -2 \quad -1 \quad 1]^T$ and scale the data points through the scaling matrix $S = \begin{bmatrix} \sqrt{3} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{3} \end{bmatrix}$, and then rotate the scaled data points by 45°. This parameter setting would result in

the following scaled and rotated covariance matrix by applying Eq. 12 and the rotation matrix defined in Eq. 14: $\Sigma'' = \begin{bmatrix} 2.5 & 0.5 & 0 & 0 \\ 0.5 & 2.5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$. **Exp2.5** and **Exp2.6** denote the experiments whose class posterior probability function is defined in Eqs. 8 and 9, respectively.

4 Evaluation methodology

In this section, the methodology to address the research questions and quantify the robustness of ML models' performance to covariate shift situations is explained. The ML models used for evaluation are first iterated, followed by a description of the evaluation metrics used to assess performance. Lastly, the decomposition of the pdfs by the density ratio regions is described.

4.1 ML models

In this paper, the performance of several popular conventional ML algorithms is compared under covariate shift to address **RQ1** and **RQ2**. To draw conclusions and gain better insight into the robustness to distributional shifts, the chosen ML algorithms have been diversified by selecting classification algorithms from different families.

The selected ML algorithms for our evaluation are the following:

- *Support vector machines (SVM)*: SVM [48] is one of the most popular ML classification algorithms. SVM classifies the points by finding the optimal separating hyperplane between the classes.
- *Logistic regression (LR)*: LR models are used to predict the likelihood of the target class and are usually solved by maximum likelihood methods [49]. For binary classification, the logistic regression model predicts the probability as follows:

$$p(y = \pm 1 \mid \mathbf{x}, \mathbf{w}) = \frac{\mathbf{1}}{\mathbf{1} + \exp(-\mathbf{y}\mathbf{w}^{\mathrm{T}}\mathbf{x})}. \tag{15}$$

- *Random forests (RF)*: Random forests are an ensemble of decision tree models and an extension of the bagging method [50]. It creates a collection of de-correlated decision trees and then aggregates the predictions of each individual tree.
- *Gaussian Naive Bayes (GNB)*: GNB is a simple probabilistic classification algorithm that implements Bayes' Theorem [51]. It involves calculating the posterior probability by applying the Bayes rule:

$$p(y = \pm 1 \mid \mathbf{x}) = \mathbf{p}(\mathbf{y} = \pm \mathbf{1}) \frac{\prod_{\mathbf{i}=\mathbf{1}}^{\mathbf{d}} \mathbf{p}(\mathbf{x_i} \mid \mathbf{y} = \pm \mathbf{1})}{\mathbf{p}(\mathbf{x})}. \tag{16}$$

GNB algorithm assumes that the class-conditional probability distribution, i.e., $p(\mathbf{x} \mid \mathbf{y} = \pm \mathbf{1})$, follows the Gaussian distribution and estimates the mean and standard deviation parameters from the training data.

- *K-Nearest-Neighbors (KNN)*: KNN is a popular algorithm that uses the neighborhood of the data point to make predictions [52]. A voting mechanism is used to determine the class of the new data point. The votes are retrieved from the k data points that are the closest to the new data point.

4.2 Evaluation metrics

Observing the loss rate in the performance of ML models is a commonly adopted approach to measure the robustness of the learning algorithms to distributional changes [53, 54]. The performance is usually measured on two data samples, the training samples and out-of-distribution samples. This paper evaluates the robustness of the ML models used in the experiments using the degradation rate of different performance metrics, including accuracy, F-score, and Matthews coefficient of correlation (MCC). The percentage of data points that are correctly classified represents the accuracy of the model. With **TP** representing the count of true positives, **TN** represents the count of true negatives, **FP** represents the count of false positives, and **FN** the count of false negatives, the accuracy can be expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{17}$$

On the other hand, the F-score, also called the F1 score, is the harmonic mean between the precision and recall metrics [55]. Recall measures the ratio of positive data points that are correctly classified, whereas precision measures the ratio of data points that are classified as positive that are truly positive:

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{18}$$

where

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{19}$$

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{20}$$

Similarly, MCC uses the confusion matrix to score the quality of the classification in the interval $[-1, +1]$, with

-1 denoting perfect misclassification and $+1$ denoting perfect classification, where 0 means prediction more of a random prediction:

$$\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)\cdot(TP+FN)\cdot(TN+FP)\cdot(TN+FN)}}. \tag{21}$$

4.3 PDF domain region decomposition

To address **RQ3** and evaluate the performance on sub-populations of the dataset according to the density ratio between the test and training datasets, the input space domain of the test dataset is decomposed into two regions. The first region, denoted **R1**, is the region with a high density of the training dataset; i.e., where the density ratio is $r = \frac{p_{te}(x)}{p_{tr}(x)} \leq 1$. The other region denoted **R2**, is the region with high density of the test dataset, that is, where the density ratio $r = \frac{p_{te}(x)}{p_{tr}(x)} > 1$. To decompose the pdfs by density regions, the following equation must be solved:

$$p_{te}(x) = p_{tr}(x). \tag{22}$$

Since the distribution is multivariate in our experiments, the pdfs are hypersurfaces of dimension d embedded in $(d+1)$-dim space, and they intersect in a set of d-dim points that lie on a hypersurface $\mathfrak{H}$, also embedded in $(d+1)$-dim space; where d is the dimension of the data points. The solution of Eq. 22 would provide the hypersurface $\mathfrak{H}$ equation.

Solving Eq. 22 for the d-dimensional case of our experiments; $X \sim \mathcal{N}_d(\mu, \Sigma)$:

$$\begin{aligned}\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)\\ = \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}X^TX\right),\end{aligned} \tag{23}$$

results in the hypersurface $\mathfrak{H}$ defined by the following equation:

$$\mathfrak{H}(X) : (X-\mu)^T\Sigma^{-1}(X-\mu) + \log(|\,\Sigma\,|) - X^TX = 0. \tag{24}$$

The data points that lie in the region **R1** are those points which satisfy the inequality:

$$(X-\mu)^T\Sigma^{-1}(X-\mu) + \log(|\,\Sigma\,|) - X^TX \leq 0, \tag{25}$$

otherwise, the points lie in the region **R2**.

Specifically, solving Eq. 22 for the two-dimensional datasets of our experiments; i.e., where the test input density is $\begin{pmatrix} x \\ y \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}\right]$:

$$\begin{aligned}\frac{1}{2\pi(1-\rho^2)^{1/2}\sigma_1\sigma_2}\exp\left\{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right.\right.\\ \left.\left.-2\rho\left(\frac{x-\mu_1}{\sigma_1}\right)\left(\frac{y-\mu_2}{\sigma_2}\right)+\left(\frac{y-\mu_2}{\sigma_2}\right)^2\right]\right\}\\ = \frac{1}{2\pi}\exp\left[-\frac{1}{2}\left(x^2+y^2\right)\right],\end{aligned} \tag{26}$$

results in a 2D curve that lies on a surface **S**. We obtain the following equation defining the surface **S** in the 3D space:

$$\begin{aligned}S(x,y) : &\left[\frac{-1}{2}+\frac{b}{\sigma_1^2}\right]x^2 + \left[\frac{-1}{2}+\frac{b}{\sigma_2^2}\right]y^2\\ &-\left[\frac{2b\mu_1}{\sigma_1^2}-\frac{2b\rho\mu_2}{\sigma_1\sigma_2}\right]x-\left[\frac{2b\mu_2}{\sigma_2^2}-\frac{2b\rho\mu_1}{\sigma_1\sigma_2}\right]y\\ &-\frac{2b\rho}{\sigma_1\sigma_2}xy = r,\end{aligned} \tag{27}$$

where:

$$r = c + \frac{2b\rho}{\sigma_1\sigma_2}\mu_1\mu_2, \tag{28}$$

$$c = a - \frac{b\mu_1^2}{\sigma_1^2} - \frac{b\mu_2^2}{\sigma_2^2}, \tag{29}$$

$$b = \frac{1}{2(1-\rho^2)}, \tag{30}$$

$$a = \log\frac{1}{\sigma_1\sigma_2\sqrt{1-\rho^2}}. \tag{31}$$

The data points that lie in the region **R1** are those points which satisfy the inequality:

$$\begin{aligned}\left[\frac{-1}{2}+\frac{b}{\sigma_1^2}\right]x^2 + \left[\frac{-1}{2}+\frac{b}{\sigma_2^2}\right]y^2 - \left[\frac{2b\mu_1}{\sigma_1^2}-\frac{2b\rho\mu_2}{\sigma_1\sigma_2}\right]\\ x-\left[\frac{2b\mu_2}{\sigma_2^2}-\frac{2b\rho\mu_1}{\sigma_1\sigma_2}\right]y-\frac{2b\rho}{\sigma_1\sigma_2}xy \geq r,\end{aligned} \tag{32}$$

otherwise, the points lie in the region **R2**.

The surface **S** has a particular shape based on the type of drift simulated in the pdf of the dataset. Specifically, in the case of drifts simulated by two-axis translation, the surface **S** is a vertical plane in the 3D space defined by the following equation:

$$2\mu_1 x - \mu_1^2 + 2\mu_2 y - \mu_2^2 = 0. \tag{33}$$

For a translation on one axis, the surface **S** is a vertical plane that is parallel to the xz-plane or the yz-plane. In particular, if the translation is along the x-axis, the surface **S** is a vertical plane parallel to the yz-plane and is given by equation:

$$x = \frac{\mu_1}{2}. \tag{34}$$

On the contrary, if the translation is along the y-axis, the surface **S** is a vertical plane parallel to the xz-plane and is given by equation:

$$y = \frac{\mu_2}{2}. \tag{35}$$

In the case of two-axis scaling; $\sigma_1 > 1$ and $\sigma_2 > 1$; the surface **S** is an elliptic cylinder defined by equation:

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1, \tag{36}$$

where:

$$\alpha^2 = \frac{2\sigma_1^2 \log(\sigma_1\sigma_2)}{\sigma_1^2 - 1}, \tag{37}$$

and:

$$\beta^2 = \frac{2\sigma_2^2 \log(\sigma_1\sigma_2)}{\sigma_2^2 - 1}. \tag{38}$$

In the special case of scaling at the same magnitude in both dimensions; i.e., $\sigma_1 = \sigma_2 \Rightarrow \alpha = \beta$; the surface **S** is a right-circular cylinder.

