UniTs - University of Trieste

Distribution Shift Report

Authors:

Andrea Spinelli, Giacomo Amerio, Giovanni Lucarelli, Tommaso Piscitelli

January 19, 2025

github.com/giovanni-lucarelli/distribution-shift

Problem Statement

This project focuses on evaluating the effects of simple covariate shifts in the input distribution on the performance of various robust models in the context of a synthetic binary classification task. Specifically, it investigates the extent of performance degradation caused by different types of covariate shifts and explores potential strategies to mitigate these challenges.

Key questions addressed in this study include:

- How do different types of covariate shifts affect the performance of robust models?
- Are certain models inherently more robust to simple covariate shifts?
- What strategies can be employed to improve model performance following such shifts?

Contents

1 Introduction

1.1 Dataset Shift

In the field of machine learning and predictive modeling, it is often assumed that data distributions remain static, meaning they do not change between the training and deployment phases of the model. However, in practice, this assumption is rarely satisfied: data distributions can undergo significant changes between the training and testing scenarios.

This phenomenon is known as "**dataset shift**"[1] and is closely related to another field of study, referred to by various terms such as "**transfer learning**" or "**inductive transfer**". Transfer learning addresses the problem of how information can be drawn from a number of only partially related training scenarios and used to provide better predictions in one of those scenarios compared to using only that specific scenario. Therefore, dataset shift represents a more specific case: it deals with relating information in, typically, two closely related environments to improve prediction in one given the dataset in the other. Given this issue, it is crucial to develop an understanding of the suitability of particular models under such changing conditions, and it is necessary to consider whether a different predictive model should be employed.

Among the various forms of dataset shift, covariate shift, studied and described by Shimodaira[2], is one of the most extensively researched. It encompasses situations where the distribution of the covariates, $P(X)$ changes, while the conditional relationship $P(Y \mid X)$, representing the relationship between the covariates X and the target Y, remains unchanged. In this case, the typical values of the covariates observed during testing differ from those observed during training.

1.1.1 Most common causes of dataset shift

The two most common and well-studied causes of dataset shift are:

1. Sample selection bias
2. Non-stationary environments

Sample selection bias occurs when there is a discrepancy in the data distribution due to the training data being obtained through a biased method, and therefore not reliably representing the real environment in which the classifier will be used (the test set). This bias is not necessarily a flaw of the algorithm or data management process but a systematic defect in the way the data is collected or labeled, causing a non-uniform selection of training examples from the population. This often leads to bias being introduced during training. Dataset shift resulting from sample selection bias is particularly relevant when dealing with imbalanced classification problems, as, in highly imbalanced domains, the minority class is especially sensitive to misclassification errors due to its typically low number of samples.

In real-world applications, data often is not stationary (in time or space). One of the most relevant **non-stationary** scenarios involves adversarial classification problems, such as spam filtering and network intrusion detection.

1.2 Simple Covariate Shift

The most fundamental form of dataset shift, known as *covariate shift*, can be formally defined as follows. Consider an input variable X and a response variable Y, where $X \to Y$ represents the relationship between the two. Let P_{tra} denote the probability distribution of the training data and P_{tst} denote the probability distribution of the test data. A covariate shift occurs when:

$$P_{\text{tra}}(Y \mid X) = P_{\text{tst}}(Y \mid X) \quad \text{but} \quad P_{\text{tra}}(X) \neq P_{\text{tst}}(X) .$$

In other words, the conditional distribution of Y given X remains unchanged across training and test datasets, while the marginal distribution of X differs. Figure 1.1 shows an example of different distribution between training data and test data, creating a division between the two datasets.

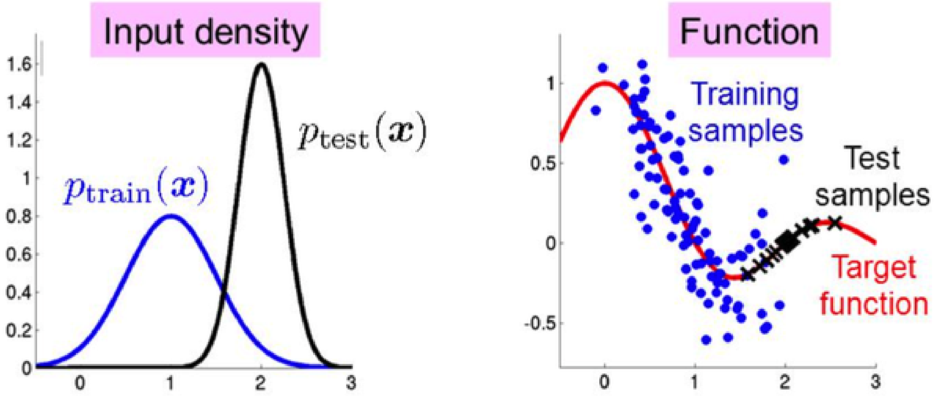

Figure 1.1. Example of covariate shift.

