Distribution Shift Report

A Study on Its Effects on Statistical Models and Strategies for Mitigation

Authors:

Andrea Spinelli, Giacomo Amerio,

Giovanni Lucarelli, Tommaso Piscitelli

January 27, 2025

Purpose of the Study

This project aims to evaluate the impact of simple covariate shifts in the input distribution on the performance of robust models within the context of a synthetic binary classification task. Specifically, we investigate the degree of performance degradation induced by various types of covariate shifts and explore potential strategies to mitigate these effects.

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

<div style="text-align: right; font-size: 2em;">1</div>

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.

Among the various forms of dataset shift, covariate shift, studied and described by Shimodaira[1], 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. 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.
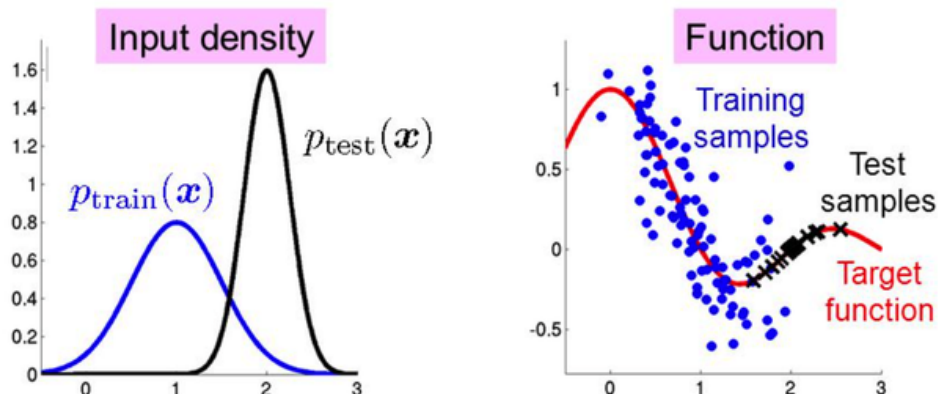

Figure 1.1. Example of covariate shift.

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 in testing scenarios, changes in the input distribution can significantly impact the model's *effectivness*.
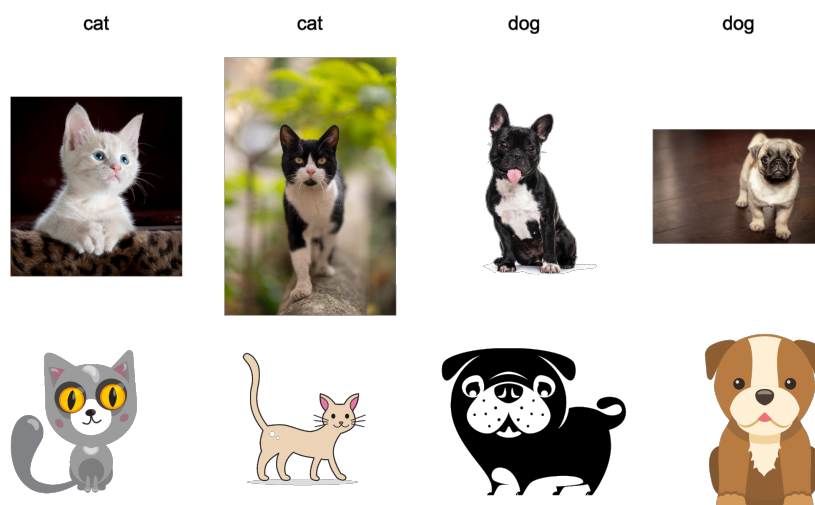

Figure 1.2. Training (top) and testing (bottom) data for distinguishing cats and dogs.

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.

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 perfomance levels post-deployment.

The primary objective of this project is to assess the extent of the covariate shift, implement mitigation strategies, and try to enhance the model's effectivness. 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.
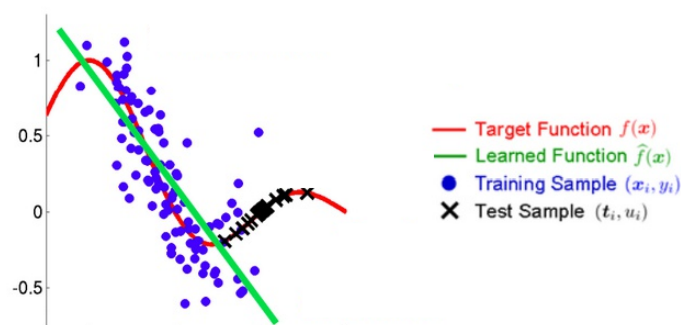

Figure 1.3. Example of inaccurate model.

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 transformed using the standard logistic function to generate probabilities, which were then fed into a Bernoulli random number generator to produce the binary values of Y (Figure 2.1).
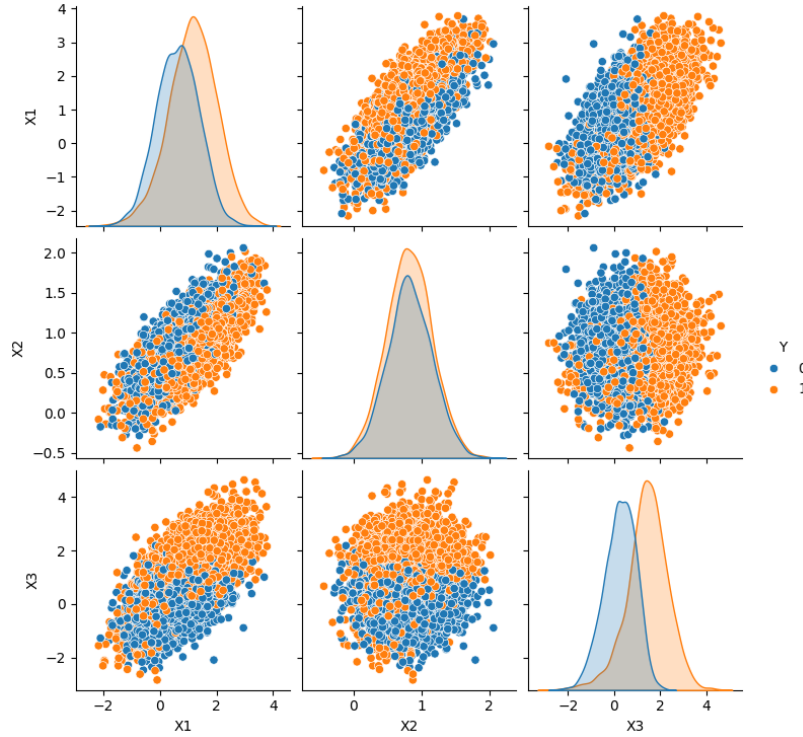

Figure 2.1. Distribution of the target variable in the training dataset.