In the case of a one-axis scaling, the two pdfs intersect in curves that lie on two parallel planes. If $\sigma_1 = 1$; these parallel planes are defined by the following equation:

$$y^2 = \frac{2\sigma_2^2 \log \sigma_2}{\sigma_2^2 - 1}. \tag{39}$$

Whereas if $\sigma_2 = 1$; the parallel planes are defined by the following equation:

$$x^2 = \frac{2\sigma_1^2 \log \sigma_1}{\sigma_1^2 - 1}. \tag{40}$$

Similarly, in the case of translation and scaling at the same time, the surface where the two pdfs intersect is a cylinder that is centered at $\left(\frac{\mu_1}{1-\sigma_1^2}, \frac{\mu_2}{1-\sigma_2^2}\right)$. Figure 3 shows the densities of the training and test data along with the corresponding intersection surface between the two pdfs of the two-dimensional experiments.

5 Empirical results and analysis

The results of the experiments detailed in Sect. 3 will be scrutinized in this section to answer the research questions. Specifically, the overall results of applying the ML models to different datasets characterized by various types of drift will be presented. Additionally, a model-wise performance analysis will follow. To execute the experiments, the scikit-learn library[1] was utilized, a widely used open-source Python ML package that provides the implementation of several classification algorithms.

5.1 Degradation rate in evaluation metrics

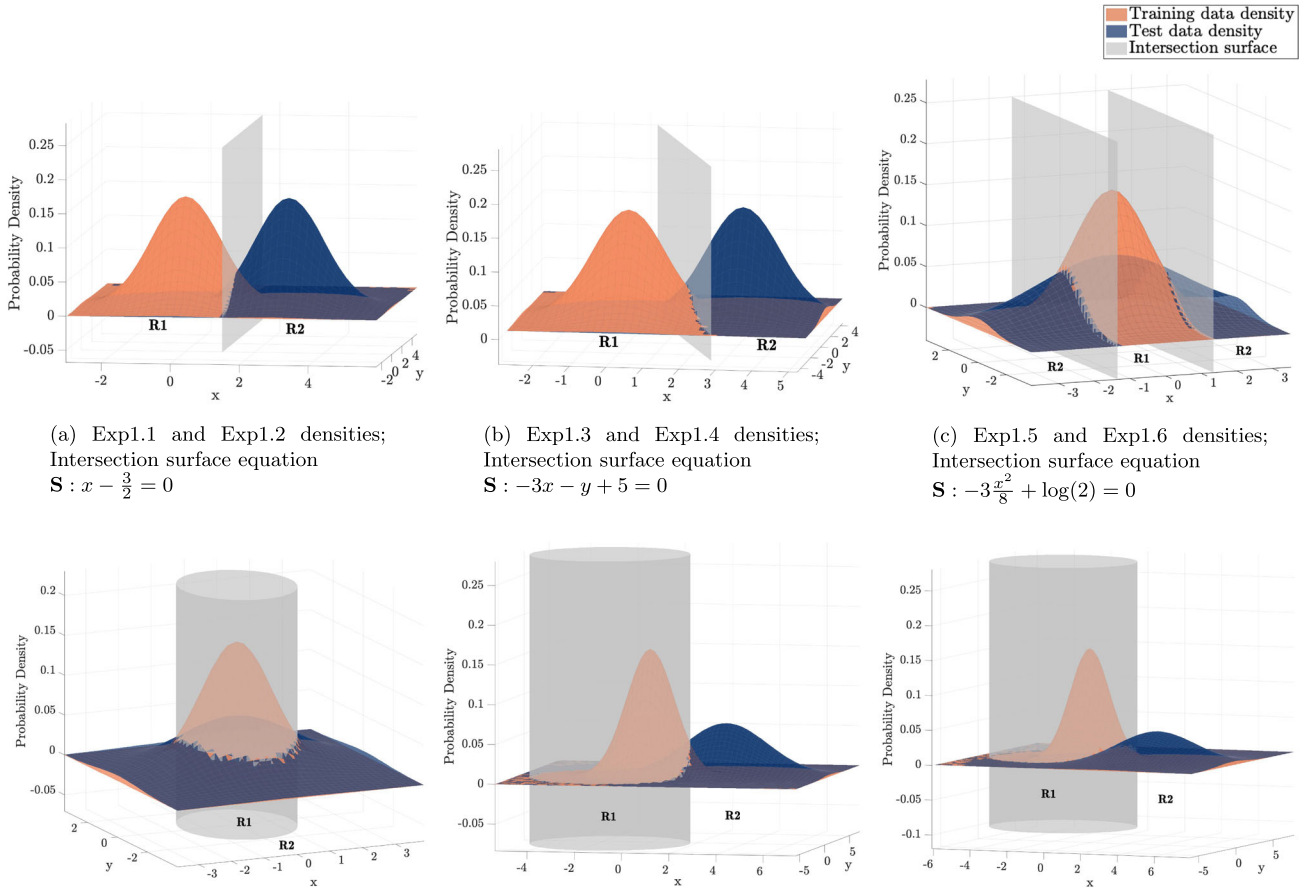
Changes in data distribution usually lead to a degradation of performance metrics [56]. To measure the performance loss after the drift and address **RQ1**, the degradation rate is calculated, this rate represents the percentage of the performance drop of the ML model on the drifted dataset. Table 1 recapitulates the results of the two-dimensional experiments across the different performance metrics of the ML models.

The first observation from the results is that the Random Forests algorithm demonstrated a high robustness level in most experiments across all performance metrics. This is an advantage of ensemble-based methods that can alleviate the biases of the datasets by combining individual classifiers' votes [57]. However, this is not the case in the four-dimensional experiment results summarized in Table 2. RF has shown much poorer robustness in higher-dimensional data. This could be attributable to the fact that learners which use more covariates to make predictions tend to show greater robustness than those that rely on a subset of covariates [32]. The default value provided by the scikit-learn library was used to select the parameter that controls the subset of covariates used to find the best tree split in the Random Forests algorithm. The library assigns the square root value of the total covariates to the parameter. Furthermore, LR showed the highest degradation rate in most of the two-dimensional experiments, which is a recognized drawback of the algorithm in the case of out-of-sample predictions, where the maximum likelihood estimator tends to display poor performance due to the overfitting effect [58].

For the rest of the models, there are no precise general conclusions that can be deduced, since each model's degradation rates were affected by the settings of the experiments. It can be seen from Table 1 that SVM is the model with the second lowest degradation rate after RF in most cases where multiple drifts were simulated, that is, in experiments **Exp1.9** to **Exp1.12**. On the other hand, GNB and KNN are highly affected by multiple drifts since $p(x)$ has been altered to a great extent. Each model's performance will be further investigated in the following subsection.

In terms of the effects of the complexity of the decision surface, it can be seen that the performance on the test data is worse and the degradation rates are higher in the four-dimensional experiments than in the two-dimensional cases

[1] https://scikit-learn.org/.



(a) Exp1.1 and Exp1.2 densities; Intersection surface equation $\mathbf{S}: x - \frac{3}{2} = 0$

(b) Exp1.3 and Exp1.4 densities; Intersection surface equation $\mathbf{S}: -3x - y + 5 = 0$

(c) Exp1.5 and Exp1.6 densities; Intersection surface equation $\mathbf{S}: -3\frac{x^2}{8} + \log(2) = 0$

(d) Exp1.7 and Exp1.8 densities; Intersection surface equation $\mathbf{S}: \frac{-x^2}{3} - \frac{y^2}{4} + \frac{\log(6)}{2} = 0$

(e) Exp1.9 and Exp1.10 densities; Intersection surface equation $\mathbf{S}: \frac{-x^2}{3} - x - \frac{y^2}{4} - \frac{y}{2} + \frac{\log(6)}{2} + \frac{7}{4} = 0$

(f) Exp1.11 and Exp1.12 densities; Intersection surface equation $\mathbf{S}: \frac{-17x^2}{48} - \frac{xy}{24} - \frac{29x}{24} - \frac{17y^2}{48} + \frac{11y}{24} + \frac{\log(12)}{2} + \frac{127}{48} = 0$

Fig. 3 Training and test input densities decomposed by the **R1** and **R2** regions and the intersection surface

when using the more complex decision function F_2, except for the LR and GNB algorithms, as they display low degradation rates due to poor performance on the original dataset. This can be associated with the positive correlation between the complexity of the decision boundary and the dimensional space of the problem [59]. Furthermore, the effect of the complexity of the decision boundary is more discernible in experiments with mixed drifts, i.e., a higher magnitude of drift.

To see how the performance metrics are acting in the experiments, the correlation matrices of the two- and four-dimensional experiments between the degradation rates of different performance metrics used to evaluate the models have been plotted. The correlation matrices are shown in Figs. 4 and 5. From the matrices it can be seen that a strong correlation between the degradation rates of the performance metrics in the two-dimensional experiments (see Fig. 4). This indicates that the performance has been stirred in the same direction across the different metrics. However, this correlation between the degradation rates is lower in the four-dimensional experiments (see Fig. 5). Furthermore, by comparing the degradation rates in Tables 1 and 2, it can be seen that the MCC metric is the most affected by changes, showing higher rates than accuracy and F-score. However, accuracy and F-score have shown similar degradation rates in most experiments.