This phenomenon can affect a wide range of machine learning models, regardless of the task they are designed to perform. It is commonly encountered in scenarios where models classify data or predict trends based on input features. This issue is particularly relevant in diverse machine learning applications, including but not limited to:

1. Image categorization and facial recognition systems
2. Speech recognition and translation software
3. Diagnostic and screening tools in healthcare

For example, consider a model designed to distinguish between cats and dogs. Our training data might consist of images like those shown in Figure 1.2 (top). At test time, we are asked to classify the images from the test set as the one in the bottom of the same figure. Once deployed, the model will not accurately distinguish between cats and dogs because the feature distribution will differ. Model may achieve a high degree of accuracy on a labeled training dataset, identifying and classifying the object in an image. However, when deployed with real-time data, changes in the input distribution can significantly impact the model's accuracy.

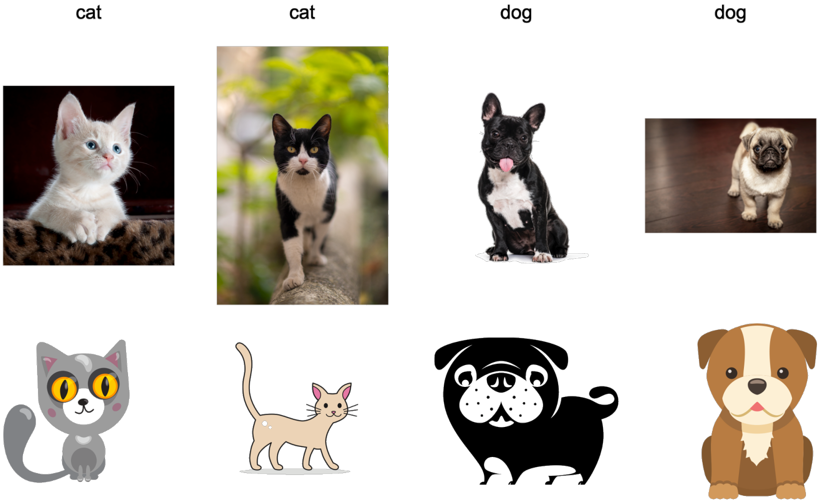

Figure 1.2. Training (top) and testing (bottom) data for distinguishing cats and dogs.

The same can accure in facial recognition: the training data might not include subjects from specific ethnicities or age groups. When the model is deployed in a real-world environment, subjects that do not align with the training data may exhibit an unrecognizable feature distribution. Another cause of covariate shift could be variations in environmental conditions, such as lighting. For instance, an image categorization model trained under specific lighting conditions may perform poorly when deployed in an operational setting with different lighting.

Covariate shift, also known as covariate drift, is a common challenge in machine learning. Supervised learning models are typically trained on labeled data, which is prepared and curated by data scientists to ensure high quality through outlier detection and analysis. However, this level of control is not feasible in operational environments, as input data becomes unpredictable once the model is deployed. Consequently, the training data often differs in distribution from real-world input data, both in availability and characteristics.

This mismatch can negatively impact the model's accuracy. Trained algorithms may fail to recognize features from a different data distribution, leading to reduced performance or even complete ineffectiveness, as illustrated in Figure 1.3. This highlights a critical issue in machine learning: a model that performs well on training data may not retain its accuracy post-deployment.

The primary objective is to assess the extent of the covariate shift, implement mitigation strategies, and enhance the model's accuracy. Addressing covariate shift also reveals the model's generalization ability: its capacity to apply learned features from training data to unseen data. Poor generalization often stems from overfitting, where the model becomes overly tailored to the training data, making it ineffective for inputs with differing distributions.

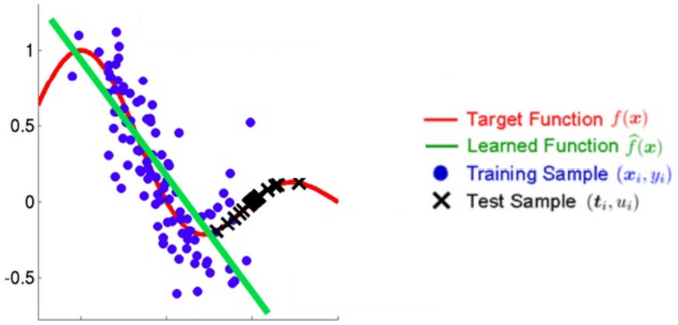

Figure 1.3. Example of inaccurate model.

2
Data Generation

To evaluate the performance of the proposed models under varying degrees of covariate distribution shift, we simulated a training dataset and multiple test datasets, each exhibiting a distinct degree of shift. This setup enables a comprehensive assessment of the models' generalization capabilities.