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 each feature of the training set $\mathbf{X}_{\text{train}}$, and $\boldsymbol{\Sigma}_s$ is a new covariance matrix distinct from $\boldsymbol{\Sigma}$ and such that $[\boldsymbol{\Sigma}]_{i,j}$ sampled from $\mathcal{U}_{[-0.5, 0.5]}$. This shift was deliberately designed to focus on a smaller region of the sample space centered in a point with limited representation in the training data, thereby introducing a significant covariate shift. The target variable Y for this dataset was generated in the same manner as in the training dataset.
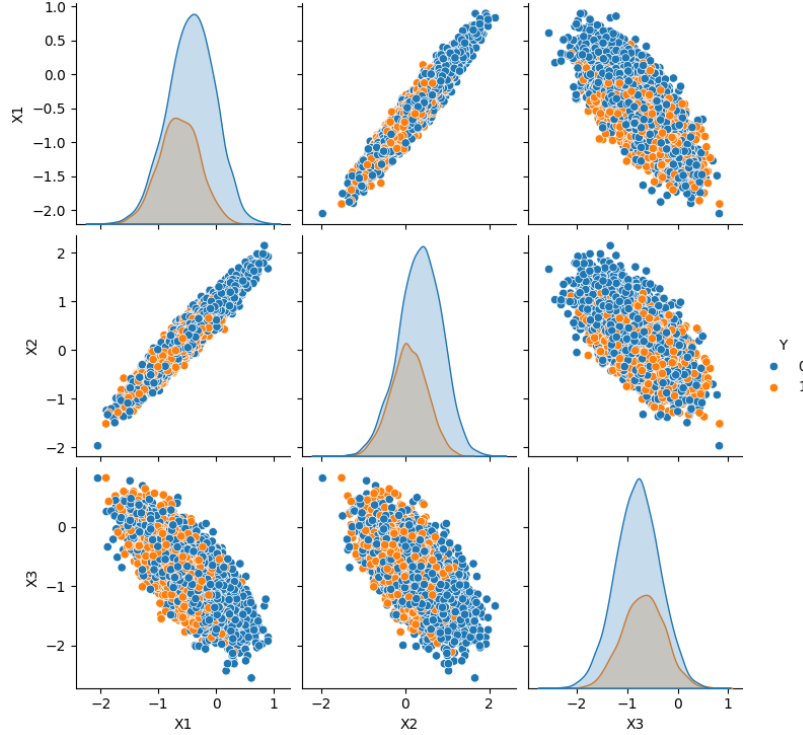

Figure 2.2. Distribution of the target variable in the fully shifted dataset.

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 $p \in \{0.0, 0.1, \dots, 1.0\}$ 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.

As illustrated in Figure 2.2, the target variable distribution in the fully shifted dataset differs substantially from that of the training dataset, exhibiting a higher proportion of instances with $Y = 0$. Notably, the **imbalance ratio** (IR) of the fully shifted dataset is 2.36, compared to 1.19 in the training dataset, and this ratio increases as the mixing probability p rises. It is important to take into account this variation in the distribution of the target variable when assessing the effectiveness of different models. The proposed approach is to use threshold-independent metrics such as the **ROC AUC**. This metric is particularly well-suited for evaluating model performance in the presence of class imbalance and varying decision thresholds.
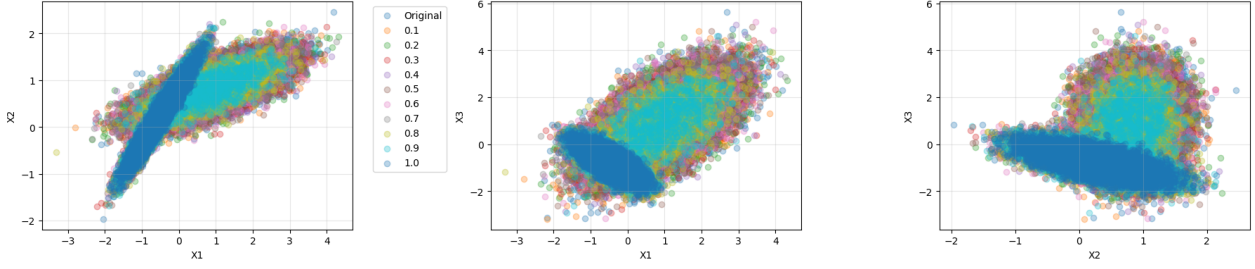
Figure 2.3. Sparseplot of the three features for all different mixing probability values of the mixtures. The full shifted dataset is the smaller one, in blue.
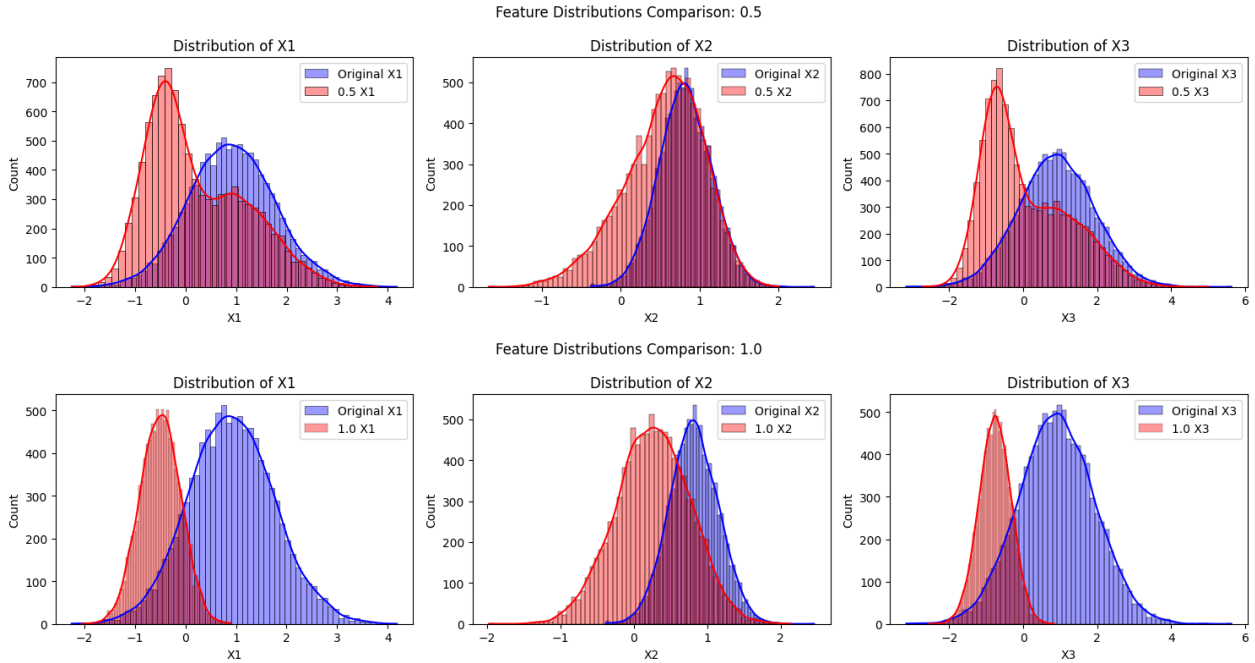