5.2 Model-wise analysis

In this section, the ML model's performance in the different experimental settings to address **RQ2** is scrutinized. The model-wise results of the two-dimensional experiments are organized in Fig. 6, while Fig. 7 details the results of the four-dimensional experiments. The common ground between all ML models in the two-dimensional datasets is that they are mostly damaged in the experiments formulated by three types of transformation, i.e., in experiments **Exp1.11** and **Exp1.12**. Whereas in the four-

Table 1 Two-dimensional robustness evaluation results

Experiment	ML model	Same distribution			Drifted distribution			Degradation rate		
		Acc.	F1	MCC	Acc.	F1	MCC	**Acc. (%)**	**F1 (%)**	**MCC (%)**
Exp1.1	**SVM**	0.998	0.9978	0.9959	0.9846	0.9847	0.9695	1.34	1.31	2.65
	LR	0.9768	0.975	0.9535	0.8916	0.9014	0.8022	8.72	7.55	15.87
	RF	0.9998	0.9998	0.9997	0.9987	0.9986	0.9974	**0.11***	**0.12***	**0.23***
	GNB	0.971	0.9686	0.9417	0.9523	0.9495	0.9087	1.93	1.97	3.50
	KNN	0.9978	0.9977	0.9957	0.9634	0.9638	0.9274	3.45	3.40	6.86
Exp1.2	**SVM**	0.9976	0.9973	0.9951	0.9612	0.9765	0.8742	3.65	2.09	12.15
	LR	0.9786	0.9768	0.9569	0.9457	0.9688	0.7825	3.36	0.82	18.23
	RF	1.0	1.0	1.0	0.9988	0.9988	0.9977	**0.12***	**0.12***	**0.23***
	GNB	0.9711	0.9687	0.9419	0.9663	0.9796	0.8886	0.49	1.13	5.66
	KNN	0.9973	0.9971	0.9946	0.9823	0.9896	0.9318	1.50	0.75	6.31
Exp1.3	**SVM**	0.997	0.9967	0.9939	0.986	0.9832	0.9712	1.10	1.35	2.28
	LR	0.9768	0.9747	0.9533	0.955	0.9487	0.9107	2.23	2.67	4.47
	RF	0.9983	0.9981	0.9966	0.9931	0.9919	0.986	**0.52***	**0.62***	**1.06***
	GNB	0.9655	0.9622	0.9305	0.9502	0.9415	0.8982	1.58	2.15	3.47
	KNN	0.9962	0.9958	0.9922	0.9896	0.9878	0.9788	0.66	0.80	1.35
Exp1.4	**SVM**	0.9966	0.9963	0.9932	0.9936	0.993	0.9872	0.30	0.33	0.60
	LR	0.9766	0.9747	0.953	0.9699	0.9675	0.9402	0.69	0.74	1.34
	RF	0.9983	0.9982	0.9966	0.997	0.9968	0.9941	0.13	**0.14***	**0.25***
	GNB	0.97	0.9674	0.9397	0.9689	0.9657	0.9373	**0.11***	0.18	0.26
	KNN	0.9965	0.9962	0.993	0.9922	0.9914	0.9844	0.43	0.48	0.87
Exp1.5	**SVM**	0.9982	0.998	0.9964	0.872	0.9077	0.7375	12.64	9.05	25.98
	LR	0.9776	0.9756	0.9548	0.944	0.9643	0.8463	3.44	1.16	11.36
	RF	0.9998	0.9999	0.9996	0.9988	0.9988	0.9977	**0.10***	**0.11***	**0.19***
	GNB	0.9684	0.9656	0.9365	0.9539	0.9686	0.888	1.50	0.31	5.18
	KNN	0.9969	0.9966	0.9938	0.9906	0.9938	0.9744	0.63	0.28	1.95
Exp1.6	**SVM**	0.9978	0.9976	0.9956	0.8842	0.7538	0.7177	11.39	24.44	27.91
	LR	0.9752	0.9729	0.9501	0.9232	0.8834	0.8398	5.33	9.20	11.61
	RF	1.0	0.9999	0.9999	0.9984	0.9982	0.9967	**0.16***	**0.17***	**0.32***
	GNB	0.9696	0.9667	0.9389	0.9532	0.9127	0.8874	1.69	5.59	5.49
	KNN	0.9958	0.9954	0.9914	0.9868	0.9773	0.9681	0.90	1.82	2.35
Exp1.7	**SVM**	0.9907	0.9902	0.9814	0.9478	0.9481	0.8956	4.33	4.25	8.74
	LR	0.8198	0.8036	0.6391	0.7532	0.7816	0.5241	8.12	2.74	17.99
	RF	0.9998	0.9998	0.9996	0.9956	0.9954	0.9913	**0.42***	**0.44***	**0.84***
	GNB	0.7645	0.7137	0.5408	0.7558	0.7197	0.5301	1.14	0.84	1.98
	KNN	0.9906	0.9901	0.9812	0.9242	0.9254	0.8487	6.70	6.53	13.50
Exp1.8	**SVM**	0.9903	0.9899	0.9806	0.9325	0.94	0.8636	5.84	5.04	11.93
	LR	0.8231	0.8046	0.6474	0.6383	0.7581	0.2909	22.45	5.78	55.07
	RF	0.9952	0.995	0.9905	0.9868	0.9884	0.9735	**0.84***	**0.66***	**1.72***
	GNB	0.769	0.7192	0.5524	0.7049	0.7885	0.418	8.34	9.64	24.33
	KNN	0.9918	0.9914	0.9835	0.8882	0.9072	0.7734	10.45	8.49	21.36
Exp1.9	**SVM**	0.99	0.9894	0.9798	0.9698	0.9683	0.9394	2.04	2.13	4.12
	LR	0.8188	0.7997	0.6375	0.7405	0.714	0.4803	9.56	10.72	24.66
	RF	0.9942	0.9939	0.9884	0.9805	0.9796	0.961	**1.38***	**1.44***	**2.77***
	GNB	0.7664	0.7138	0.5449	0.7264	0.66	0.4683	5.22	7.54	14.06
	KNN	0.9916	0.9912	0.9833	0.9625	0.9607	0.9249	2.93	3.08	5.94

Table 1 (continued)

Experiment	ML model	Same distribution			Drifted distribution			Degradation rate		
		Acc.	F1	MCC	Acc.	F1	MCC	**Acc. (%)**	**F1 (%)**	**MCC (%)**
Exp1.10	**SVM**	0.9906	0.99	0.9811	0.9674	0.9671	0.935	2.34	2.31	4.70
	LR	0.8198	0.7985	0.6392	0.6716	0.6548	0.3445	18.08	18.00	46.10
	RF	0.9954	0.9952	0.9909	0.9792	0.9791	0.9584	**1.63***	**1.62***	**3.28***
	GNB	0.7677	0.7119	0.547	0.6486	0.6106	0.3023	15.51	14.23	44.73
	KNN	0.9918	0.9912	0.9834	0.9618	0.9618	0.9236	3.02	2.97	6.08
Exp1.11	**SVM**	0.9902	0.9898	0.9805	0.849	0.8452	0.7006	14.26	14.61	28.55
	LR	0.822	0.8062	0.6434	0.6114	0.7094	0.2809	25.62	12.01	56.34
	RF	0.9952	0.995	0.9905	0.993	0.9931	0.986	**0.22***	**0.19***	**0.45***
	GNB	0.7721	0.7315	0.5559	0.6644	0.7242	0.366	13.95	1.01	34.16
	KNN	0.9916	0.9912	0.9832	0.8474	0.853	0.6951	14.54	13.94	29.30
Exp1.12	**SVM**	0.9909	0.9904	0.9817	0.7302	0.6574	0.5161	26.31	33.62	47.43
	LR	0.8208	0.7998	0.6421	0.5327	0.4914	0.0681	35.10	38.56	89.39
	RF	0.9958	0.9955	0.9915	0.9752	0.9748	0.9512	**2.07***	**2.08***	**4.06***
	GNB	0.7709	0.7183	0.5548	0.4958	0.3764	0.0051	35.69	47.60	100.92
	KNN	0.9916	0.9912	0.9833	0.728	0.7185	0.458	26.58	27.51	53.42

The bold asterisk indicates the machine learning model with the lowest degradation rate compared to other models in the same experiment

dimensional datasets, the models are more affected by the complexity of the classification function, i.e., in the experiments where the true classification function is F_2, namely, the experiments with even numbers.

In the two-dimensional experiments, as shown in Fig. 6a, the SVM algorithm was found to be more susceptible to scaling than to translating the data distribution, i.e., in experiments **Exp1.5** −1.8. This is interwoven with the decision boundary built by the SVM algorithm for the classification problem. Scaling the covariance matrix will lead to changes in the dispersion of the data points that could require a change in the shape of the decision boundary, whereas translating the mean would require shifting the decision boundary without adjusting its shape. In contrast, as shown in Fig. 6b, the LR algorithm has shown poorer performance in the translated one-axis distribution and the scaled two-axis distribution. RF algorithm has exhibited a steady degradation rate that increases with the complexity of the drift simulated in the dataset, as shown in Fig. 6c. Similarly to RF, GNB and KNN algorithms have shown relative degradation rates to the types of drift simulated in the data distribution, by scoring low degradation rates in experiments that include single drifts, and higher rates in experiments that include mixed types of drift, as shown in Fig. 6d and e.

In four-dimensional experiments, it is clear that performance degradation is most dominant by the complexity of the class posterior probability function used across all ML models; see Fig. 7. Specifically, the MCC metric was the most sensitive, showing the highest degradation rates in all experiments. However, the degradation rates of accuracy and F-score are quite close to each other.

5.3 Region-based performance evaluation

After evaluating the overall performance of the ML models in the presence of distributional shifts, the performance of the ML models have been evaluated on subpopulations of the data to address **RQ3**. To ensure an impartial evaluation, we assessed the performance on 10,000 data points in each region. The findings of the region-based evaluation of the two-dimensional experiments are shown in Tables 3 and 4 for the four-dimensional experiments.

