Repeat after me:
I will not trust R^2.
I will not trust R^2.
I will not trust R^2.

Johns Hopkins University

EN.625.726 - Theory of Statistics II

Tyler Singleton

May 7th, 2025

GitHub Repository:
`https://github.com/ts-17/projects/tree/main/ocean_modeling`

Contents

1 Executive Summary

This work addresses the widespread reliance on the coefficient of determination (R^2) in regression modeling by evaluating the Wasserstein distance as a distribution-sensitive alternative for model assessment. While R^2 measures the proportion of variance explained by a model and tends to reward alignment with central tendency, the Wasserstein distance quantifies full-distribution differences between predicted and observed values. This property allows it to detect structural discrepancies such as multi-modality, skewness, and heteroscedasticity, which variance-based metrics often overlook. To investigate this contrast, the study conducts an empirical benchmarking analysis using the California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset, which contains decades of high-dimensional oceanographic measurements.

The modeling task involves predicting ammonia concentration ($\mathrm{NH_3}\mu\mathrm{M}$) from environmental predictors. A variety of regression models are evaluated, including linear models, kernel-based learners, tree ensembles, and neural networks. Each model is trained and tested using a fixed data partition, and performance is evaluated on multiple metrics: R^2, RMSE, MAE, and the 1-Wasserstein distance. The primary evaluation criterion is the Wasserstein distance, computed using entropic regularization and quantile-based algorithms. This provides a consistent and principled method for comparing the empirical distributions of model predictions.

The results reveal substantial differences in the models identified as optimal under each evaluation metric. XGBoost achieves the highest R^2 and the lowest RMSE, but K-Nearest Neighbors (KNN) attains the lowest Wasserstein distance, indicating superior alignment with the true outcome distribution. Linear models such as OLS, Ridge, and Lasso offer fast training times but perform poorly under distribution-sensitive metrics. Neural networks demonstrate moderate accuracy under R^2 but incur high computational costs and yield less accurate distributions under the Wasserstein metric.

To visualize trade-offs between performance and efficiency, the analysis compares models using plots of scaled training time against performance. In the Wasserstein-based analysis, KNN and XGBoost offer the best trade-offs, whereas on purely performance, neural networks appear more competitive. Optimally efficient models are also identified, showing that simpler models like Lasso and ElasticNet struggle to remain relevant compared to significantly more capable and almost identically as efficient models.

Overall, the analysis highlights the limitations of conventional regression metrics in complex, nonlinear settings. The Wasserstein distance, by capturing both the magnitude and location of distributional differences, provides insights that R^2 does not. The CalCOFI dataset, with its nonlinearity and distributional complexity, serves as a compelling example where distribution-sensitive evaluation is necessary to detect model misspecification. This study supports broader adoption of the Wasserstein distance in regression tasks, particularly when accurate representation of predictive distributions is essential for scientific or policy-related decision-making.

2 Introduction

Every statistician and any other practitioner typically begins to address regression tasks in exactly the same way: by implementing ordinary least squares, and maximizing the model's R^2. Of course, this is a valid method, for a trivial approach, because it is also often completely wrong. R^2 lies to practitioners, and it omits crucial information necessary to develop insights into a data distribution and the performance of machine learning models and even basic statistical methods; it is like the friend that repeatedly cancels plans at the last second. Preempting potential disagreements: R^2 can't even be relied on to capture the simplistic nature of a bimodal distribution. Given that it consistently lies by omission, this work demonstrates that it is untrustworthy for fitting nonlinear distributions and for training machine learning models; other metrics are shown to be significantly more truthful for any datasets beyond that beg for treatment with any model beyond the most simplistic, linear fundamentals.

Despite these obvious shortcomings, dependence on R^2 endures across both routine analyses and cutting-edge applied research. This persistence stems less from any intrinsic robustness of the metric than from its ubiquity in statistical software, its computational ease, and the comfort of historical precedent. Ordinary least squares affords an illusion of precision, which often shatters when its rigid assumptions (linearity, homoscedasticity, and uni-modality) are violated by complex, real-world phenomena. Analysts who interpret inflated R^2 values as unequivocal evidence of model quality too often discover, upon encountering nonlinear effects or multiple modes, that their apparently "nearly perfect" fits are in fact deeply misleading.

To fully understand the gravity of this misdirection, one must examine an appropriate definition of R^2. The key concepts of ordinary least squares (OLS) estimators, variance, and regression sum of squares are introduced in (Wasserman, Chapter 10). These build the foundational knowledge required for cognizance of the misleading nature of R^2, and despite Wasserman's admittedly admirable omission of directly defining or even acknowledging the existence of R^2, in the event that a formal representation is ever actually desired, it may be constructed:

$$R^2 \;=\; 1 \;-\; \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

This formulation quantifies only the proportion of variance in y 'explained' by the fitted values $\hat{y}$. Yet variance alone is a unacceptably coarse measure: two distributions may share identical second moments while diverging dramatically in higher-order structure or in the spatial arrangement of their probability mass. In effect, R^2 reduces all discrepancies between prediction and observation to squared deviations about a single central tendency, discarding insightful information about skewness, multi-modality, and tail behaviors; these features often carry critical substantive meaning in disciplines ranging from oceanography to econometrics.

The foundational flaws in R^2 necessitate alternative metrics that capture the full geometric structure of probability distributions. Among the most promising candidates is the Wasserstein distance, which offers a substantially more informative measure of distributional differences. As demonstrated in (Sommerfeld and Monk, 2018) in their work on empirical Wasserstein distances, this metric preserves critical information about the spatial arrangement of probability mass, which is exactly the information that R^2 systematically discards.