Figure 2.4. Comparison of feature distributions for the mixture dataset (red) and the training dataset (blue). Top: mixture with $p = 0.5$. Bottom: mixture with $p = 1.0$ (fully shifted dataset).

3

Performance Degradation

3.1 Evaluation Metric: ROC-AUC

We used the **Area under the Reciever Operating Characteristic Curve** (**ROC-AUC**) to evaluate model performance. The **ROC** plots the true positive rate against the false positive rate and ranges from 0 to 1, with higher values indicating better model performance.

Because the ROC-AUC focuses on how well the model ranks positive instances above negative ones, it is generally more robust to imbalanceness in data distributions rather than metrics like accuracy.

3.2 Evaluated 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 to comprehensively assess their robustness under covariate shift conditions:

- **Logistic Regression**: A generalised linear model that serves as a baseline model;

- **Random Forest**: An ensemble learning method that builds multiple decision trees and averages their predictions to improve performance;

- **Gradient Boosting**: An ensemble learning method that builds sequential weak learners (in our case decision trees) to correct prediction errors iteratively, focusing on residuals from previous predictions;

- **eXtreme Gradient Boosting (XGBoost)**: A highly optimized implementation of gradient boosting machines known for its efficiency and performance.

As we will see, these models (except the logistic regression) have different hyperparameters that can be tuned to improve performance. We used a grid search with 5-fold cross-validation to find the best hyperparameters for each model.

> **What are Grid Search and Cross Validation?**
>
> **Grid Search** is a hyperparameter optimization technique that systematically searches through a fixed grid of hyperparameters with a brute-force method that evaluates all possible combinations of hyperparameters to find the best model.
>
> **Cross-validation** is a technique used to evaluate the performance of a model by splitting the data into k subsets (folds) and training the model on $k-1$ folds while using the remaining fold for validation. This process is repeated k times, with each fold used once as a validation set.
>
> <u>Note</u>: This method only uses training data to optimize hyperparameters, reflecting real scenarios where test data is unavailable during training.

3.2.1 Logistic Regression

Logistic regression models the relationship between features and the target variable using linear combinations of the input variables, which may not capture complex nonlinear patterns in the data. Given the assumed nonlinear relationships between variables, which also includes intersactions, we decided to use as baseline model the one including all the interaction terms (`X1:X2`, `X1:X3`, `X2:X3`, and `X1:X2:X3`).

We used the `statsmodels` library to fit a logistic regression model on the training dataset and evaluated it on the test set. The table below summarizes the coefficients, significance ($P>|z|$), and confidence intervals for the fitted logistic regression baseline model.

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.9761	0.122	-8.022	0.000	-1.215	-0.738
X1	-0.6550	0.120	-5.462	0.000	-0.890	-0.420
X2	-0.2049	0.180	-1.139	0.255	-0.558	0.148
X1:X2	-0.2653	0.125	-2.118	0.034	-0.511	-0.020
X3	1.3187	0.115	11.478	0.000	1.094	1.544
X1:X3	1.3248	0.106	12.453	0.000	1.116	1.533
X2:X3	-0.3253	0.172	-1.894	0.058	-0.662	-0.011
X1:X2:X3	0.0510	0.120	0.427	0.670	-0.183	0.285

Table 3.1.   Summary of Logistic Regression coefficients.
Coefficients with statistical significance ($p < 0.05$) are hilighted in red.

As we can see from the table, for this data instance, the features X_2, $X_2 : X_3$ and $X_1 : X_2 : X_3$ are not statistically significant, as their p-values are above the 0.05 threshold. This suggests that these features may not be relevant for predicting the target variable Y, but bare in mind that this is only one instance of the data and the results may vary across different instances.

The following plots show the ROC curves and the AUC scores for the logistic regression model under different levels of covariate shift.
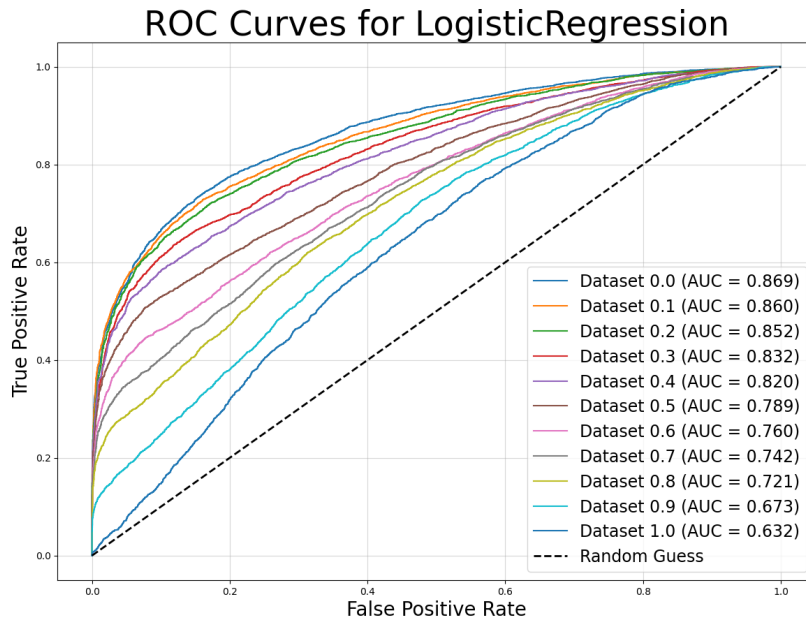

Figure 3.1. Performance of the Logistic Regression model under covariate shift.
The AUC scores of the model are plotted against the mixing probability p, which represents the proportion of shifted data in the test set.