2.1 Training Data Generation

The training dataset consists of three features, denoted as X_1, X_2, and X_3, and a binary target variable Y. The features were generated from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where each element of the mean vector $\boldsymbol{\mu i}$ was sampled from a uniform distribution $\mathcal{U}_{[0,1]}$, and the elements of the covariance matrix $[\boldsymbol{\Sigma}]_{i,j}$ were sampled from $\mathcal{U}_{[-1,1]}$. To ensure the validity of the covariance matrix, after the generation, it was then transformed into a symmetric positive semi-definite matrix. The target variable Y is a binary variable with values in $\{0, 1\}$. It was generated by first constructing a second-order polynomial model with all possible interaction terms and random coefficients drawn from $\mathcal{U}_{[-1,1]}$. The output of this polynomial was then transformed using the standard logistic function to produce probabilities, which were thresholded to determine the binary values of Y.

2.2 Testing Data Generation

A fully shifted test dataset was generated from a new multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_{0.05}, \boldsymbol{\Sigma}_s)$. Here, $\boldsymbol{\mu}_{0.05}$ represents a mean vector centered at the 5th percentile of the original $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}_s$ is a covariance matrix distinct from $\boldsymbol{\Sigma}$. This shift was deliberately designed to focus on a region of the sample space with limited representation in the training data, thereby introducing a significant covariate distribution shift. The target variable Y for this dataset was generated in the same manner as in the training dataset.

To assess model performance across a continuum of distribution shifts, we created a series of datasets using statistical mixtures of the training dataset and the fully shifted dataset. Specifically, we defined ten datasets, $\mathcal{D}_p$, where $p \in \{0.0, 0.1, \ldots, 1.0\}$ represents the mixing probability. The dataset $\mathcal{D}_{0.0}$ corresponds to data generated from the same distribution as the training dataset (but with new samples), allowing for an evaluation of the models on unseen, unshifted data. Conversely, $\mathcal{D}_{1.0}$ corresponds to the fully shifted dataset. Intermediate values of p represent datasets with increasing proportions of shifted data, enabling a systematic investigation of model robustness under various degrees of covariate distribution shift.

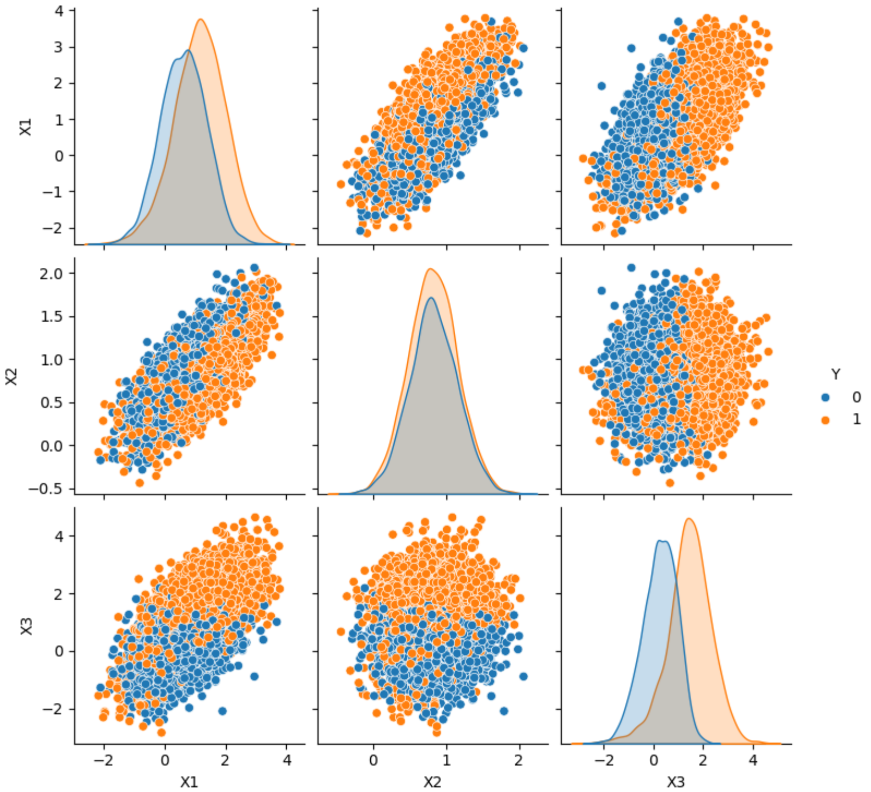

Figure 2.1. Distribution of the target variable in the training dataset.