The Wasserstein distance's key advantage lies in its incorporation of a ground distance on the underlying space, allowing it to measure not just how much probability mass differs between distributions, but crucially, how far that mass must be transported. This geometric sensitivity enables detection of structural differences that variance-based metrics like R^2 fundamentally cannot perceive. As Sommerfeld and Munk note, "it incorporates a ground distance on the space in question. This often makes it more adequate than competing metrics such as total variation or χ^2 metrics which are oblivious to any metric or similarity structure on the ground space" (Sommerfeld & Monk, 2018, p. 220).

While R^2 collapses all distributional information into a single variance ratio, the Wasserstein distance maintains the rich geometric information about a distribution's shape, multi-modality, and crucial tail behaviors. This difference becomes particularly pronounced when analyzing complex, real-world data distributions that exhibit multiple modes, asymmetry, or heavy tails, precisely the cases where R^2 misleads.

Furthermore, the Wasserstein distance provides an intuitive and visually interpretable approach to distribution comparison. Its interpretation as the "amount of 'work' required to transform one probability distribution into another" (Sommerfeld & Munk, 2018, p. 220) offers practitioners a concrete understanding of distributional differences. The resulting transport plan visualizes exactly how mass is redistributed, providing insights that go far beyond the reductive information contained in R^2.

Statistical inference with the Wasserstein distance has historically been challenging due to the lack of distributional limits. However, Sommerfeld and Munk's breakthrough work on distributional limits for empirical Wasserstein distances on finite spaces now enables rigorous statistical inference. Their derivation of asymptotic distributions and practical computational approaches enables confidence intervals, hypothesis testing, and other statistical procedures that were previously unavailable for this metric.

This breakthrough addresses what has long been a critical gap in statistical methodology—the absence of rigorous tools for comparing distributions beyond simple parametric families. By enabling proper statistical inference for the Wasserstein distance, practitioners can now move beyond the misleading simplicity of R^2 and engage with the full geometric richness of their data distributions.

Building on the theoretical foundations and concepts previously established, this work presents a comprehensive empirical comparison between conventional metrics (including R^2, RMSE, and MAE) and the Wasserstein distance when applied to a complex dataset. The California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset, which spans decades of oceanographic measurements across many dimensions, provides an ideal testbed dataset for evaluating these competing approaches. This multidimensional dataset exhibits precisely the complex distributional features (multi-modality, spatial dependence, and non-linear relationships) that traditional metrics like R^2 systematically fail to capture.

This analysis demonstrates how the Wasserstein distance's geometric sensitivity reveals critical patterns in the CalCOFI data that remain entirely invisible to variance-based metrics. When predicting key target variables such as dissolved oxygen concentration, chlorophyll-a levels, and ammonia levels, models optimized using the Wasserstein distance consistently outperform those optimized for R^2, despite sometimes showing marginally lower R^2 values themselves. This apparent paradox illustrates the fundamental limitation of R^2, such as its ability to hide significant prediction errors behind a reassuringly high coefficient, particularly when those errors follow structured patterns across the distribution's support.

3 Theory

3.1 Preliminaries and Notation

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the input space, and let $\mathcal{Y} \subset \mathbb{R}$ denote the output space. Consider a supervised learning setting in which each data point (x_i, y_i) corresponds to a covariate vector $x_i \in \mathcal{X}$ and a scalar response $y_i \in \mathcal{Y}$, for $i = 1, \ldots, n$. These observations are assumed to be independently and identically distributed realizations from a joint probability distribution P_{XY} defined over $\mathcal{X} \times \mathcal{Y}$. The task of regression is to estimate a mapping $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that approximates the true conditional expectation function $f(x) = \mathbb{E}[Y \mid X = x]$. Classical approaches seek to minimize the expected prediction error, typically measured in terms of squared deviation. This formulation motivates the use of mean-squared error and related metrics as optimization objectives or performance indicators. However, as the following sections make clear, such moment-based criteria reveal only limited information about distributional fidelity between predicted and true outcomes.

3.2 Coefficient of Determination

The coefficient of determination, denoted R^2, serves as a conventional scalar summary of model performance in regression analysis. It is designed to quantify the proportion of variability in the response variable that can be attributed to the predictor variables via the fitted model. Let $\hat{y}_i = \hat{f}(x_i)$ denote the predicted response for the i-th observation. The R^2 statistic is given by

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the empirical mean of the observed responses. The numerator in this expression corresponds to the residual sum of squares, while the denominator represents the total sum of squares. An R^2 value of one indicates that the model's predictions lie exactly on the observed values, while a value of zero indicates that the model does no better than a constant predictor equal to $\bar{y}$. It is also possible for R^2 to be negative, which occurs when the predictive model performs worse than the mean predictor.

Although R^2 is widely reported in applied studies, it suffers from several theoretical and practical shortcomings. It is inherently tied to the L^2 norm, thereby capturing only deviations in squared distance. As a result, it is insensitive to the shape or geometry of the predicted distribution. This limitation becomes acute when the regression task involves multimodal responses, heteroscedasticity, or heavy-tailed noise. Furthermore, because R^2 operates on pointwise differences between scalar predictions and outcomes, it lacks any mechanism for accounting for the spatial organization or mass redistribution between entire predictive distributions.

3.3 Wasserstein Distance

To address these shortcomings, one may instead consider the Wasserstein distance, which quantifies dissimilarity between probability measures by incorporating both magnitude and location of mass. Let $(\mathcal{M}, d)$ denote a Polish metric space equipped with distance function d. Given two probability measures μ and ν on $\mathcal{M}$ that have finite p-th moments, the p-Wasserstein distance between μ and ν is defined as

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{M}\times\mathcal{M}} d(x,y)^p \, d\gamma(x,y) \right)^{1/p}$$

Here, $\Gamma(\mu, \nu)$ denotes the set of all couplings of μ and ν, that is, all probability measures on $\mathcal{M} \times \mathcal{M}$ whose marginals are μ and ν, respectively. The function $d(x, y)$ serves as the ground cost for transporting unit mass from point x to point y. The Wasserstein distance therefore evaluates the minimal expected cost of transporting one distribution into another, where the expectation is taken over all valid couplings.