3.2.2 Ranfom Forests

Random Forests are an ensemble learning technique that constructs multiple independent decision trees and combines their outputs through averaging to enhance predictive accuracy.

We used the `RandomForestClassifier` from the `scikit-learn` library. This model presents several hyperparameters that can be tuned to improve performance. To find the best hyperparameters we used the `scikit-learn` `GridSearchCV` method with 5-fold cross-validation.

The best hyperparameters found are shown in the following table:

Parameter	Value
n_estimators	125
criterion	gini
max_depth	6
min_samples_leaf	1
min_samples_split	5
bootstrap	True

Table 3.2. Best hyperparameters found for the Random Forest model using 5-fold cross validation

We also used a fixed `random_state=0` to ensure reproducibility of results.

The following plot shows the ROC curves and the AUC scores for the Random Forest model under different levels of covariate shift.
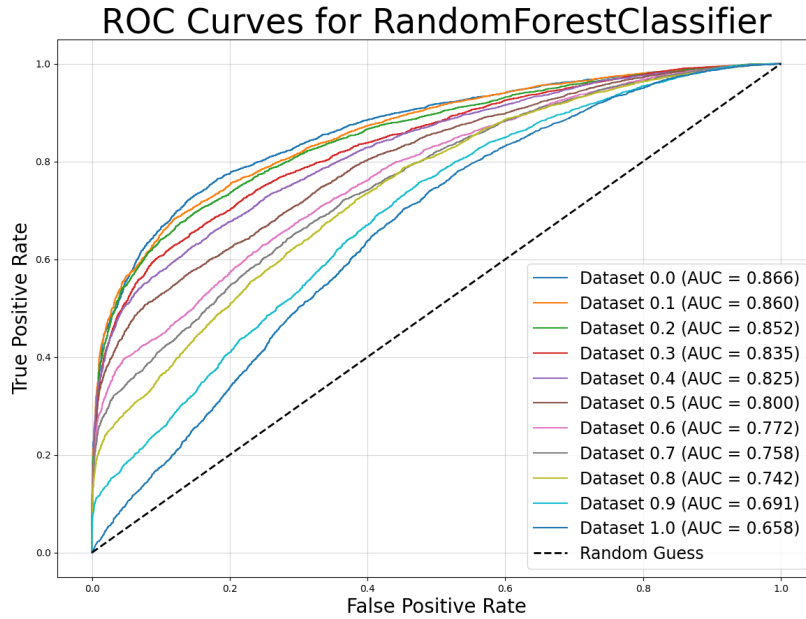

Figure 3.2. Performance of the Random Forest model under covariate shift.
The AUC scores of the model are plotted against the mixing probability p, which represents the proportion of shifted data in the test set.

3.2.3 Gradient Boosting

Gradient Boosting builds sequential weak learners (in our case decision trees) to iteratively correct prediction errors by focusing on residuals from previous predictions. Unlike Random Forests where trees are built independently, each tree in Gradient Boosting learns from the mistakes of previous trees.

We used `scikit-learn`'s `GradientBoostingClassifier`. Also in this case we used `GridSearchCV` method from `scikit-learn` library with a 5-fold cross-validation to find the following optimized hyperparameters:

Parameter	Value
n_estimators	500
learning_rate	0.005
max_depth	3
subsample	0.4

Table 3.3. Best hyperparameters found for the Gradient Boosting model using 5-fold cross validation

As with the Random Forest model, we used a fixed `random_state=0` to ensure reproducibility of results.

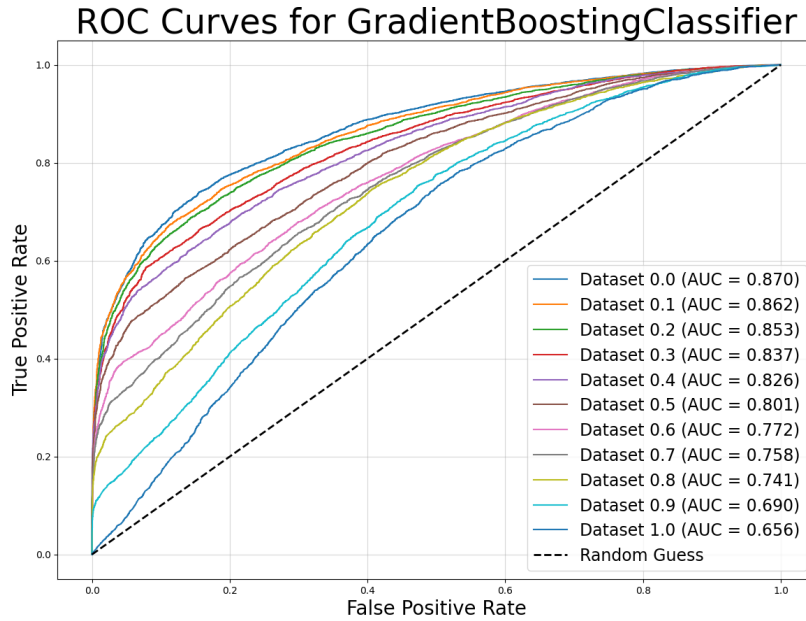The following plot shows the ROC curves and AUC scores under different levels of covariate shift:

Figure 3.3. Performance of the Gradient Boosting model under covariate shift. The AUC scores are plotted against the mixing probability p, representing the proportion of shifted data in the test set.

3.2.4 XGBoost

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting machines that offers enhanced regularization to prevent overfitting and superior computational efficiency through parallel processing.

We used the `XGBoost` library's `XGBClassifier`.

Due to incompatibility between the `XGBoost` library and `scikit-learn`'s `GridSearchCV`, we implemented a **custom parallel grid search with 5-fold cross-validation**.

This custom implementation leverages XGBoost's native **multi-threading** capabilities and **GPU acceleration** to efficiently evaluate hyperparameter combinations across multiple processing cores, significantly reducing the computational time required for model optimization.