From the results, it can be seen that the performance of the different ML models is very high in the **R1** region, that is, in the region with a high training input density in most experiments. However, the models are more erroneous in the **R2** region, that is, in the region with a low training input density. This is an important observation that the models tend to be accurate in the high-training input density regions and work well in such regions, even though the distribution has changed. Furthermore, upon comparing Tables 1 and 3, and Tables 2 and 4, it can be observed that the results obtained from data sampled from the same training distribution are similar to the outcomes observed in region **R1**. This is because the majority of data points are concentrated in this region due to a higher training input density. Likewise, for the drifted distribution, the results

Table 2 Four-dimensional robustness evaluation results

Experiment	ML model	Same distribution			Drifted distribution			Degradation rate		
		Acc.	F1	MCC	Acc.	F1	MCC	Acc. (%)	F1 (%)	MCC (%)
Exp2.1	**SVM**	0.993	0.9923	0.9859	0.9739	0.9812	0.9385	1.92	1.12	4.81
	LR	0.9691	0.9714	0.9795	0.9662	0.9377	0.9327	**0.24***	1.38	**0.53***
	RF	0.9768	0.9746	0.9533	0.9337	0.9517	0.8471	4.41	2.35	11.14
	GNB	0.9641	0.9599	0.9281	0.92	0.9391	0.8368	4.57	2.17	9.84
	KNN	0.974	0.9716	0.9477	0.9592	0.9706	0.9038	1.52	**0.10***	4.63
Exp2.2	**SVM**	0.7518	0.7449	0.5038	0.6056	0.6132	0.2113	19.45	17.68	58.06
	LR	0.5002	0.6346	0.0053	0.4982	0.4689	0.0004	**0.40***	35.34	92.45
	RF	0.9166	0.9158	0.8333	0.7844	0.7811	0.5692	14.42	14.71	**31.69***
	GNB	0.501	0.5155	0.0091	0.4952	0.5124	0.002	1.17	**0.60***	78.02
	KNN	0.904	0.9029	0.808	0.7756	0.77	0.5518	14.20	14.72	31.71
Exp2.3	**SVM**	0.9938	0.993	0.9874	0.9818	0.9793	0.9632	1.21	1.38	2.45
	LR	0.9704	0.9668	0.9401	0.9627	0.9591	0.926	0.79	0.80	1.50
	RF	0.9796	0.9772	0.9587	0.9652	0.961	0.9296	1.47	1.66	3.04
	GNB	0.9708	0.9666	0.9411	0.9674	0.9626	0.9341	**0.35***	**0.41***	**0.74***
	KNN	0.9768	0.974	0.953	0.969	0.9653	0.9374	0.80	0.89	1.64
Exp2.4	**SVM**	0.7512	0.7468	0.5026	0.5994	0.5875	0.1987	20.21	21.33	60.47
	LR	0.5011	0.4817	0.0038	0.4983	0.4734	0.0017	0.56	**1.75***	55.26
	RF	0.9216	0.9211	0.8433	0.7838	0.7821	0.5676	14.95	15.09	32.69
	GNB	0.5042	0.5182	0.0088	0.5017	0.4014	0.0013	**0.50***	22.54	85.23
	KNN	0.9008	0.9001	0.8015	0.7741	0.7731	0.5482	14.07	14.11	**31.60***
Exp2.5	**SVM**	0.9935	0.9927	0.9869	0.9174	0.9285	0.8411	7.66	6.47	14.77
	LR	0.9698	0.9737	0.9389	0.9673	0.9661	0.9312	**0.26***	0.79	**0.82***
	RF	0.976	0.9731	0.9514	0.9301	0.9413	0.8575	4.70	3.27	9.87
	GNB	0.9686	0.9642	0.9369	0.9445	0.9527	0.8914	2.49	1.19	4.86
	KNN	0.974	0.9709	0.9473	0.959	0.9669	0.9133	1.54	**0.41***	3.59
Exp2.6	**SVM**	0.758	0.7558	0.5161	0.5383	0.5558	0.077	28.98	26.46	85.08
	LR	0.5025	0.5234	0.0048	0.498	0.2368	0.0037	0.90	54.76	**22.92***
	RF	0.92	0.92	0.8401	0.6544	0.6663	0.3099	28.87	27.58	63.11
	GNB	0.5012	0.51	0.006	0.497	0.4913	0.0026	**0.85***	**3.81***	56.67
	KNN	0.9014	0.9015	0.8029	0.6662	0.6837	0.3348	26.09	24.16	58.30

The bold asterisk indicates the machine learning model with the lowest degradation rate compared to other models in the same experiment

are similar to those obtained from region **R2** since the test density is high in this region, resulting in a higher number of data points contributing to the experiments.

This result can be beneficial for interpreting where models tend to fail when making predictions. Additionally, it can be used to develop an adaptive solution for covariate shift situations, where a region-based importance weight can be introduced to correct the bias and ensure high-robustness in the affected regions. Moreover, introducing such region-based importance weights can reduce the cost of computing point-wise importance weights to region-wise importance weights in the cases of large datasets. This would also solve the drawback of the conventional importance weighting technique, where $w(x)$ can become unbounded and very large for a few points, leading to large variances of the estimated ratios [60].

In the two-dimensional experiments, it can be seen that the LR and GNB algorithms perform poorly in the **R1** region in the experiments with mixed drifts. However, they do not tend to be sensitive to this region in the four-dimensional experiments, as they exhibit similar performance across the evaluation metrics in all the experiments. It is also noteworthy to see that the RF algorithm has a high performance in **R1** in all two- and four-dimensional experiments in all evaluation metrics, and most of the misclassifications were in the **R2** region. Similarly to the overall robustness results presented in the previous section, the MCC appears to be the most susceptible metric

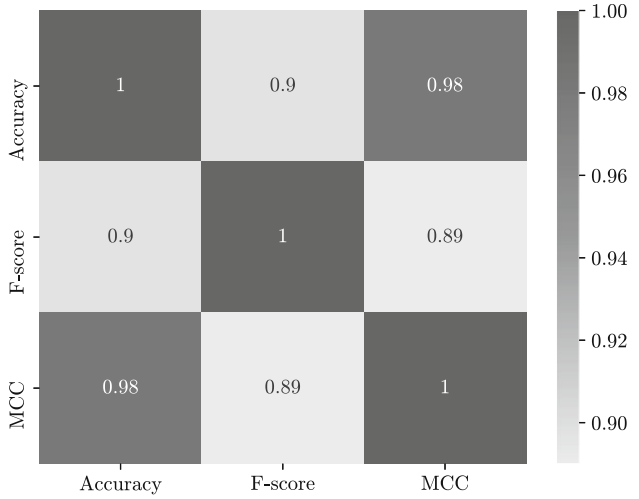


Fig. 4 Correlation between degradation rates of performance metrics in the two-dimensional experiments

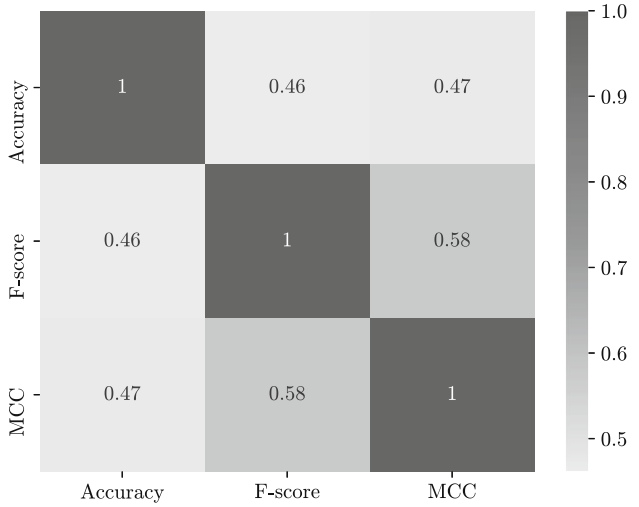


Fig. 5 Correlation between degradation rates of performance metrics in the four-dimensional experiments

compared to accuracy and F-score in the **R2** region, showing higher performance loss rates. Furthermore, the complexity of the decision function has been shown to have a higher impact on the **R2** region.

The performance of ML models in the context of region-based analysis was further investigated by conducting an examination of the data points based on the density ratio values. Specifically, the region **R1** consists of data points with a density ratio of a positive number that is less than 1, while region **R2** includes data points with a density ratio higher than 1. To provide a comprehensive overview of the density ratio values in the experiments, quartiles were calculated, which can be found summarized in Table 7 for the two-dimensional experiments, and Table 8 in the appendix Sect. B. The performance metric used in this