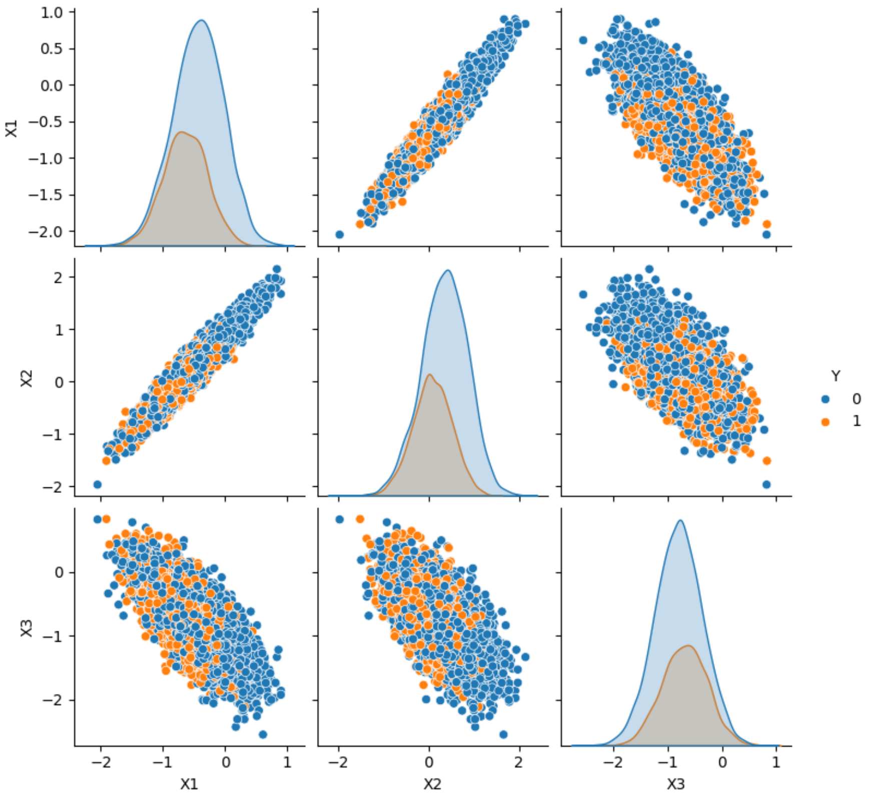

Figure 2.2. Distribution of the target variable in the fully shifted dataset.

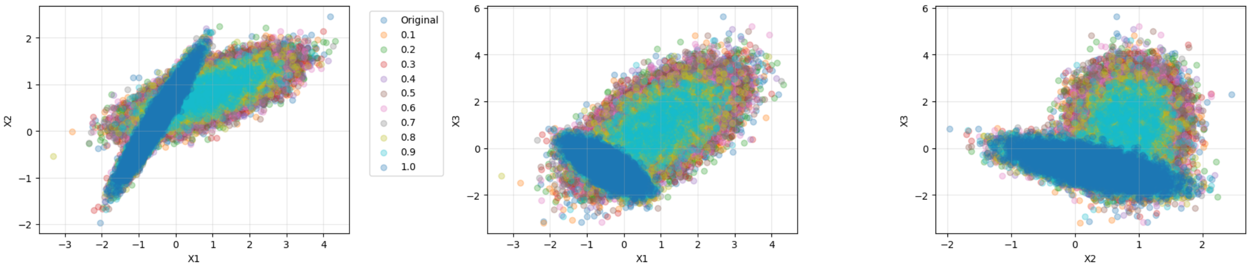

Figure 2.3. Sparseplot of the three features for all different mixing probability values of the mixtures.

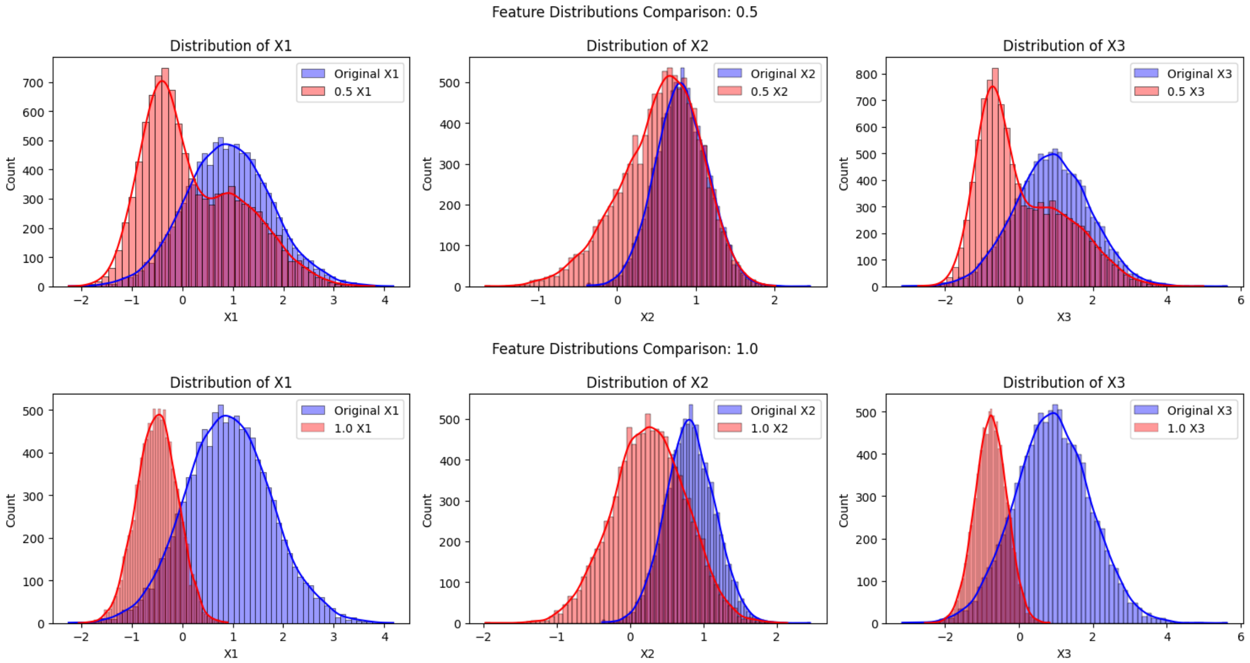

Figure 2.4. Comparison of feature distributions for the mixture dataset (red) and the training dataset (blue). Top: mixture with $p = 0.5$. Bottom: mixture with $p = 1.0$ (fully shifted dataset).