The best hyperparameters found are shown in the following table:

Parameter	Value
n_estimators	100
learning_rate	0.1
max_depth	3
min_child_weight	1
subsample	0.8
colsample_bytree	0.8

Table 3.4. Best hyperparameters found for the XGBoost model using 5-fold cross validation

The following plot shows the ROC curves and AUC scores under different levels of covariate shift:
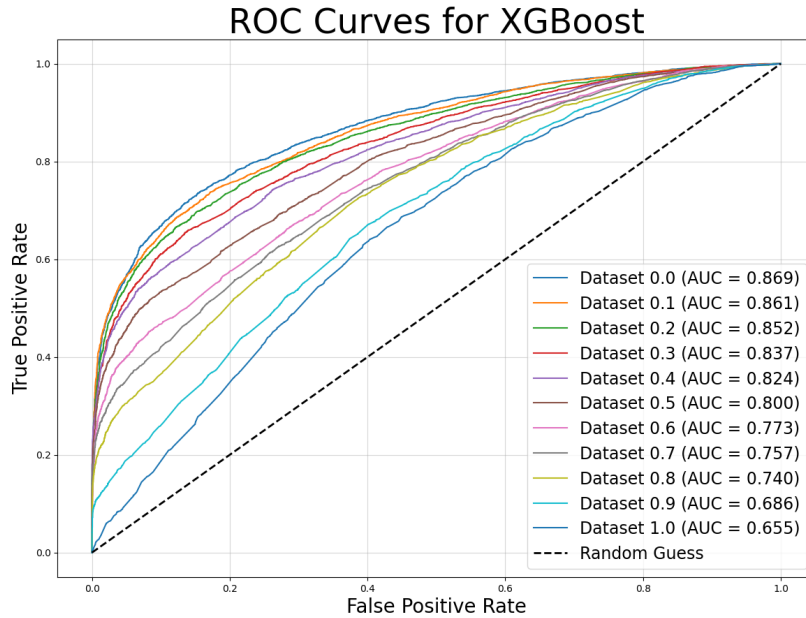

Figure 3.4. Performance of the XGBoost model under covariate shift.
The AUC scores are plotted against the mixing probability p, representing the proportion of shifted data in the test set.

3.3 Model Comparison

3.3.1 Preliminary Analysis

The following plot shows the AUC scores of the fine-tuned models under different levels of covariate shift:
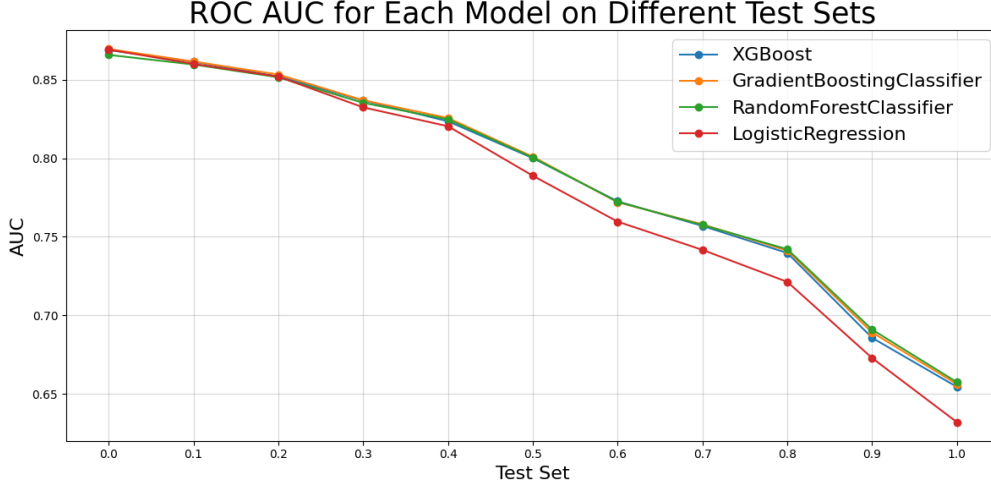

Figure 3.5. Performance comparison of fine tuned models under covariate shift.

As we can see from the plot, all the ensemble models (Random Forest, Gradient Boosting, and XGBoost) behaves very similarly under covariate shift, with a monotone decrease in performance as the mixing probability p increases. The logistic regression model, on the other hand, shows a decrease in performance as the mixing probability increases. Notably, the model performs identically on datasets with statistical mixture probabilities ranging from 0% to 20%, showing the first signs of a performance decline starting at a 30% mixture probability.

3.3.2 Statistical Analysis

To give statistical support to our study we repeated this experiment $N = 50$ times. Keeping always the same training set (in order to not train the models several times), for each repetition we defined a new shifted ditribution $\mathcal{N}(\boldsymbol{\mu}_{\text{shift}}, \boldsymbol{\Sigma}_{\text{shift}})$; created 11 **testing** datasets $\mathcal{D}_\alpha$ with $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$, where α represents the mixing probability as before and finally computed the ROC-AUC score for each model on each testing dataset.

In Figure 3.6, the boxplots depict the 50 ROC-AUC values for each model, calculated by averaging the AUC scores across the 11 testing datasets for each mixing probability α. As observed in the plot, all ensemble models exhibit comparable performance in terms of median values and interquartile range (IQR). Furthermore, all ensemble models demonstrate higher median values compared to the baseline logistic regression model.

A similar trend is evident in Figure 3.7, which presents the boxplots of the 50 ROC-AUC values for each model, focusing exclusively on datasets with a mixing probability of $\alpha = 1.0$ (fully shifted datasets). In this case, as expected, the overall performance deteriorates compared to the results in Figure 3.6. However, the difference between the median values of the ensemble models and the logistic regression model becomes even more pronounced (approximately 10%).

In both scenarios, the ensemble models exhibit lower variability compared to the logistic regression model while producing very similar results among themselves. Interestingly, there are occasional instances where the logistic regression model achieves performance comparable