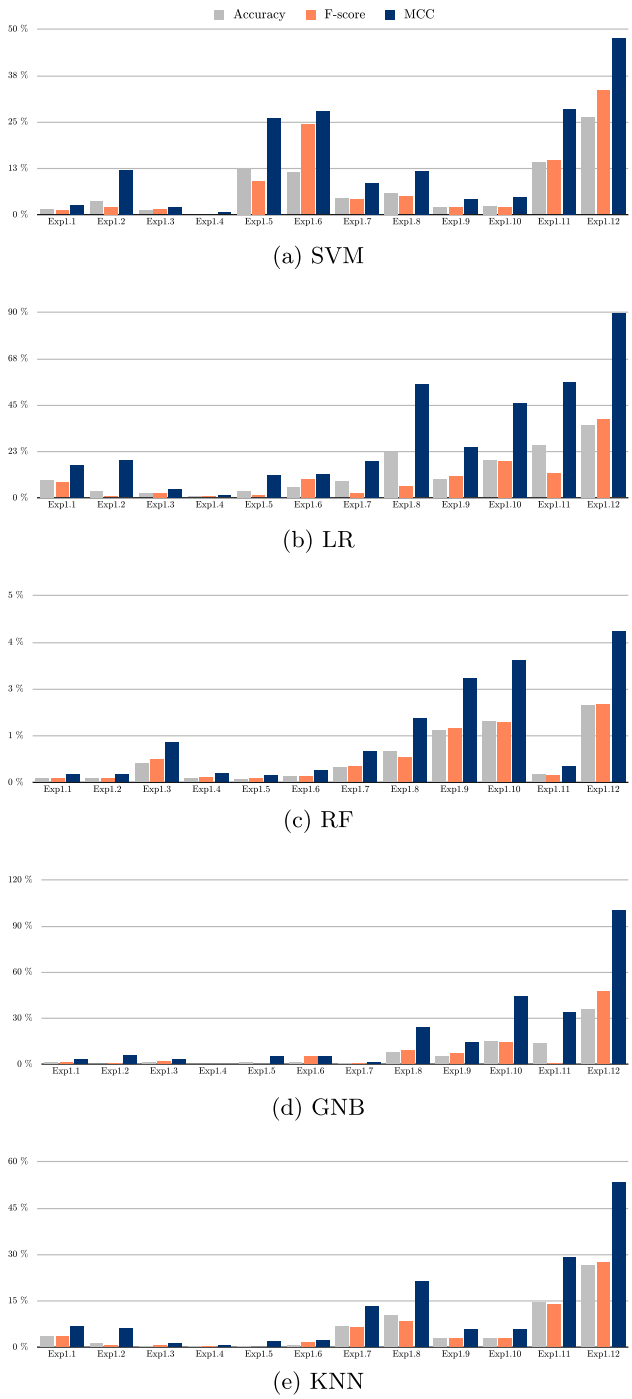


Fig. 6 Model-wise performance evaluation in two-dimensional experiments

analysis is the accuracy, as it was found to have high correlations with other performance metrics, see Tables 4 and 5. Therefore, similar insights could be obtained by analyzing the different performance metrics.

The performance of the ML models in each quartile of the density ratio for the two-dimensional experiments is shown in Fig. 8. As can be observed, in most cases, the

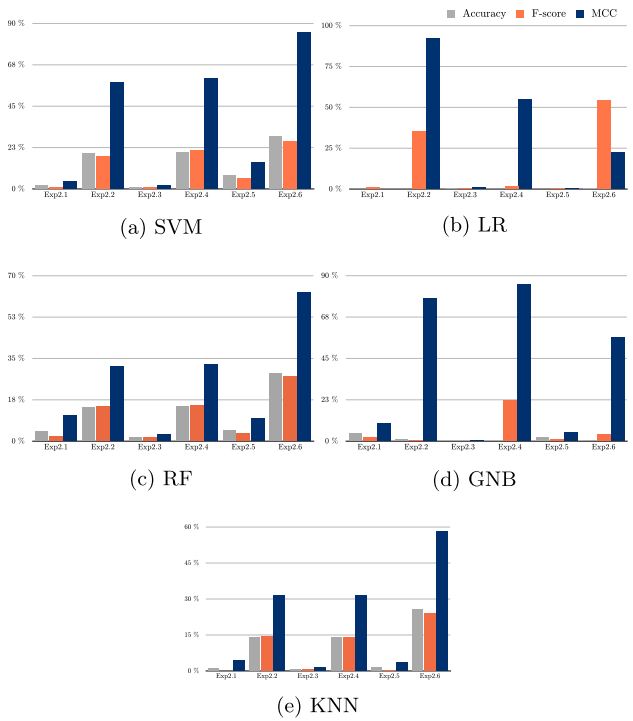


Fig. 7 Model-wise performance evaluation in four-dimensional experiments

performance of the models degrades significantly as the density ratio increases, indicating that the models become less reliable as we move further from the training domain region. In particular, SVM exhibits the highest level of sensitivity as the density ratio increases, while RF demonstrates the highest level of robustness at high density ratio values. Similar trends are observed in the four-dimensional experiments, as shown in Fig. 9. However, LR and NB notably exhibit relatively stable performance across the density ratio quartiles in most experiments.

6 Threats to validity

Like any other empirical study, the main threat to external validity is related to the generalization of the findings. Although the scope of this paper is to compare the performance of different ML algorithms under distributional shifts, the default parameters provided by the scikit-learn library have been used. Using a fixed set of parameters would narrow the focus to monitor the performance of the different algorithms rather than to monitor the same algorithm under different parameter settings. Furthermore, taking the impact of the models' parameters into account as a factor, in addition to the drift parameters, would extend the parameter space of the experiments expansively. However, different parameter settings would yield different results clearly, but we argue that the concluding remarks would be the same.

On the other hand, the experiments have been conducted using synthetic data, which might not always be similar and mimic the real-world context. As a result, this may affect the generalization of our findings to real-world datasets. Despite this, in the literature, synthetic datasets are widely used in research on changing environments [9, 20], as they include information about drift, such as drift time and type [61, 62], and they provide valuable insights and a basis for further research. Moreover, real-world datasets lack annotations of ground-truth changes, and their underlying PDFs are often unknown [63]. Additionally, these generated datasets allow us to work on controlled classification settings and simulate various drift situations, and draw concrete conclusions about the performance of the ML models.

7 Conclusion

In this paper, the robustness of the performance of popular machine learning algorithms was investigated in covariate shift situations through an exhaustive comparative study. Several problems using a tractable classification framework have been generated. Each problem is parameterized by specific conditions that simulate a particular type of change. Our results show that the Random Forests algorithm is the most robust algorithm in the two-dimensional experiments, showing the lowest degradation rates in the evaluation metrics. The complexity of the classification rule has a major impact on the performance in higher-dimensional experiments.

The decomposition of the input space domain into regions based on the test-to-training density ratio have allowed to diagnose the models' performance on subpopulations of the data points. The experimental results show the high bias of the algorithms in the regions where the training input density is high, despite inducing a change in the distribution of the test data points. Our future work will consist of using this observation to develop a covariate shift solution based on region-based importance weights.

Additionally, and to address the threats presented in Sect. 6, we would like to investigate the effects of the models' hyperparameters in coping with distributional changes. Also, the impact of using real-world datasets in our evaluation environment. Another potential continuation for future research could be the evaluation of deep learning-based models to further validate the conclusions drawn from conventional ML models.

Table 3 Comparison of region-based performance in the two-dimensional experiments

Exp #	ML model	R1 region performance			R2 region performance		
		Acc.	F-score	MCC	Acc.	F-score	MCC
Exp1.1	**SVM**	**1.0***	**1.0***	**1.0***	0.9835	0.9836	0.9673
	LR	**0.9846***	**0.9846***	**0.9692***	0.8848	0.8959	0.7912
	RF	**1.0***	**1.0***	**1.0***	0.9998	0.9998	0.9997
	GNB	**0.9721***	**0.9713***	**0.9457***	0.9509	0.9478	0.906
	KNN	**0.9949***	**0.9948***	**0.9897***	0.961	0.9615	0.923
Exp1.2	**SVM**	**0.9965***	**0.9974***	**0.9923***	0.9591	0.9755	0.8616
	LR	**0.9784***	**0.9838***	**0.9518***	0.9437	0.9681	0.7571
	RF	**1.0***	**1.0***	**1.0***	0.9981	0.998	0.997
	GNB	**0.9697***	0.9765	**0.9361***	0.9661	**0.9798***	0.8821
	KNN	**0.9974***	**0.998***	**0.9942***	0.9814	0.9892	0.9234
Exp1.3	**SVM**	**0.9977***	**0.9975***	**0.9954***	0.9739	0.9653	0.9446
	LR	**0.9826***	**0.981***	**0.9651***	0.9268	0.9125	0.8601
	RF	**0.9989***	**0.9988***	**0.9978***	0.9871	0.9834	0.9733
	GNB	**0.9708***	**0.968***	**0.9412***	0.9291	0.91	0.8527
	KNN	**0.9967***	**0.9965***	**0.9934***	0.9824	0.9772	0.963
Exp1.4	**SVM**	**0.9966***	**0.9963***	**0.9931***	0.9911	0.9899	0.982
	LR	**0.9807***	**0.9792***	**0.9612***	0.9605	0.9573	0.9236
	RF	**0.9988***	**0.9987***	**0.9976***	0.9955	0.9949	0.9909
	GNB	**0.9727***	**0.9704***	**0.9454***	0.9656	0.9615	0.9305
	KNN	**0.9968***	**0.9966***	**0.9935***	0.9883	0.9866	0.9764
Exp1.5	**SVM**	**0.9993***	**0.9994***	**0.9986***	0.8499	0.8941	0.6906
	LR	**0.9838***	**0.9867***	**0.9661***	0.9371	0.9614	0.8104
	RF	**1.0***	**1.0***	**1.0***	0.9989	0.9982	0.9979
	GNB	**0.9774***	**0.9811***	**0.9536***	0.9498	0.9669	0.8716
	KNN	**0.9973***	**0.9978***	**0.9944***	0.9894	0.9933	0.9688
Exp1.6	**SVM**	**0.9981***	**0.9974***	**0.9958***	0.8746	0.7204	0.6879
	LR	**0.9819***	**0.9758***	**0.9617***	0.9183	0.8745	0.8296
	RF	**1.0***	**1.0***	**1.0***	0.9981	0.998	0.996
	GNB	**0.9806***	**0.9729***	**0.9587***	0.951	0.9057	0.8801
	KNN	**0.9974***	**0.9965***	**0.9944***	0.986	0.9752	0.9654
Exp1.7	**SVM**	**0.9918***	**0.9919***	**0.9836***	0.9446	0.9449	0.8893
	LR	**0.856***	**0.859***	**0.7118***	0.7459	0.7766	0.5118
	RF	**1.0***	**1.0***	**1.0***	0.995	0.995	0.991
	GNB	**0.756***	**0.7291***	**0.5307***	0.7558	0.719	0.53
	KNN	**0.9881***	**0.9883***	**0.9761***	0.9196	0.9209	0.8396
Exp1.8	**SVM**	**0.9895***	**0.9908***	**0.9786***	0.9291	0.9369	0.8567
	LR	**0.8273***	**0.8631***	**0.6566***	0.6269	0.7528	0.2628
	RF	**0.9982***	**0.9985***	**0.9964***	0.9862	0.9878	0.9722
	GNB	**0.7844***	**0.8105***	**0.5605***	0.7001	0.7874	0.4139
	KNN	**0.9895***	**0.9908***	**0.9786***	0.8821	0.9024	0.7609
Exp1.9	**SVM**	**0.9909***	**0.9904***	**0.9817***	0.9482	0.9463	0.8963
	LR	**0.837***	**0.8191***	**0.6746***	0.642	0.6084	0.2826
	RF	**0.9962***	**0.996***	**0.9925***	0.9645	0.9629	0.929
	GNB	**0.7788***	**0.7309***	**0.5701***	0.673	0.5847	0.3631
	KNN	**0.9935***	**0.9931***	**0.9869***	0.9309	0.928	0.8618
Exp1.10	**SVM**	**0.9903***	**0.9896***	**0.9805***	0.9482	0.9499	0.8965
	LR	**0.8889***	**0.8773***	**0.7779***	0.4882	0.4831	−0.0206
	RF	**0.9967***	**0.9965***	**0.9934***	0.9644	0.9658	0.9288
	GNB	**0.8265***	**0.7857***	**0.6703***	0.4985	0.4886	0.0008
	KNN	**0.9941***	**0.9937***	**0.9882***	0.9345	0.9376	0.8688