3
Performance Degradation

3.1 Evaluation Metric: AUC

We used the **Area under the Curve** (**AUC**) to evaluate model performance. The AUC measures the performance of a classification model by calculating the area under the ROC curve, which plots the true positive rate against the false positive rate. The AUC ranges from 0 to 1, where 0 indicates no predictive power and 1 indicates perfect predictive power.

Because the AUC focuses on how well the model ranks positive instances above negative ones, it is generally robust to shifts in the data distribution: if a positive and a negative example are chosen at random, the model should consistently rank the positive example higher than the negative one.

3.2 Models

As metioned in the first chapter, simple covariate shift can lead to serious degradations in model performance. In this chapter, we will analyze how different statistical learning models' performances are affected by covariate shift.

We evaluated a diverse set of models, ranging from simple linear classifiers to more sophisticated ensemble methods, to comprehensively assess their robustness under covariate shift conditions:

- **Logistic Regression**: A linear classifier that serves as a baseline model;
- **Decision Tree**: A non-linear model that creates a tree-like structure of decision rules;
- **Generalized Additive Model (GAM)**: A flexible statistical model that combines the interpretability of linear models with the ability to capture non-linear relationships;
- **Gradient Boosted Trees**: An ensemble learning method that creates sequential decision trees to iteratively correct prediction errors. Each tree focuses on the residuals from previous predictions, with optimized step sizes to minimize the overall loss function using gradient descent;
- **eXtreme Gradient Boosting (XGBoost)**: A highly optimized implementation of gradient boosting machines known for its efficiency and performance.

Each model was selected for its unique characteristics and widespread use in practical applications, providing a comprehensive view of how different learning approaches handle distribution shifts.

3.3 Results

We firstly evaluated the performance of each vanilla model on the aforementioned statistical mixtures of original and shifted test data.

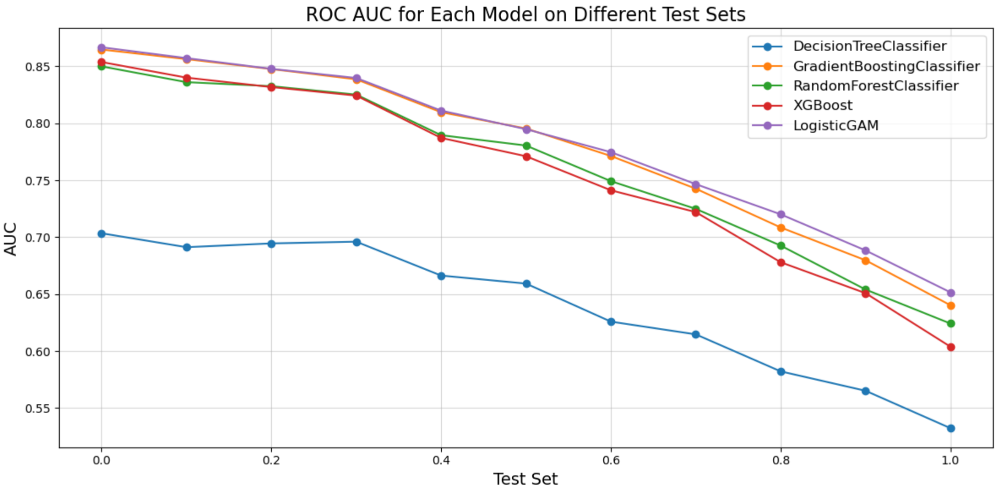

Figure 3.1. Performance comparison of vanilla models under covariate shift. The AUC scores of the models are plotted against the mixing probability p, which represents the proportion of shifted data in the test set.

As we can see from the plot, at low levels of mixed data, the models can still perform realatively well. However, as the proportion of mixed data increases, the performance of the models degrades significantly. The decision tree model is the most sensitive to covariate shift, while the GAM model is the most robust.

Given these preliminary results, we proceeded with hyperparameter optimization to potentially enhance model performance.

Fine Tuning

The best choice for hyperparameters in these models might be different depending on the dataset. In order to find the hyperparameters that best fit the data, we performed a hyperparameter tuning using the `GridSearchCV` function from the `scikit-learn` library, which performs a cross-validation (k=5) to select the best hyperparameters for each model.