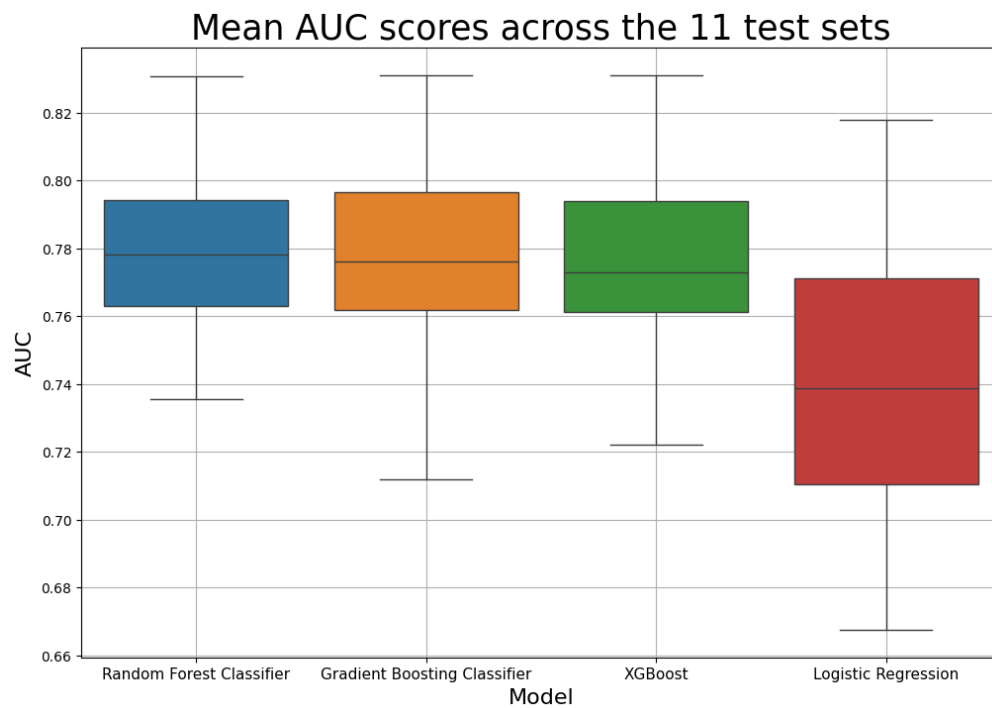to that of the ensemble models. This is likely attributable to the stochastic nature of the data generation process.
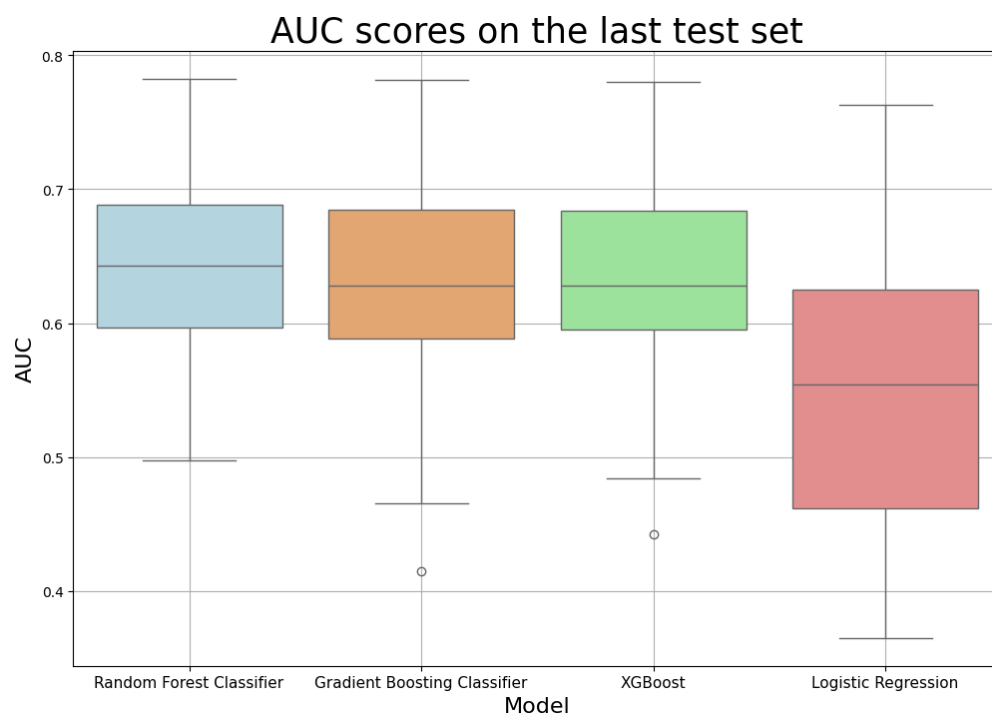

Figure 3.6.

Figure 3.7.

<div style="text-align: right; font-size: 4em;">4</div>

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 [2]. 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 [3]. In this chapter, we tested an innovative approach we term ***Random Augmentation Walk***. In particular, we will apply this pre-processing method to the training step of a Gradient Boosting model. While the results are not as promising as expected, we believe that this method has potential and should be further investigated. Since it simplifies implementation by eliminating the need for shift estimation and is adaptable to various datasets without additional shift information.

4.1   Random Augmentation Walk

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

Algorithm 4.1 Let $Data_{\text{train}}$ represent the training dataset, Size denote the size of $Data_{\text{train}}$, N specify the percentage of data to be augmented, and ε define the magnitude of the applied shift. Since excessively large or domain-irrelevant shifts can degrade performance, the parameter ε is a list of sample values drawn from a Uniform Distibution, hence each datapoint will be shifted differently. The direction of the shift is randomly selected.

Input: $Data_{\text{train}}$, *Size*, N, ε

1: $Data_{\%} \leftarrow$ random subset of N% of $Data_{train}$
2: **for** x_i in $Data_{\%}$ **do**

3:     $x_i' \leftarrow \begin{cases} X_i + \varepsilon & \text{with probability } 0.5 \\ X_i - \varepsilon & \text{with probability } 0.5 \end{cases}$
4:     $y_i' \leftarrow y_i$

5: **end for**

6: $Data_{\text{aug}} \leftarrow Data_{train} \cup Data_{\%}$
7: $Data_{\text{final}} \leftarrow \text{Downsample}(Data_{\text{aug}}, Size)$

8: **return** $Data_{\text{final}}$

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. It is important to note that despite the variation in the x_i' values, the y_i values remain the same.

> **Why Keep the Same Label?**
>
> Because these noise-shifted samples are meant to represent plausible perturbations of the same underlying data distribution. By labeling these new synthetic points consistently, we teach the model that "even if X changes by some amount, the correct label remains Y". This strategy surely holds correct up until the degree of class imbalance is kept under a reasonable proportion in the shifted sets.

4.1.1 Statistical analysis

We evaluate the impact of the R.A.W. method on the same binary classification task, repeating the experiment as illustrated in chapter 3. The experiment is conducted as follows: Each model is evaluated on 11 distinct test sets, where each test set is generated with a varying percentage of shifted data points, representing different statistical mixtures.

The models' performance is simulated and assessed across $N = 50$ instances of 11 distinct statistical mixtures, ensuring a sufficient number of trials to rigorously validate significance using Student's t-Test.

Applying R.A.W. to Gradient Boosting Classifier

Traditional Gradient Boosting Classifiers (GBC) fits an ensemble of weak learners (often decision trees) to the residuals between your training labels and current predictions. Each successive tree is trained to reduce these residuals. However, the model typically only observes the original training points, optimizing its performance for that specific feature distribution. This approach can degrade performance when test-time distributions deviate from training data as shown in chapter 3.