Table 3 (continued)

Exp #	ML model	R1 region performance			R2 region performance		
		Acc.	F-score	MCC	Acc.	F-score	MCC
Exp1.11	**SVM**	**0.9909***	**0.9916***	**0.9816***	0.8244	0.8163	0.652
	LR	**0.814***	**0.8369***	**0.625***	0.5763	0.6911	0.2213
	RF	**0.9956***	**0.996***	**0.9911***	0.9926	0.9926	0.9851
	GNB	**0.7588***	**0.7722***	**0.5174***	0.6481	0.7257	0.3546
	KNN	**0.9909***	**0.9916***	**0.9816***	0.8226	0.8276	0.6458
Exp1.12	**SVM**	**0.9865***	**0.9858***	**0.973***	0.7086	0.6234	0.4801
	LR	**0.7478***	**0.7056***	**0.4966***	0.5145	0.4746	0.0321
	RF	**0.9961***	**0.9959***	**0.9923***	0.9734	0.9731	0.9479
	GNB	**0.6996***	**0.61***	**0.412***	0.4785	0.3576	−0.0402
	KNN	**0.9878***	**0.9872***	**0.9757***	0.706	0.6961	0.4146

The bold asterisk represents the region with the highest score in the specific performance metric being evaluated

Table 4 Comparison of region-based performance in the four-dimensional experiments

Exp #	ML model	R1 region performance			R2 region performance		
		Acc.	F-score	MCC	Acc.	F-score	MCC
Exp2.1	**SVM**	**0.9928***	**0.9932***	**0.9856***	0.9715	0.9801	0.93
	LR	0.9645	**0.9662***	0.9288	**0.9723***	0.9307	**0.9319***
	RF	**0.9784***	**0.9795***	**0.9567***	0.9281	0.9491	0.8281
	GNB	**0.9591***	**0.9597***	**0.9207***	0.9152	0.9372	0.8234
	KNN	**0.978***	**0.9791***	**0.9558***	0.9568	0.9698	0.894
Exp2.2	**SVM**	**0.7681***	**0.7662***	**0.5367***	0.5862	0.5955	0.1727
	LR	**0.5019***	**0.5615***	**0.0011***	0.4977	0.4617	−0.0058
	RF	**0.9214***	**0.922***	**0.8429***	0.7682	0.7639	0.5366
	GNB	0.4995	**0.5741***	−0.0047	**0.5012***	0.5074	**0.0024***
	KNN	**0.9008***	**0.9008***	**0.8017***	0.7607	0.754	0.5221
Exp2.3	**SVM**	**0.9927***	**0.9919***	**0.9853***	0.9759	0.9722	0.9514
	LR	**0.9696***	**0.9664***	**0.9386***	0.959	0.9553	0.92
	RF	**0.9767***	**0.9743***	**0.953***	0.9589	0.9538	0.917
	GNB	0.9631	**0.9584***	**0.9258***	**0.9696***	0.9649	0.9386
	KNN	**0.9728***	**0.97***	**0.9452***	0.967	0.9627	0.9331
Exp2.4	**SVM**	**0.7649***	**0.7631***	**0.5298***	0.5086	0.4879	0.0168
	LR	0.4958	0.4743	−0.0088	**0.504***	**0.4857***	**0.0075***
	RF	**0.9319***	**0.9311***	**0.864***	0.7026	0.701	0.4051
	GNB	**0.5039***	**0.5114***	**0.008***	0.5005	0.3178	−0.0032
	KNN	**0.9208***	**0.9204***	**0.8416***	0.6937	0.6924	0.3873
Exp2.5	**SVM**	**0.9933***	**0.9928***	**0.9865***	0.9026	0.9187	0.8121
	LR	0.9648	0.9626	0.9295	**0.9678***	**0.9753***	**0.93***
	RF	**0.9758***	**0.9742***	**0.9514***	0.9212	0.9365	0.8365
	GNB	**0.9608***	**0.9565***	**0.9229***	0.9413	0.9521	0.8833
	KNN	**0.9743***	**0.9726***	**0.9484***	0.956	0.966	0.9039
Exp2.6	**SVM**	**0.7269***	**0.7179***	**0.4532***	0.502	0.5272	0.0037
	LR	**0.5045***	**0.4402***	0.0032	0.5021	0.1798	**0.0108***
	RF	**0.9189***	**0.9158***	**0.8376***	0.6035	0.6222	0.2077
	GNB	0.4896	0.4341	−0.026	**0.5035***	**0.5227***	**0.0068***
	KNN	**0.9059***	**0.9033***	**0.8116***	0.6201	0.6452	0.242

The bold asterisk represents the region with the highest score in the specific performance metric being evaluated

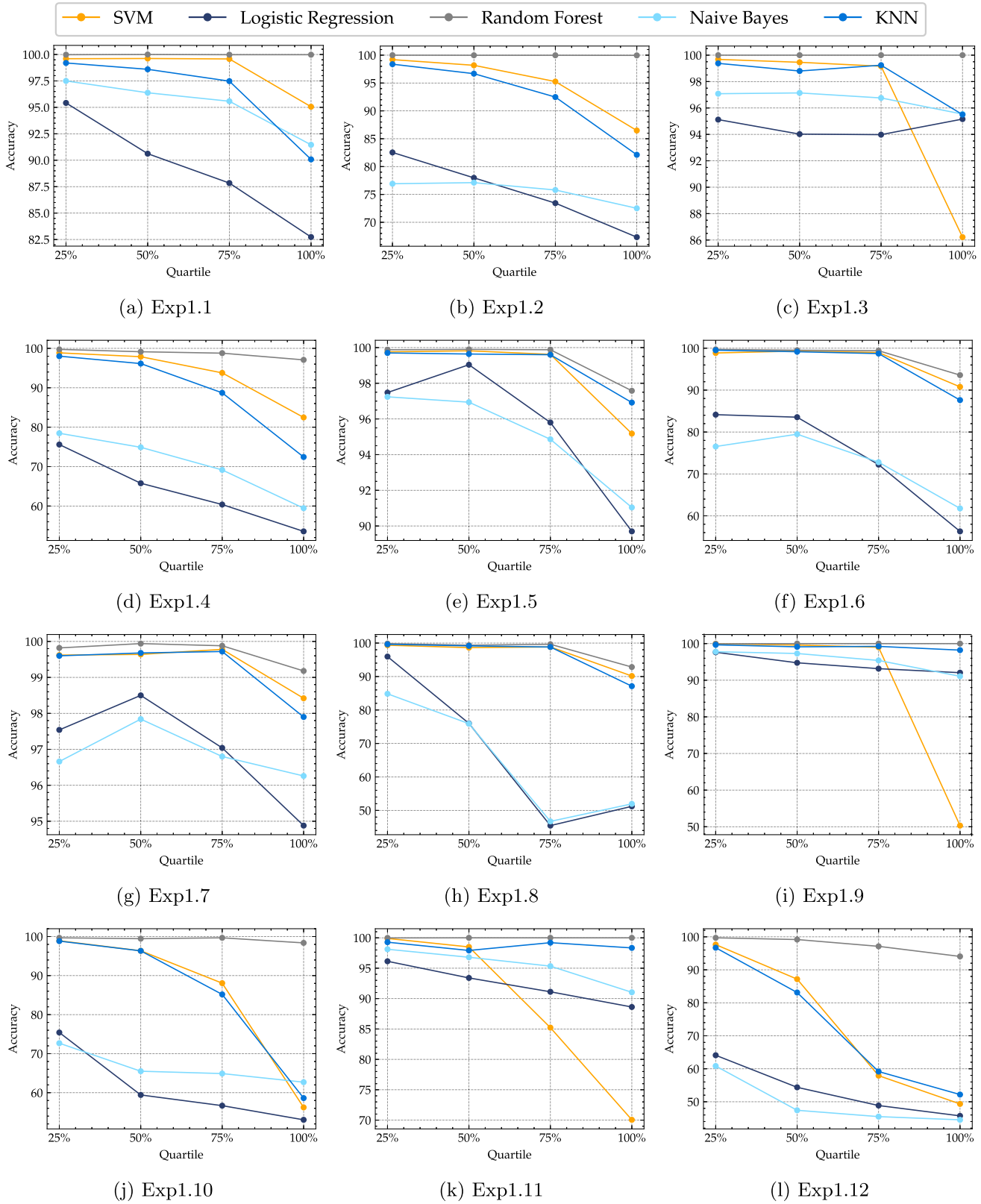


Fig. 8 Accuracy of machine learning models for each quartile of test-to-training density ratios in the two-dimensional experiments