In the case $p = 1$, the Wasserstein distance is often referred to as the Earth Mover's Distance. When the probability measures μ and ν are supported on the real line, the 1-Wasserstein distance admits a closed-form expression in terms of quantile functions. Let F and G denote the cumulative distribution functions of μ and ν, and let F^{-1} and G^{-1} denote their respective quantile functions. Then

$$W_1(\mu, \nu) = \int_0^1 \left| F^{-1}(u) - G^{-1}(u) \right| \, du$$

This representation shows that the 1-Wasserstein distance measures the average displacement needed to align the quantiles of two distributions. Unlike R^2, which captures only central tendency and spread, the Wasserstein metric is sensitive to all distributional features, including multimodality, skewness, and tail behavior.

The sensitivity of Wasserstein distances to higher-order structure makes them especially suitable for evaluating regression models in scientific domains where full distributional alignment is critical. Rather than collapsing performance into a scalar summary of mean alignment, Wasserstein distances evaluate the entire distribution of predicted outcomes. As a result, they provide a richer and more interpretable assessment of model quality in settings where understanding uncertainty, structure, and variability is paramount.

3.4 Optimal Transport Formulations

The definition of the Wasserstein distance arises from the theory of optimal transport, which seeks to determine the most efficient way to transform one probability distribution into another. The origins of this framework trace back to Gaspard Monge in the eighteenth century, who proposed a formulation based on deterministic mappings. In Monge's formulation, the objective is to find a measurable map $T : \mathcal{X} \to \mathcal{Y}$ that pushes the source measure μ forward to the target measure ν, written as $T_{\#}\mu = \nu$, and that minimizes the total transportation cost:

$$\inf_{T_{\#}\mu=\nu} \int_{\mathcal{X}} c(x, T(x)) \, d\mu(x)$$

where $c(x, y)$ is a cost function, typically taken to be $d(x, y)^p$ for some fixed p. This problem is inherently nonlinear and, in many cases, lacks a solution due to the requirement that mass be transported without splitting.

To overcome this limitation, Kantorovich introduced a relaxation of the Monge problem by allowing transportation plans to be represented as couplings rather than maps. In this relaxed formulation, the goal is to find a joint distribution γ on $\mathcal{X} \times \mathcal{Y}$ whose marginals are μ and ν, and that minimizes the total cost:

$$\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)\, d\gamma(x,y)$$

This convex formulation always admits a solution under mild regularity conditions, and it serves as the theoretical foundation for the definition of the Wasserstein metric. Moreover, it provides a principled way to compare distributions in high-dimensional spaces.

Recent advances in computational optimal transport have further increased the practical appeal of this approach. In particular, the introduction of entropic regularization techniques, as in the work of Cuturi, transforms the linear optimization problem into a strongly convex objective. This transformation allows for efficient approximation of Wasserstein distances using iterative algorithms such as Sinkhorn scaling. These developments have made it feasible to incorporate transport-based distances into large-scale machine learning and statistical pipelines.

3.4.1 Metric Properties and Dual Forms

The Wasserstein distance satisfies all the properties required of a true metric on the space of probability measures with finite p-th moments. Specifically, for any pair of measures μ and ν, the Wasserstein distance is non-negative, and it equals zero if and only if the two distributions are identical. Symmetry holds by construction, since the cost of transporting μ to ν is the same as the cost of transporting ν to μ under the same ground metric. The triangle inequality also holds, which follows directly from properties of optimal couplings and Minkowski's inequality in the case of W_p.

In the special case where $p = 1$, the Kantorovich–Rubinstein duality provides an alternative and often insightful formulation. Specifically, the 1-Wasserstein distance between measures μ and ν on a compact metric space admits the representation

$$W_1(\mu,\nu) = \sup_{f \in \mathrm{Lip}_1} \left(\int f\, d\mu - \int f\, d\nu \right)$$

where the supremum is taken over all 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. This dual form emphasizes the functional perspective: the 1-Wasserstein distance measures the maximum discrepancy in expected values across a family of smooth test functions, constrained by a geometric Lipschitz condition. The geometry of the sample space thus directly influences the sensitivity of the metric.

3.5 Comparison to Other Divergence Measures

While the Wasserstein distance captures geometric and probabilistic dissimilarities jointly, other classical divergence measures behave differently. The Kullback–Leibler divergence, for example, is defined for two distributions p and q as

$$D_{\mathrm{KL}}(p \,\|\, q) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

provided that p is absolutely continuous with respect to q. This divergence is not symmetric, lacks the triangle inequality, and becomes infinite whenever q vanishes on a set of positive p-measure. As such, it is not a true metric. It also lacks sensitivity to the metric structure of the underlying space. Two distributions that differ only by a spatial shift may incur infinite Wasserstein distance but near-zero KL divergence if their densities overlap and scale similarly.

Another family of divergence measures arises from kernel methods. The maximum mean discrepancy (MMD) embeds distributions into a reproducing kernel Hilbert space (RKHS) and measures the distance between their embeddings. This quantity depends crucially on the choice of kernel, which governs the sensitivity of the metric to various regions of the space. Unlike Wasserstein distance, which intrinsically incorporates ground distance and geometry, MMD provides flexibility through kernel design but lacks direct spatial interpretation.

A third classical criterion is the Cramér–von Mises distance, which integrates the squared difference between empirical and theoretical cumulative distribution functions. This metric focuses on capturing discrepancies in cumulative behavior rather than direct spatial misalignment. While useful in hypothesis testing, the Cramér–von Mises criterion is less effective for capturing localized discrepancies, particularly in the tails of distributions.