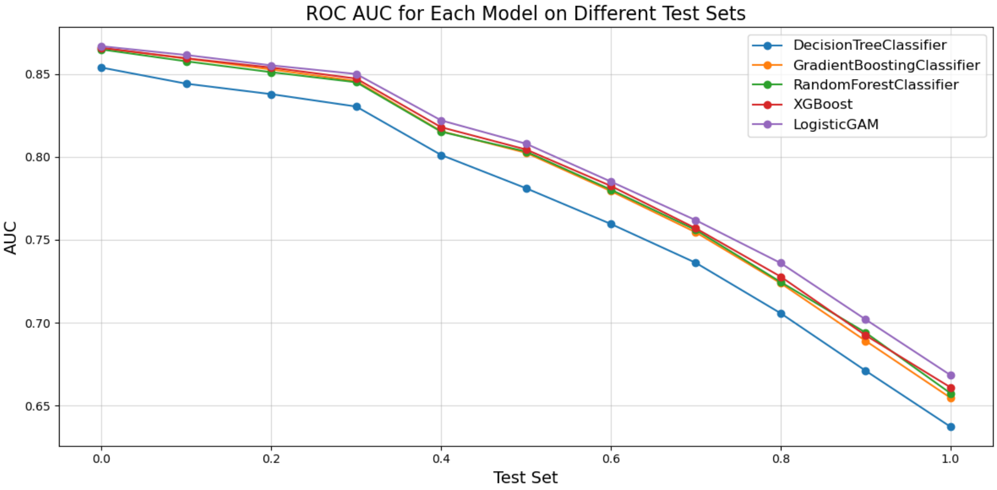

Figure 3.2. Performance comparison of fine tuned models under covariate shift.

As shown in Figure 3.2, the fine-tuned models exhibit improved performance compared to the vanilla models. The decision tree model, which was the most sensitive to covariate shift, now performs almost on par with the other models. The GAM model remains the most robust, with consistently high AUC scores across all mixing probabilities. Bear in mind that we are still working on the fine tuning.

4
Performance Enhancement

One of the most used approach to mitigate covariate shift consequences is *Reweighting*, which consists in quantify the degree of distribution shift and then apply a correction to the model [3]. Another approach is *Data Augmentation*, which consists in generating new data points from the original ones, in order to make the model more robust to the distribution shift [4]. In this chapter, we introduce two distinct strategies to improve model performance under covariate shift conditions: *Overfitting* and an innovative approach we term ***Random Walk Augmentation***.

4.1 Overfitting

The simplest approach to improving model performance is to overfit the model on the training data. The idea is that by overfitting to the training data, the model may better capture the underlying data distribution, potentially leading to improved performance on the shifted test data—particularly for data points that lie within the intersection of the training and test distributions.

In order to overfit the models, we created a custom script to perform grid search withot cross-validation. We favored the hyperparameters that achieved the highest score on the training set, while ignoring the necessary precaution to avoid overfitting, i.e. we overshot the maximum depth reachable by the trees.

Since the logistic GAM uses different methods to perform fine tuning from the other ones, we are still trying to figure out how to implement a proper overfit. The results for the other models are shown in Figure 4.1.

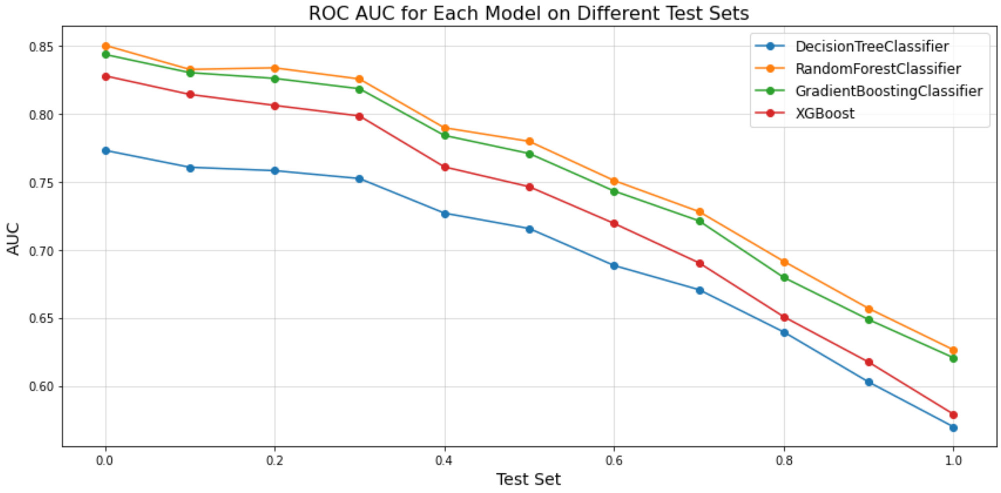

Figure 4.1. Performance comparison of overfitted models under covariate shift

As shown in Figure 4.1, we can clearly see that as we go through increasing percentage of mixed data points, the performance of the models are not on par with the ones that were fine tuned. As the plot further points out, the overfit led to a diversification in the models' AUC scores, with the decision tree model being the most affected by the overfitting process.