The results in the next session were obtained by training two instances of the same model with and without the Random Augmentation Walk preprocessing method. The models are configured as follows:

- **Baseline Model**: A Gradient Boosting Classifier is employed as a baseline model. The GBR is configured with the following hyperparameters: `n_estimators=100, max_depth=5, learning_rate=0.05`.

- **R.A.W. Model**: This 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 N to be augmented is set to 90%. Meanwhile ε controls the magnitude of shifts considered in training (in this experiment we sampled it ten times from a Uniform Distrution which ranges from 0.015 to 0.075).

4.1.2 Classification Results

The boxplot below shows the ROC AUC scores of the baseline and augmented models averaged across all shifts.
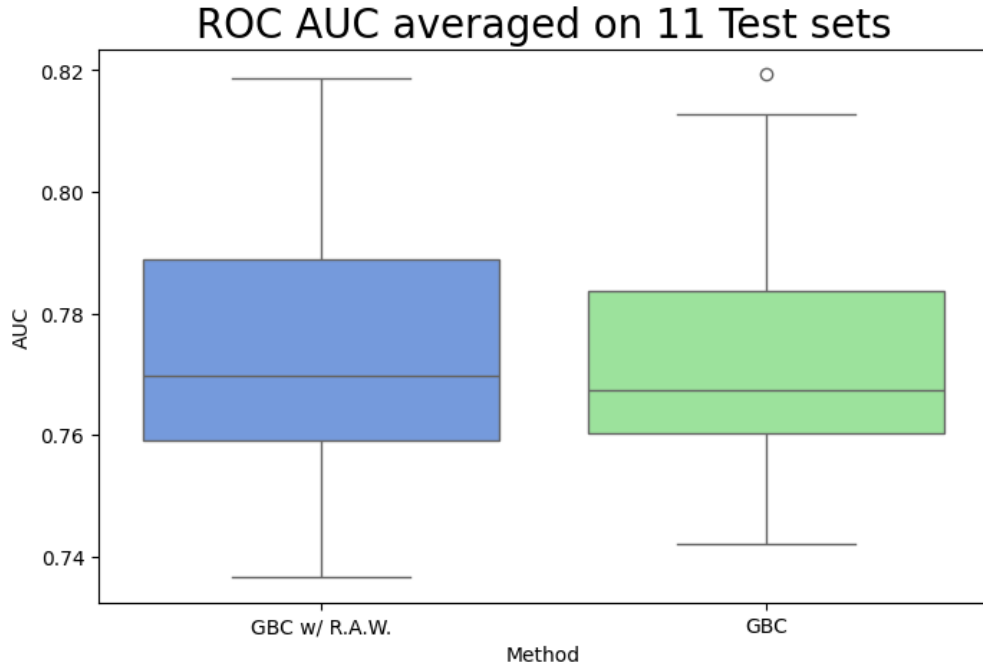
Figure 4.1. Model Performance over Shifted Test Sets. The model with R.A.W. does not outperform the baseline model averaging on test sets.

The interquartile ranges of the two models are very similar to each other, suggesting that the Random Augmentation Walk method does not significantly alter the model's performance.

To test the significance of the R.A.W. method, we conducted a Student's t-Test. The null hypothesis H_0 is that the difference between the averaged ROC AUC scores of model with and without R.A.W. is zero. The alternative hypothesis H_1 is that this difference is not zero. The results of the t-Test are shown in the table below:

	Δ_{AUC}	t-stat	p-value	95% CI
$\Delta_{\overline{AUC}}$	0.0003	0.224	0.82	[-0.0023, 0.0029]

The table shows an elevated p-value, which suggests that the difference in ROC AUC scores between the baseline and augmented models is not statistically significant. This result is consistent with the boxplot above, where the AUC scores of the two models are very close to each other. The R.A.W. method does not show any sign of influence on the model's performance.
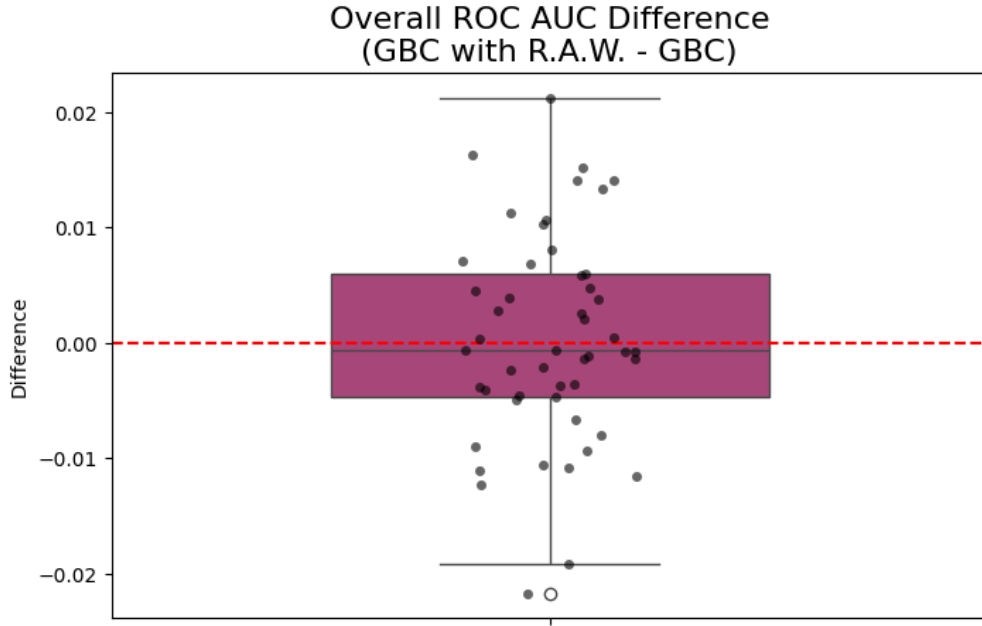
Figure 4.2. 50 ROC AUC score differences between the baseline and augmented models. Note that the difference is computed on the average AUC score across 11 test sets.