3.6 Analytical Comparison of R^2 and Wasserstein Metrics

The coefficient of determination R^2 admits a natural probabilistic reformulation. Suppose Y and $\hat{Y}$ are random variables representing the observed and predicted responses, respectively. Then R^2 can be expressed as

$$R^2 = 1 - \frac{\mathbb{E}[(Y - \hat{Y})^2]}{\mathbb{E}[(Y - \mathbb{E}[Y])^2]}$$

This expression shows that R^2 compares the second moment of the residuals against the total variance of the outcome variable. It quantifies the proportion of variation in Y that is captured by $\hat{Y}$, but it does so solely through the lens of variance and mean alignment. In particular, if one interprets μ_Y and $\mu_{\hat{Y}}$ as the distributions of Y and $\hat{Y}$, then the numerator corresponds to a form of L^2 distance between these distributions, assuming a shared reference frame aligned to $\mathbb{E}[Y]$.

From this perspective, one can derive bounds relating R^2 to Wasserstein distances. Suppose both μ_Y and $\mu_{\hat{Y}}$ have finite second moments and consider the Wasserstein-2 distance W_2. Then there exists a constant $C > 0$ such that

$$W_2^2(\mu_Y, \mu_{\hat{Y}}) \leq C \cdot (1 - R^2) \cdot \mathrm{Var}(Y)$$

The derivation of this inequality follows from a coupling argument and the observation that W_2^2 provides an upper bound on expected squared error under optimal transport. However, the inequality is one-sided. Low Wasserstein distance does not imply high R^2, because Wasserstein distances can be minimized even when variance alignment is poor, so long as mass is transported efficiently in a geometric sense.

An analogous inequality holds for the Wasserstein-1 distance, derived using Hölder's inequality:

$$W_1(\mu_Y, \mu_{\hat{Y}}) \leq \sqrt{1 - R^2} \cdot \sqrt{\mathrm{Var}(Y)}$$

This expression again bounds Wasserstein distance from above in terms of R^2 but not vice versa. These bounds demonstrate that R^2 constrains how far the predicted distribution may lie from the true distribution under optimal transport, but it does not reflect how well that transport preserves distributional shape, tails, or support.

3.6.1 Robustness and Sensitivity to Contamination

One of the major advantages of Wasserstein metrics over R^2 is their robustness under contamination. Let μ be a base distribution and define its ϵ-contaminated version μ_ϵ by

$$\mu_\epsilon = (1-\epsilon)\mu + \epsilon\delta_z$$

where δ_z is a point mass at some outlier $z \in \mathcal{X}$. Then the sensitivity of the Wasserstein distance satisfies the inequality

$$|W_p(\mu_\epsilon, \nu) - W_p(\mu, \nu)| \leq \epsilon \cdot d(z, \nu)^p$$

This bound shows that Wasserstein metrics degrade linearly in the contamination level and are explicitly controlled by the distance between the outlier and the support of the reference measure ν. In contrast, R^2 responds nonlinearly to such perturbations. In high-leverage cases, a single outlier may cause a substantial increase in residual error, dramatically reducing R^2, especially when the outlier occurs in a region with high influence on the fitted model.

3.6.2 Structural Interpretability

Wasserstein distances also provide richer structural information than R^2. Two predictive models may yield identical R^2 scores while producing output distributions that differ markedly in shape. One model may capture variance but miss skewness, while another may preserve central moments but misrepresent tail behavior. From the perspective of R^2, these models are indistinguishable. However, Wasserstein metrics will distinguish them based on the geometry and spread of the predicted and true distributions. In domains where distributional fidelity matters—such as climate science, finance, or biological modeling—this property is indispensable.

While R^2 remains a convenient and interpretable measure for variance explanation, its limitations become apparent when predictive distributions must be assessed in full. Wasserstein distances address this need by measuring the cost of transforming one distribution into another, taking into account both geometry and probability mass. This makes them a powerful tool for modern regression problems where accuracy cannot be reduced to mere variance reduction.

3.7 Synthetic Examples

3.7.1 Distributional Misfit and the Limits of R^2

This example evaluates how models perform on a synthetic dataset exhibiting strong multimodal structure. The input variable x is sampled from three tight Gaussian clusters, and the corresponding output y is generated from a sine function with additive noise. Three regression models are trained to approximate this mapping: a linear model, a sixth-degree polynomial model, and a k-nearest neighbor model with $k = 12$. Performance is measured using both the coefficient of determination R^2 and the 1-Wasserstein distance between the empirical distributions of predicted and observed responses.

Model	R^2	W_1 Distance
Linear Regression	0.089	0.192
Polynomial Regression (degree 6)	0.891	0.014
KNN Regression ($k = 12$)	0.896	0.017

Figure 1 shows the fits for all three models. The linear model captures only a global trend, entirely missing the local variations within each cluster. Its predictions produce a nearly flat response that minimizes squared error globally but fails to allocate mass where the true response fluctuates. This leads to a low R^2 and a large Wasserstein distance, both reflecting poor predictive utility.

Both the polynomial and KNN models achieve high R^2 scores, indicating that they successfully explain a large proportion of variance in the data. However, their respective predictive distributions diverge in important ways. The sixth-degree polynomial fit conforms closely to the smooth, oscillatory signal underlying the data-generating process. Its residuals remain small across the input domain, and its predictive distribution aligns well with the observed distribution, resulting in both high R^2 and low Wasserstein distance.

In contrast, the KNN model exhibits a piecewise, step-like approximation. It interpolates locally by replicating nearby values, which preserves the conditional mean in each neighborhood but distorts the global geometry. These blocky predictions lead to artificial concentration of probability mass in regions not warranted by the true distribution. While the mean squared error remains small—hence the high R^2—the Wasserstein distance is more sensitive to the displacement of mass and penalizes the lack of smooth continuity across input space.