4.2 Random Walk Augmentation

This method is based on the idea of **Data Augmentation**. Instead of using training data as it is, we create new data applying the following transformation to the original data:

Algorithm 4.1 Custom Data Augmentation

```
Input: $Data$, $N$
Output: $Data_{\text{aug}}$

$Size \leftarrow len(Data)$
$Data_{\text{new}} \leftarrow Data$
$Data_{\text{tr}} \leftarrow$ random subset of $N\%$ of $Data$
for $x_i$ in $Data_{\text{tr}}$ do
    $x'_i \leftarrow \begin{cases} X_i + \varepsilon & \text{with probability } 0.5 \\ X_i - \varepsilon & \text{with probability } 0.5 \end{cases}$
    $y'_i \leftarrow y_i$
end for
$Data_{\text{aug}} \leftarrow Data_{\text{new}} \cup Data_{\text{tr}}$
$Data_{\text{aug}} \leftarrow Downsample(Data_{\text{aug}}, Size)$
return $Data_{\text{aug}}$
```

Interestingly, this method does not require any knolewdge of the shifted test distributions, it just performs a noising step on a variable percentage of the training data. Then it downsamples the augmented data to the original size. Despite the variation in the x'_i values, the y_i values remain the same, this leads to a **looser fit** on the training data and enhances the perfomarce on the shifted test data.

4.2.1 Experiments

We firstly evaluated this method on the same classification task as the other models, but, since we needed a better way to theoretically understand the inner processes of the training, we decided to apply it to a simple 1-dimensional regression problem.

4.2.2 Training Set

The training set consists of 10,000 data points generated with x values linearly spaced between -3 and 3, and y values generated with the following formula:

$$y = \sin(x)\exp(-x^2) + \varsigma \tag{4.1}$$

Where ς is a random noise term drawn from a normal distribution with mean 0 and standard deviation 0.1.

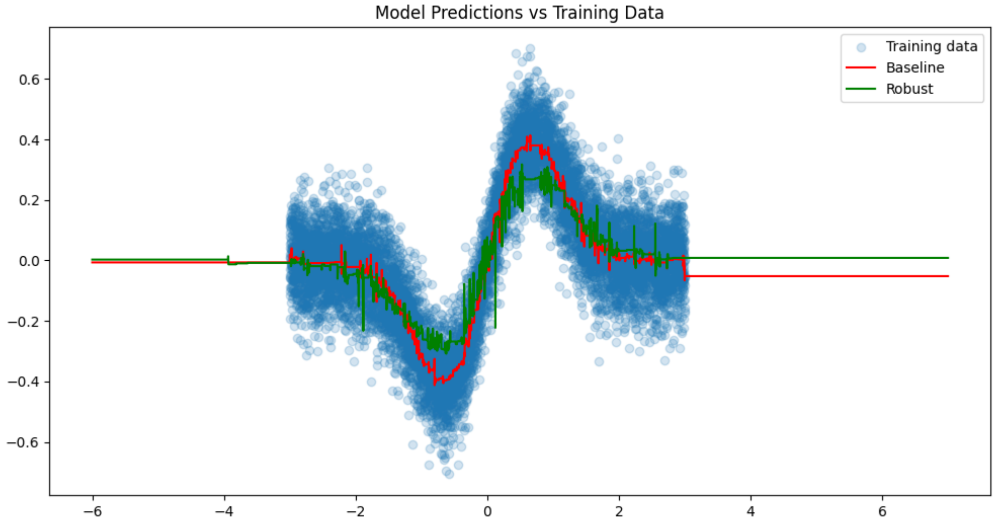

Figure 4.2. Baseline and Robust models fit on the training data To note that the robust model has a **looser fit** on the training data.

4.2.3 Test Sets

Thirty test sets are created by shifting the x values by factors ranging from 1.5 to 5.5. Each shifted test set is generated independently using the same underlying function and noise process as the training data.