During the experiment, we also checked what was the models' behaviour in the most shifted test set, i.e. the statistical mixture which does not include any datapoints from the non-shifted dataset:
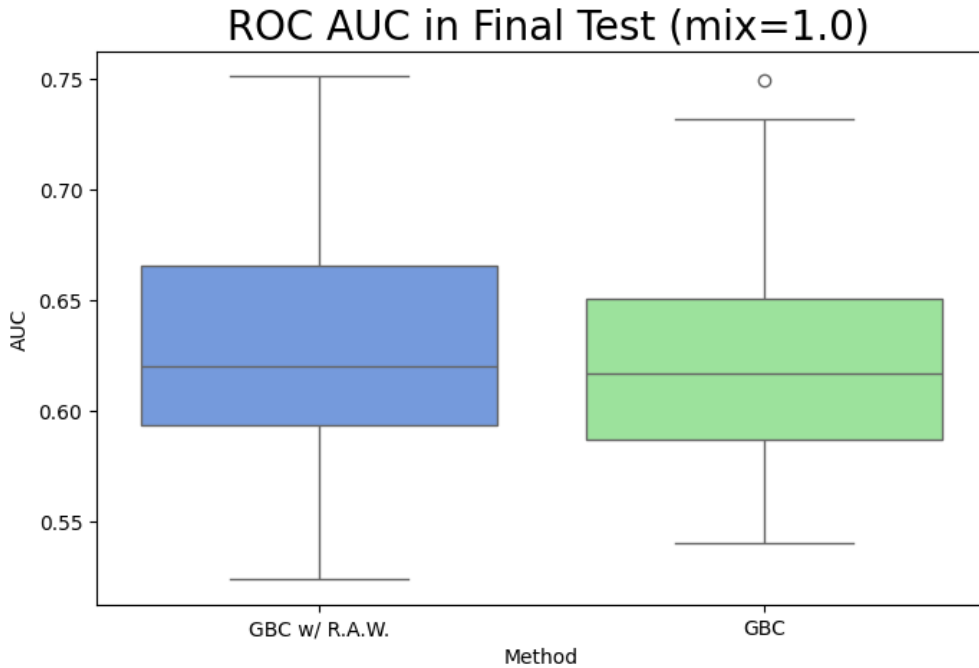

Figure 4.3. Model Performances on the Most Shifted Test Set.

A second t-Test was conducted on the differences in ROC AUC scores in the 50 instances of this last test set (mix 1.0):

	Δ_{AUC}	t-stat	p-value	95% CI
$\Delta_{\text{AUC}_{\text{last}}}$	0.0027	0.841	0.404	[-0.0038, 0.0093]

Even in the last test, the R.A.W. does not show any influence in increasing the ROC AUC score.
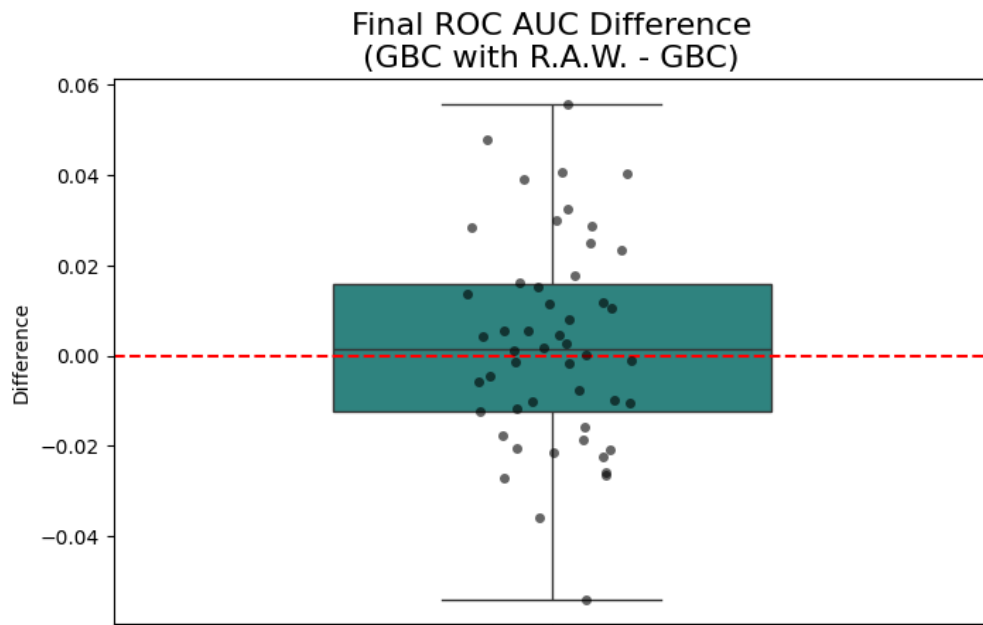

Figure 4.4. 50 ROC AUC score differences between the baseline and augmented models. Note that the difference is computed on the AUC score of the last test set.

5
Conclusions

In this work we explored how covariate shift leads to performance degradation in the roc-auc performance in a variety of models. We defined an original procedure in order to generate syntetic datasets with peculiar properties, *i.e.*, multinormality of the features, second-order polinomial relationship with random coefficients. This properties togheter with random covariance matrices generation, allowed us to have a controlled and continuos way to simulate covariate shifts. In this framework, we trained several ensemble models and evaluated their performances against a logistic classifier baseline. We found ways to efficiently parallelize the workload of tuning the ensemble models and generating data. Nontheless what the analysis shows is that the degree of degradation is equal in the more complex models – as they all perform similiarly– while the logistic classifier served as a good baseline. Interestingly, in some instances of the problem, the latter performed as well as the ensemble models; we think this result is due to the logistic classifier's higher variance. While our analysis addressed the primary question of determining the best model to handle distribution shifts among the evaluated options, we found no substantial differences in performance. This raises a subsequent question: *Why do all models behave so similarly in this context?* Our hypothesis is that the dataset was never complex enough to really challenge the models: we had to limit the features number to 3 because of compute constraints, moreover the quadratic relationship was not enough to make the problem hard. For this particular problem definition, all the models are able to achive an upperbound that is intrinsic to the problem nature itself.

Our original goal was to develop an algorithm that, differing from the literature we read, didn't have any prior knowledge on the shift, *i.e.*, on the testing datasets. At the core of this problem lays the fact that no data augmentation can be relevant enough when its not reflecting the nature of the real shift. The R.A.W. algorithm quickly became yet another way to assess the difficulty of performing well in a covariate shift scenario. Unfortunately, it didn't led to significant results but we still believe that not relying on any estimation on the testing set is a novel and still to be walked path.

While we do not propose specific directions for future work, our findings raise several important questions that merit further investigation. For instance, what insights might be gained by applying these methods to real-world datasets instead of synthetic ones? Additionally, is it possible to refine and enhance the R.A.W. algorithm to achieve better results, and if so, what approaches might be most effective? Addressing these questions could lead to meaningful advancements in understanding and mitigating the challenges posed by covariate shift.

Bibliography

[1]   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.

[2]   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.

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