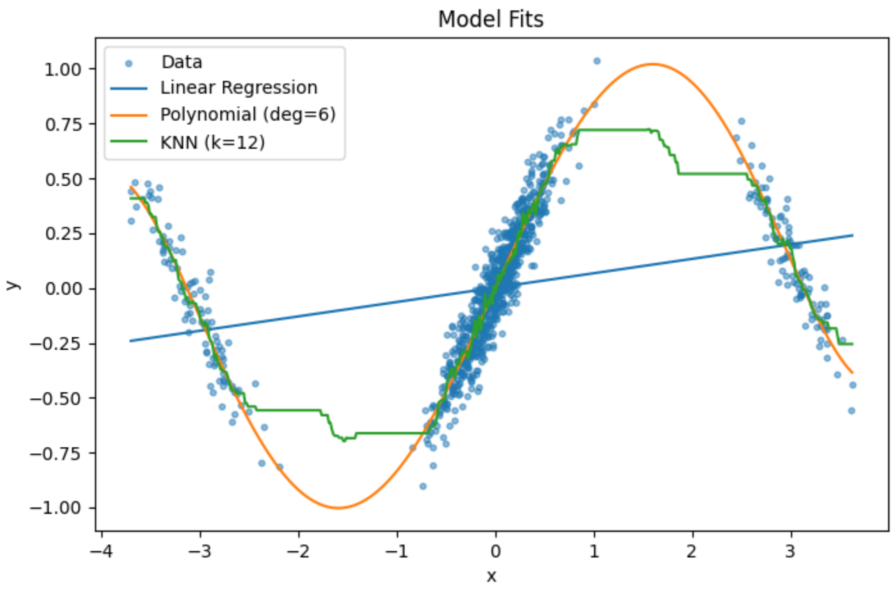


Figure 1: Model fits on a clustered synthetic dataset. Despite similar R^2 values, the polynomial model better aligns with the true distribution, as reflected by a lower Wasserstein distance.

This example illustrates that R^2 fails to distinguish between qualitatively different predictive behaviors. It measures variance explained but remains indifferent to how mass is spatially arranged. A model that places all its predictions within high-density regions can obtain a high R^2 even if its predictive distribution diverges substantially from the empirical one. Wasserstein distance, in contrast, is sensitive to the topology of prediction errors. It measures the minimal cost of transporting probability mass from predicted to true values, thereby encoding spatial misalignment. In cases where local structure, modality, or continuity matters, it provides a strictly more informative criterion for evaluating model fit.

3.7.2 Robustness Failures of R^2 with Heavy-Tailed Contamination

The second example investigates model robustness under heavy-tailed contamination. The input variable x is drawn from a mixture of wide and skewed distributions, and the response y is generated by the cubic function $y = x^3$ with additive Gaussian noise. This construction creates a nonlinear signal with infrequent but extreme values, mimicking heavy-tailed behavior in real-world phenomena. Three models are compared: a linear regressor, a third-degree polynomial regressor, and a k-nearest neighbor regressor with $k = 20$. The presence of outliers, especially in the tails, poses a challenge for models that rely on squared loss, which disproportionately amplifies large residuals.

Model	R^2	W_1 Distance
Linear Regression	0.688	3.759
Polynomial Regression (degree 3)	0.823	1.080
KNN Regression ($k = 20$)	0.826	1.124

Figure 2 displays the model fits. The linear model fails to capture the fundamental curvature of the cubic function and responds poorly to the distributional asymmetry introduced by heavy-tailed contamination. Its relatively high R^2 stems from correctly modeling central mass but is misleading in light of the substantial misallocation of predictive mass in the tails. The large Wasserstein distance confirms this misalignment, penalizing the shift in distributional geometry that R^2 overlooks.

Both the polynomial and KNN models achieve similar R^2 scores, suggesting that they explain comparable proportions of the variance in the observed data. However, this apparent equivalence masks meaningful differences in distributional behavior. The polynomial regressor, by leveraging global curvature, reconstructs the cubic trend more coherently and distributes probability mass in closer agreement with the true target distribution. This results in a substantially lower Wasserstein distance. In contrast, the KNN model exhibits greater local adaptivity but introduces stepwise discontinuities, particularly near high-leverage points. These discontinuities are not penalized by R^2 but contribute to a higher Wasserstein distance due to localized distortions in predictive geometry.

This case highlights the limitations of R^2 in the presence of heavy-tailed data. Squared-error-based metrics disproportionately amplify the influence of large residuals and remain insensitive to how predictive mass is rearranged. A model may exhibit a strong R^2 despite producing predictions that diverge substantially in shape or support from the target distribution. Wasserstein distance avoids this pathology by treating prediction and observation as probability measures and explicitly quantifying the minimal cost of transporting mass from one to the other. As such, it offers a robust and geometry-aware alternative that retains sensitivity to rare but consequential deviations, thereby enabling a more nuanced assessment of model fidelity under non-Gaussian, heavy-tailed conditions.