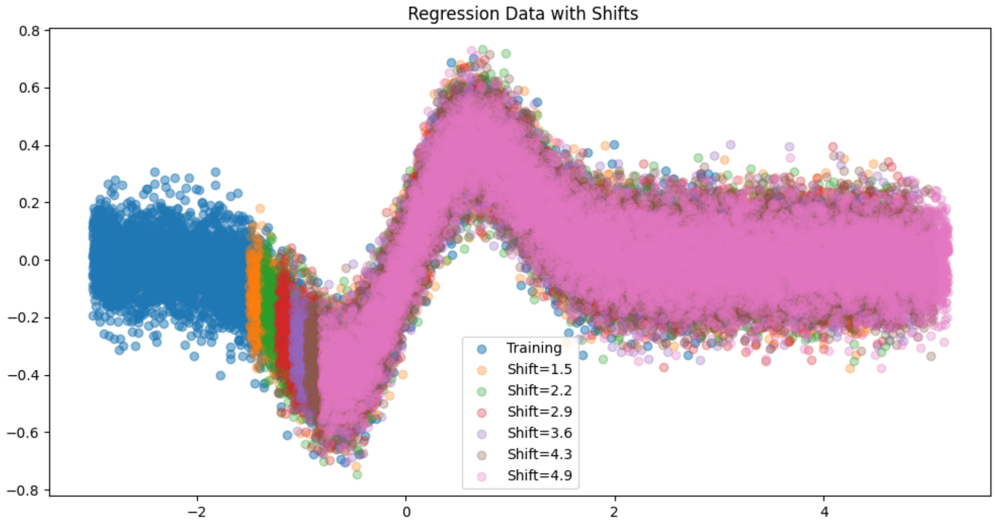

Figure 4.3. A handful of test sets depicted together with the training set

4.2.4 Models

Two types of models are trained:

- **Baseline Model**: A Gredient Boosting Regressor (GBR) is employed as a baseline model. The GBR is configured with the following hyperparameters: `n_estimators=100`, `max_depth=5`, `learning_rate=0.05`.
- **Robust Model**: The robust model has the same features as the baseline GBR but leverages the key parameters of the custom data augmentation method. The percentage of training data (`fraction_to_shift`) to be augmented is set to 40%. Meanwhile the `base_shift_factor` controls the magnitude of shifts considered in training (set to 1 in this experiment).

4.2.5 Results

Evaluation Metric

The model performance is evaluated using the Mean Squared Error (MSE). For each shifted test set, the MSE is computed for both the baseline and robust models. The improvement is defined as the relative reduction in MSE computed as :

$$\text{Improvement} = \left(\frac{\text{MSE}_{\text{baseline}} - \text{MSE}_{\text{robust}}}{\text{MSE}_{\text{baseline}}} \right) \times 100\% \tag{4.2}$$

The metric is computed for each shift, and the results are shown in the figure below.

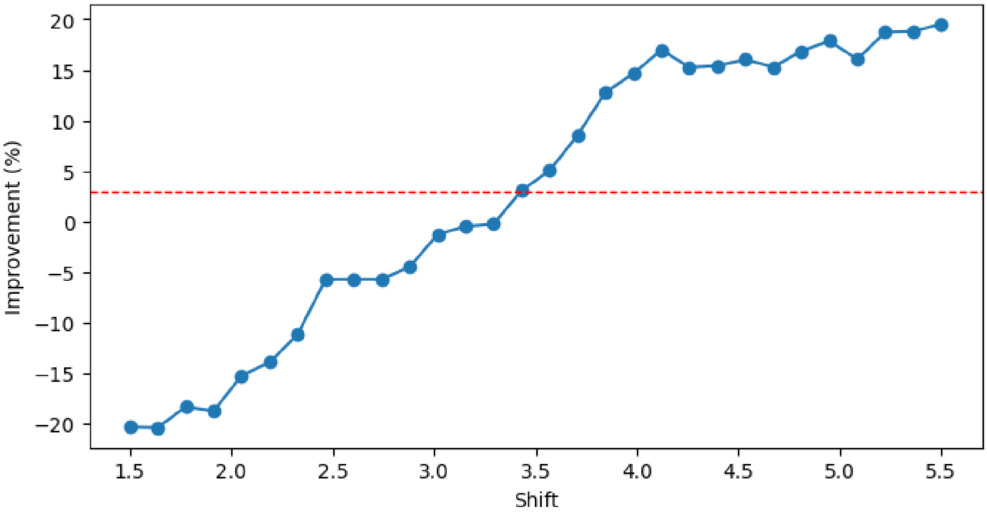

Figure 4.4. Model Improvement over Shifted Test Sets. The red dotted line is the mean improvement across all shifts.

As we can see from the plot, the robust model has worse performance then the baseline model for relatively small shifts but as the shift in data points becomes more significant, the robust model outperforms the baseline model. The mean improvement across all shifts is still positive. We believe the promising results of this methods are still to be analyzed in depth, but the preliminary results are encouraging.

Bibliography

[1] J. Quinonero-Candela et al. *Dataset Shift in Machine Learning*. Neural Information Processing series. MIT Press, 2022. ISBN: 9780262545877. URL: https://books.google.it/books?id=MBZuEAAAQBAJ.

[2] Hidetoshi Shimodaira. “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: *Journal of Statistical Planning and Inference* 90.2 (2000), pp. 227–244. ISSN: 0378-3758. DOI: https://doi.org/10.1016/S0378-3758(00)00115-4. URL: https://www.sciencedirect.com/science/article/pii/S0378375800001154.

[3] Haoran Zhang et al. *”Why did the Model Fail?”: Attributing Model Performance Changes to Distribution Shifts*. 2023. arXiv: 2210.10769 `[cs.LG]`. URL: https://arxiv.org/abs/2210.10769.

[4] Mengnan Zhao et al. *Adversarial Training: A Survey*. 2024. arXiv: 2410.15042 `[cs.LG]`. URL: https://arxiv.org/abs/2410.15042.