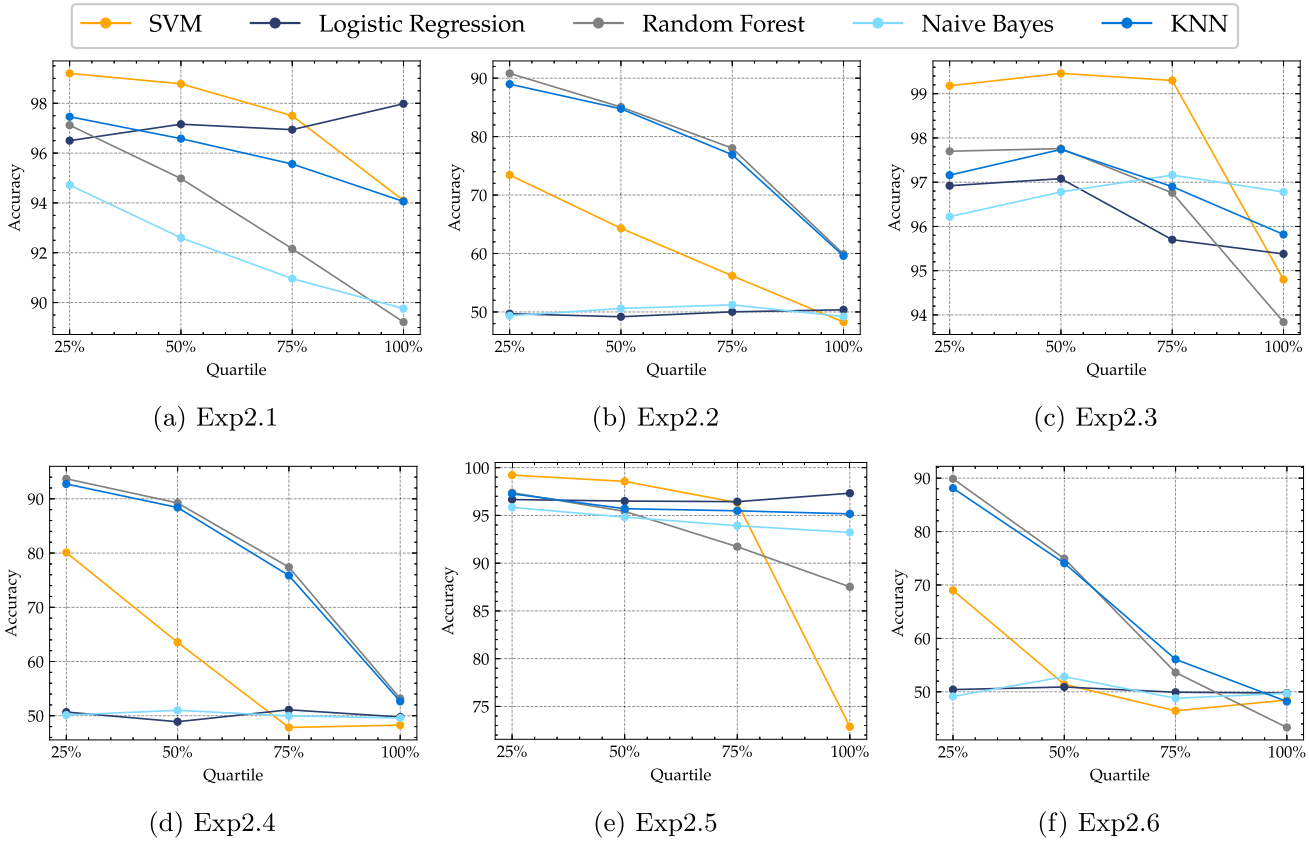


Fig. 9 Accuracy of machine learning models for each quartile of test-to-training density ratios in the four-dimensional experiments

Appendix A Summary of experimental settings

This Appendix provides a summary of the parameter settings and specifications used in the two-dimensional and four-dimensional experiments conducted in this paper. Tables 5 and 6 list the key parameters and their respective values for the two-dimensional and four-dimensional experiments, respectively.

Appendix B Summary of quartile values for test-to-training density ratios

This appendix section includes summary tables of quartile values for test-to-training density ratios in the two-dimensional and four-dimensional experiments. Table 7 shows the quartile values for the two-dimensional experiments, while Table 8 shows the quartile values for the four-dimensional experiments. The reported values include the minimum value (0), the 25th percentile (25), the median value (50), the 75th percentile (75) and the maximum value (100). These quartile values are useful in assessing the spread of the test-to-training density ratios.

Table 5 Summary of the two-dimensional experimental settings. $F_1 : \frac{1}{2}(1 + \tanh(\min(0, x_1) + 4x_2))$, $F_2 : \frac{1}{2}(1 + \sin(\min(0, x_1) + 2x_2))$

Experiment	Transformation type	Test data distribution	Classification function
Exp1.1	One-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$	F_1
Exp1.2	One-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$	F_2
Exp1.3	Two-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$	F_1
Exp1.4	Two-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$	F_2
Exp1.5	One-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}4 & 0\\0 & 1\end{bmatrix}\right)$	F_1
Exp1.6	One-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}4 & 0\\0 & 1\end{bmatrix}\right)$	F_2
Exp1.7	Two-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 2\end{bmatrix}\right)$	F_1
Exp1.8	Two-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 2\end{bmatrix}\right)$	F_2
Exp1.9	Translation and scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 2\end{bmatrix}\right)$	F_1
Exp1.10	Translation and scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 2\end{bmatrix}\right)$	F_2
Exp1.11	Translation, scaling, rotation	$N\left(\boldsymbol{x}; \begin{bmatrix}4\\-1\end{bmatrix}, \begin{bmatrix}3.5 & 0.5\\3.5 & 0.5\end{bmatrix}\right)$	F_1
Exp1.12	Translation, scaling, rotation	$N\left(\boldsymbol{x}; \begin{bmatrix}4\\-1\end{bmatrix}, \begin{bmatrix}3.5 & 0.5\\0.5 & 3.5\end{bmatrix}\right)$	F_2

Table 6 Summary of the four-dimensional experimental settings. $F_3 : \frac{1}{2}(1 + \tanh(\min(0, x_1) - x_2 + 2x_3 + 2x_4))$, $F_4 : \frac{1}{2}(1 + \sin(\min(0, x_1) + 4x_2 - 3x_3 + 2x_4))$

Experiment	Transformation type	Test data distribution	Classification function
Exp2.1	Two-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\-2\\-1\\1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}\right)$	F_3
Exp2.2	Two-axis translation	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\-2\\-1\\1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}\right)$	F_4
Exp2.3	Two-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}3 & 0 & 0 & 0\\0 & 2 & 0 & 0\\0 & 0 & 2 & 0\\0 & 0 & 0 & 3\end{bmatrix}\right)$	F_3
Exp2.4	Two-axis scaling	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}3 & 0 & 0 & 0\\0 & 2 & 0 & 0\\0 & 0 & 2 & 0\\0 & 0 & 0 & 3\end{bmatrix}\right)$	F_4
Exp2.5	Translation, scaling, rotation	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\-2\\-1\\1\end{bmatrix}, \begin{bmatrix}3 & 0 & 0 & 0\\0 & 2 & 0 & 0\\0 & 0 & 2 & 0\\0 & 0 & 0 & 3\end{bmatrix}\right)$	F_3
Exp2.6	Translation, scaling, rotation	$N\left(\boldsymbol{x}; \begin{bmatrix}0\\-2\\-1\\1\end{bmatrix}, \begin{bmatrix}2.5 & 0.5 & 0 & 0\\0.5 & 2.5 & 0 & 0\\0 & 0 & 2 & 0\\0 & 0 & 0 & 3\end{bmatrix}\right)$	F_4

Table 7 Summary of quartile values for test-to-training density ratios in the two-dimensional experiments

Experiment	0	25	50	75%	100%
Exp1.1	7.893e−04	3.550e−01	1.002e+00	1.126e+02	4.159e+04
Exp1.2	7.798e−04	3.685e−01	1.003e+00	1.187e+02	4.272e+04
Exp1.3	8.932e−04	3.779e−01	1.002e+00	1.971e+02	1.046e+05
Exp1.4	6.042e−04	3.587e−01	1.001e+00	1.876e+02	9.802e+04
Exp1.5	5.000e−01	5.852e−01	1.001e+00	3.749e+00	1.971e+03
Exp1.6	5.000e−01	5.811e−01	1.002e+00	3.698e+00	1.670e+03
Exp1.7	4.083e−01	5.922e−01	1.003e+00	2.781e+00	1.910e+02
Exp1.8	4.083e−01	5.961e−01	1.004e+00	2.821e+00	2.043e+02
Exp1.9	2.611e−02	2.942e−01	1.002e+00	2.958e+02	4.480e+08
Exp1.10	2.618e−02	3.024e−01	1.001e+00	2.838e+02	4.531e+08
Exp1.11	6.460e−03	2.264e−01	1.005e+00	1.150e+04	6.803e+12
Exp1.12	6.009e−03	2.405e−01	1.005e+00	1.204e+04	5.318e+12

Table 8 Summary of quartile values for test-to-training density ratios in the four-dimensional experiments

Experiment	0%	25%	50%	75%	100%
Exp2.1	1.789e−03	4.076e−01	1.001e+00	2.818e+01	3.185e+03
Exp2.2	1.333e−03	4.001e−01	1.002e+00	2.863e+01	3.180e+03
Exp2.3	1.683e−01	4.952e−01	1.002e+00	5.169e+00	1.832e+03
Exp2.4	1.682e−01	4.875e−01	1.002e+00	5.284e+00	1.408e+03
Exp2.5	2.087e−02	3.180e−01	1.001e+00	7.914e+01	4.845e+06
Exp2.6	1.875e−02	3.291e−01	1.001e+00	8.119e+01	5.672e+06

Acknowledgements This work has been funded by the Knowledge Foundation of Sweden (KKS) through the Synergy Project AIDA - A Holistic AI-driven Networking and Processing Framework for Industrial IoT (Rek:20200067).