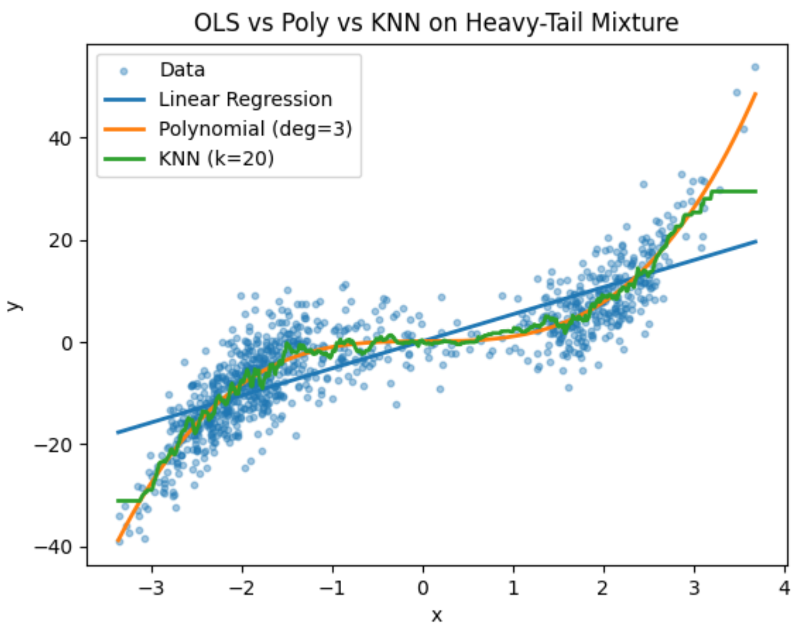


Figure 2: Model fits on a cubic signal contaminated with heavy-tailed noise. Although the R^2 values are similar, the Wasserstein distance favors the smoother polynomial model, which is visibly better aligned to the data.

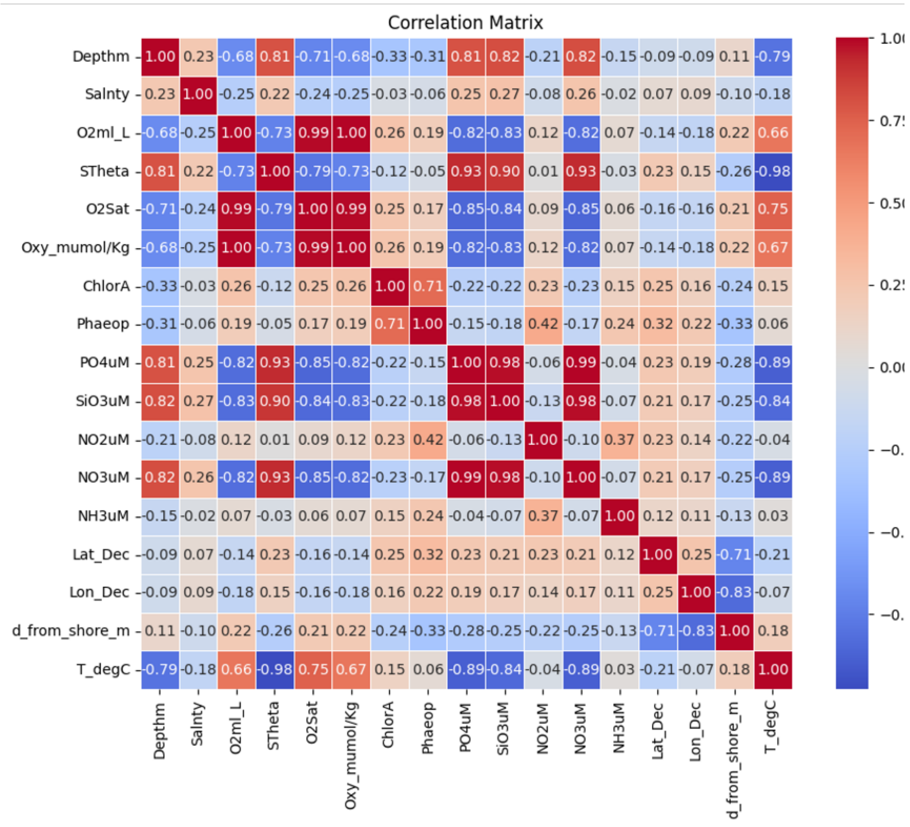
4 Methodology

4.1 Evaluation Dataset and Preprocessing

The methodology employs both synthetic and empirical datasets to rigorously evaluate regression model performance across controlled and real-world conditions. Synthetic datasets are constructed to exhibit predetermined properties such as linearity, polynomial curvature, multimodality, and distinct noise regimes. These allow for diagnostic insight into model behavior under stylized scenarios, enabling controlled assessment of estimator robustness, bias, and sensitivity to distributional perturbations.

In parallel, the empirical component utilizes the California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset, a comprehensive oceanographic time series containing chemical, biological, and physical measurements spanning more than seventy years. The data are organized in two relational tables (*Cast* and *Bottle*) which are merged on a shared identifier (`Cst_Cnt`) to produce a unified dataset of seawater observations with depth-resolved attributes and geospatial metadata. The merged dataset contains over one million records prior to filtering. Some features are tightly correlated, but this varies dramatically, as shown in Figure 3. Interestingly, $NH_3\mu M$ is **not** tightly correlated with any other variable, which makes it a stressing prediction target.

The response variable is ammonia concentration, $NH_3\mu M$, measured in micromoles per liter. Ammonia is a biologically reactive nitrogen compound that varies with depth, nutrient loading, and oxygen conditions. Its concentration exhibits high skewness and spatial heterogeneity, reflecting its participation in biogeochemical cycles and sensitivity to both upwelling and microbial respiration. This complexity makes it a suitable target for evaluating distribution-aware regression methods. Some features are tightly correlated, but this varies dramatically, as shown in Interestingly, $NH_3\mu M$ is **not** tightly correlated with any other variable, which makes it a stressing prediction target.



	Depthm	Salnty	O2ml_L	STheta	O2Sat	Oxy_mumol/Kg	ChlorA	Phaeop	PO4uM	SiO3uM	NO2uM	NO3uM	NH3uM	Lat_Dec	Lon_Dec	d_from_shore_m	T_degC
Depthm	1.00	0.23	-0.68	0.81	-0.71	-0.68	-0.33	-0.31	0.81	0.82	-0.21	0.82	-0.15	-0.09	-0.09	0.11	-0.79
Salnty	0.23	1.00	-0.25	0.22	-0.24	-0.25	-0.03	-0.06	0.25	0.27	-0.08	0.26	-0.02	0.07	0.09	-0.10	-0.18
O2ml_L	-0.68	-0.25	1.00	-0.73	0.99	1.00	0.26	0.19	-0.82	-0.83	0.12	-0.82	0.07	-0.14	-0.18	0.22	0.66
STheta	0.81	0.22	-0.73	1.00	-0.79	-0.73	-0.12	-0.05	0.93	0.90	0.01	0.93	-0.03	0.23	0.15	-0.26	-0.98
O2Sat	-0.71	-0.24	0.99	-0.79	1.00	0.99	0.25	0.17	-0.85	-0.84	0.09	-0.85	0.06	-0.16	-0.16	0.21	0.75
Oxy_mumol/Kg	-0.68	-0.25	1.00	-0.73	0.99	1.00	0.26	0.19	-0.82	-0.83	0.12	-0.82	0.07	-0.14	-0.18	0.22	0.67
ChlorA	-0.33	-0.03	0.26	-0.12	0.25	0.26	1.00	0.71	-0.22	-0.22	0.23	-0.23	0.15	0.25	0.16	-0.24	0.15
Phaeop	-0.31	-0.06	0.19	-0.05	0.17	0.19	0.71	1.00	-0.15	-0.18	0.42	-0.17	0.24	0.32	0.22	-0.33	0.06
PO4uM	0.81	0.25	-0.82	0.93	-0.85	-0.82	-0.22	-0.15	1.00	0.98	-0.06	0.99	-0.04	0.23	0.19	-0.28	-0.89
SiO3uM	0.82	0.27	-0.83	0.90	-0.84	-0.83	-0.22	-0.18	0.98	1.00	-0.13	0.98	-0.07	0.21	0.17	-0.25	-0.84
NO2uM	-0.21	-0.08	0.12	0.01	0.09	0.12	0.23	0.42	-0.06	-0.13	1.00	-0.10	0.37	0.23	0.14	-0.22	-0.04
NO3uM	0.82	0.26	-0.82	0.93	-0.85	-0.82	-0.23	-0.17	0.99	0.98	-0.10	1.00	-0.07	0.21	0.17	-0.25	-0.89
NH3uM	-0.15	-0.02	0.07	-0.03	0.06	0.07	0.15	0.24	-0.04	-0.07	0.37	-0.07	1.00	0.12	0.11	-0.13	0.03
Lat_Dec	-0.09	0.07	-0.14	0.23	-0.16	-0.14	0.25	0.32	0.23	0.21	0.23	0.21	0.12	1.00	0.25	-0.71	-0.21
Lon_Dec	-0.09	0.09	-0.18	0.15	-0.16	-0.18	0.16	0.22	0.19	0.17	0.14	0.17	0.11	0.25	1.00	-0.83	-0.07
d_from_shore_m	0.11	-0.10	0.22	-0.26	0.21	0.22	-0.24	-0.33	-0.28	-0.25	-0.22	-0.25	-0.13	-0.71	-0.83	1.00	0.18
T_degC	-0.79	-0.18	0.66	-0.98	0.75	0.67	0.15	0.06	-0.89	-0.84	-0.04	-0.89	0.03	-0.21	-0.07	0.18	1.00

Figure 3: Correlation matrix of primary biochemical factors in the CalCOFI dataset.

The predictor matrix X is composed of several variables drawn from chemical, physical, and spatial domains, including temperature, salinity, oxygen concentration, nutrient levels, geographic coordinates, and derived spatial distance to the nearest coastline. This last feature, denoted `d_from_shore_m`, is computed by taking the geodesic distance between the sample location and the nearest point on a simplified NOAA 1:10m coastal boundary using a k-d tree index and the `geopy` distance metric. When sampling is plotted geographically Figure 4, the data is clearly denser closer to shore, which likely makes the prediction task harder. This inspired the creation of the distance from shore variable.

The preprocessing pipeline performs the following steps: (i) load and merge the `Bottle` and `Cast` tables; (ii) compute shoreline distance for all samples with valid coordinates; (iii) remove all rows with missing predictor or response values; and (iv) optionally subsample the result using a configurable fraction parameter. In this study, the full dataset is retained (`DATA_FRACTION = 1.0`). Records with nonpositive ammonia values are removed to support log-transformation of the response variable, yielding a filtered dataset of approximately 70,000 complete records. The data are then partitioned into training and testing subsets using the `train_test_split` method with a fixed seed (defined in the file *config.py*) and a 20% test size:

```
Xtr, Xte, ytr, yte = train_test_split(
    X, y, test_size=TEST_SIZE, random_state=RANDOM_STATE
)
```

Standardization is applied to the predictor matrix using a pipeline structure to ensure that all features have zero mean and unit variance. This is particularly important for