Funding Open access funding provided by Karlstad University.

Data availibility The datasets generated during and/or analyzed during the current study are not publicly available currently. However, the authors are willing to provide the data per request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. L'heureux A, Grolinger K, Elyamany HF, Capretz MA (2017) Machine learning with big data, challenges and approaches. IEEE Access 5:7776–7797
2. Witten IH, Frank E, Hall MA, Pal CJ (2017) Probabilistic methods. In: Witten IH, Frank E, Hall MA, Pal CJ (eds.) Data mining (Fourth Edition), Fourth edition edn., pp 335–416. Morgan Kaufmann, San Francisco, CA, USA
3. Brownlee J (2019) Probability for machine learning: discover how to harness uncertainty with Python
4. Gama Ja, Žliobaitundefined I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4)
5. Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2019) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363
6. Moreno-Torres JG, Raeder T, Alaiz-Rodríguez R, Chawla NV, Herrera F (2012) A unifying view on dataset shift in classification. Pattern Recogn 45(1):521–530
7. Raza H, Prasad G, Li Y (2015) EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. Pattern Recogn 48(3):659–669
8. Bayram F, Ahmed BS, Kassler A (2022) From concept drift to model degradation: an overview on performance-aware drift detectors. Knowl-Based Syst 245:108632
9. Shimodaira H (2000) Improving predictive inference under covariate shift by weighting the log-likelihood function. J Stat Plann Inf 90(2):227–244

10. Sugiyama M, Kawanabe M (2012) Machine learning in non-stationary environments: introduction to covariate shift adaptation. The MIT Press, Cambridge
11. Tsymbal A (2004) The problem of concept drift: definitions and related work. Computer Science Department. Trinity College, Dublin
12. Tasche D (2017) Fisher consistency for prior probability shift. J Mach Learn Res 18(1):3338–3369
13. Li F, Lam H, Prusty S (2020) Robust importance weighting for covariate shift. In: International conference on artificial intelligence and statistics, pp 352–362. PMLR
14. Subbaswamy A, Saria S (2020) From development to deployment: dataset shift, causality, and shift-stable models in health ai. Biostatistics 21(2):345–352
15. Schneider S, Rusak E, Eck L, Bringmann O, Brendel W, Bethge M (2020) Improving robustness against common corruptions by covariate shift adaptation. Adv Neural Inf Process Syst 33:11539–11551
16. Duchi JC, Hashimoto T, Namkoong, H (2019) Distributionally robust losses against mixture covariate shifts. Under Rev 2
17. Fei G, Liu B (2015) Social media text classification under negative covariate shift. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 2347–2356
18. He H, Zha S, Wang H (2019) Unlearn dataset bias in natural language inference by fitting the residual. arXiv preprint arXiv:1908.10763
19. Wiemann PF, Klein N, Kneib T (2022) Correcting for sample selection bias in Bayesian distributional regression models. Comput Stat Data Anal 168:107382
20. Tsuboi Y, Kashima H, Hido S, Bickel S, Sugiyama M (2009) Direct density ratio estimation for large-scale covariate shift adaptation. J Inf Process 17:138–155
21. Sugiyama M, Krauledat M, Müller K-R (2007) Covariate shift adaptation by importance weighted cross validation. J Mach Learn Res 8(5)
22. Subbaswamy A, Adams R, Saria S (2021) Evaluating model robustness and stability to dataset shift. In: International conference on artificial intelligence and statistics, pp 2611–2619. PMLR
23. Liu H, Wu Y, Cao Y, Lv W, Han H, Li Z, Chang J (2020) Well logging based lithology identification model establishment under data drift: a transfer learning method. Sensors 20(13):3643
24. Sakai T, Shimizu N (2019) Covariate shift adaptation on learning from positive and unlabeled data. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 4838–4845
25. Quionero-Candela J, Sugiyama M, Schwaighofer A, Lawrence ND (2009) Dataset shift in machine learning. The MIT Press, Cambridge
26. Huang J, Gretton A, Borgwardt K, Schölkopf B, Smola A (2006) Correcting sample selection bias by unlabeled data. Adv Neural Inf Process Syst 19
27. Sugiyama M, Nakajima S, Kashima H, Buenau P, Kawanabe M (2007) Direct importance estimation with model selection and its application to covariate shift adaptation. Adv Neural Inf Process Syst 20
28. Kanamori T, Hido S, Sugiyama M (2009) A least-squares approach to direct importance estimation. J Mach Learn Res 10:1391–1445
29. Sugiyama M, Suzuki T, Nakajima S, Kashima H, von Bünau P, Kawanabe M (2008) Direct importance estimation for covariate shift adaptation. Ann Inst Stat Math 60(4):699–746
30. Chapaneri SV, Jayaswal DJ (2019) Covariate shift adaptation for structured regression with Frank–Wolfe algorithms. IEEE Access 7:73804–73818
31. Alaiz-Rodríguez R, Japkowicz N (2008) Assessing the impact of changing environments on classifier performance. In: Conference of the Canadian society for computational studies of intelligence, pp 13–24. Springer
32. Abbasian H, Drummond C, Japkowicz N, Matwin S (2010) Robustness of classifiers to changing environments. In: Canadian conference on artificial intelligence, pp 232–243. Springer
33. Chen M, Goel K, Sohoni NS, Poms F, Fatahalian K, Ré C (2021) Mandoline: Model evaluation under distribution shift. In: International conference on machine learning, pp 1617–1629. PMLR
34. Sagawa S, Koh PW, Hashimoto TB, Liang, P (2019) Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. arXiv preprint arXiv:1911.08731
35. Garg S, Balakrishnan S, Lipton ZC, Neyshabur B, Sedghi H (2022) Leveraging unlabeled data to predict out-of-distribution performance. In: ICLR
36. Guillory D, Shankar V, Ebrahimi S, Darrell T, Schmidt L (2021) Predicting with confidence on unseen distributions. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1134–1144
37. Deng W, Zheng L (2021) Are labels always necessary for classifier accuracy evaluation? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 15069–15078
38. Recht B, Roelofs R, Schmidt L, Shankar V (2019) Do imagenet classifiers generalize to imagenet? In: International conference on machine learning, pp 5389–5400. PMLR
39. Taori R, Dave A, Shankar V, Carlini N, Recht B, Schmidt L (2020) Measuring robustness to natural distribution shifts in image classification. Adv Neural Inf Process Syst 33:18583–18599
40. Miller JP, Taori R, Raghunathan A, Sagawa S, Koh PW, Shankar V, Liang P, Carmon Y, Schmidt L (2021) Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In: International conference on machine learning, pp 7721–7735. PMLR
41. Yadav C, Bottou L (2019) Cold case: The lost mnist digits. Adv Neural Inf Process Syst 32
42. Miller J, Krauth K, Recht B, Schmidt L (2020) The effect of natural distribution shift on question answering models. In: International conference on machine learning, pp 6905–6916. PMLR
43. Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250
44. Sugiyama M, Krauledat M, Müller K-R (2007) Covariate shift adaptation by importance weighted cross validation. J Mach Learn Res 8(5):1–21
45. Hachiya H, Sugiyama M, Ueda N (2012) Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. Neurocomputing 80:93–101
46. Almeida PR, Oliveira LS, Britto AS Jr, Sabourin R (2018) Adapting dynamic classifier selection for concept drift. Expert Syst Appl 104:67–85
47. Khamassi I, Mouchaweh MS, Hammami M, Ghédira K (2018) Discussion and review on evolving data streams and concept drift adapting. Evol Syst 9:1–23
48. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
49. Hastie T, Tibshirani R, Friedman JH, Friedman JH (2009) The elements of statistical learning: data mining, inference, and prediction, vol 2. Springer, New York
50. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
51. Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning, vol 4. Springer, New York

52. Hand DJ (2007) Principles of data mining. Drug Saf 30(7):621–622
53. Ovadia Y, Fertig E, Ren J, Nado Z, Sculley D, Nowozin S, Dillon J, Lakshminarayanan B, Snoek J (2019) Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. Adv Neural Inf Process Syst 32
54. Amodei D, Olah C, Steinhardt J, Christiano P, Schulman J, Mané D (2016) Concrete problems in AI safety. arXiv preprint arXiv:1606.06565
55. Lesch S, Kleinbauer T, Alexandersson J (2005) A new metric for the evaluation of dialog act classification. In: Proceedings of the 9th workshop on the semantics and pragmatics of dialogue (SEMDIAL: DIALOR), Nancy, France, pp 143–6. Citeseer
56. Rabanser S, Günnemann S, Lipton Z (2019) Failing loudly: an empirical study of methods for detecting dataset shift. Adv Neural Inf Process Syst 32
57. Clark C, Yatskar M, Zettlemoyer L (2019) Don't take the easy way out: ensemble based methods for avoiding known dataset biases. arXiv preprint arXiv:1909.03683
58. Shafieezadeh Abadeh S, Mohajerin Esfahani PM, Kuhn D (2015) Distributionally robust logistic regression. Adv Neural Inf Process Syst 28
59. Atashpaz-Gargari E, Sima C, Braga-Neto UM, Dougherty ER (2013) Relationship between the accuracy of classifier error estimation and complexity of decision boundary. Pattern Recogn 46(5):1315–1322
60. Cortes C, Mohri M (2014) Domain adaptation and sample bias correction theory and algorithm for regression. Theoret Comput Sci 519:103–126
61. Micevska S, Awad A, Sakr S (2021) SDDM: an interpretable statistical concept drift detection method for data streams. J Intell Inf Syst 56(3):459–484
62. Lima M, A Fagundes R.A.d (2021) A comparative study on concept drift detectors for regression. In: Brazilian conference on intelligent systems, pp 390–405. Springer
63. Cano A, Krawczyk B (2020) Kappa updated ensemble for drifting data stream mining. Mach Learn 109:175–218