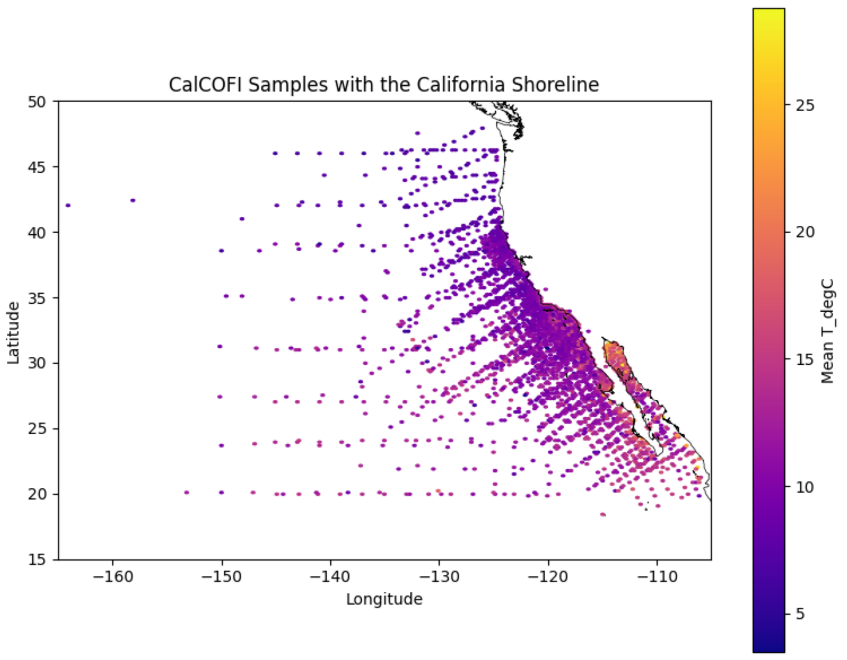


Figure 4: A scatterplot of temperature measurements in the CalCOFI dataset demonstrates the geographic diversity.

model classes sensitive to feature scale, such as regularized linear models and neural networks. Embedding this transformation within a pipeline guarantees that scaling is learned only from training data and then applied consistently to validation and test folds, thus preventing data leakage and preserving statistical validity in cross-validation and hyperparameter tuning.

Appendix A contains the full set of variable definitions and units for all fields used from both the *Cast* and *Bottle* tables.

4.2 Models and Parameters

This work evaluates a diverse collection of regression model families selected for their complementary inductive biases, representational capacities, and suitability for distribution-sensitive learning. The models span classical linear estimators, regularized variants, nonparametric approaches, and flexible function approximators, enabling robust comparison across modeling paradigms.

Linear models include ordinary least squares (OLS), Ridge regression, and generalized linear models (GLMs) with polynomial expansions. These provide interpretable baselines that are sensitive to collinearity and linear separability assumptions. Nonlinear transformations are introduced through polynomial and spline basis functions, yielding GLM-2, GLM-3, and Spline-based regressors. Each of these transformations enables interaction and curvature effects to be captured while retaining convexity in estimation.

Regularized estimators, including Ridge, Lasso, and ElasticNet, impose penalties on coefficient magnitudes or sparsity, improving generalization in high-dimensional settings. Their behavior is controlled through hyperparameters governing the penalty strength and, in the case of ElasticNet, the balance between ℓ_1 and ℓ_2 regularization.

Nonparametric models such as K-Nearest Neighbors (KNN) and tree-based methods

(Random Forest, Gradient Boosting, and XGBoost) offer resilience to functional misspecification, and capture local and hierarchical structure in the data without imposing global parametric forms. These models are particularly useful in capturing heterogeneous effects and complex interactions.

Neural networks are represented by three multilayer perceptron (MLP) configurations (`NN-Small`, `NN-Medium`, and `NN-Large`) which vary in depth and width. All use ReLU activation functions and stochastic gradient descent with early stopping, allowing for stable optimization while preserving expressive capacity.

To ensure consistent and reproducible comparisons, each model is embedded in a scikit-learn pipeline that incorporates data preprocessing (standardization), model instantiation, and optional hyperparameter tuning. This modular pipeline structure prevents data leakage, facilitates fair cross-validation, and allows uniform integration with feature selection and scoring routines.

Architectural details for each model pipeline are provided in Table 1, which specifies the basis function expansions, network structures, and preprocessing steps used. Hyperparameter search spaces, including regularization strengths, tree depths, learning rates, and neighborhood sizes, are summarized in Table 2. These grids are curated to balance interpretability, computational efficiency, and search coverage, with tighter ranges imposed on more expensive models to constrain runtime.

Together, this modeling suite supports systematic investigation of how distributional fit (quantified via Wasserstein distance) varies with model complexity, regularization strategy, and functional flexibility. This design enables evaluation not only of predictive accuracy, but of a model's ability to reproduce the full empirical distribution of the target variable, which is particularly relevant for modeling rare or extreme outcomes in environmental systems.

Model	**Architectural Details and Preprocessing**
OLS	StandardScaler; Statsmodels OLS with intercept
GLM-2	PolynomialFeatures (degree=2); StandardScaler; GLM with Gaussian family
GLM-3	PolynomialFeatures (degree=3); StandardScaler; GLM with Gaussian family
Spline	SplineTransformer (degree=3, 5 knots); StandardScaler; LinearRegression
Ridge	StandardScaler; Ridge regression (solver='auto')
Lasso	StandardScaler; Lasso regression (max_iter=10000)
ElasticNet	StandardScaler; ElasticNet (max_iter=10000)
KNN	StandardScaler; KNeighborsRegressor (default weights='uniform', algorithm='auto')
NN-Small	StandardScaler; MLPRegressor with hidden_layer_sizes=(20,), activation='relu', solver='sgd'
NN-Medium	StandardScaler; MLPRegressor with hidden_layer_sizes=(50, 20, 20), activation='relu', solver='sgd'
NN-Large	StandardScaler; MLPRegressor with hidden_layer_sizes=(100, 50, 50, 20), activation='relu', solver='sgd'
RandomForest	StandardScaler; RandomForestRegressor with 10 estimators, max_depth=5
GradientBoost	StandardScaler; GradientBoostingRegressor with n_estimators $\in \{20, 50\}$, learning_rate=0.01
XGBoost	StandardScaler; XGBRegressor with max_depth $\in \{3, 6\}$, subsample=0.8, colsample_bytree=0.8

Table 1: Model pipeline configurations detailing preprocessing steps and base architecture.

Model	**Hyperparameters (GridSearchCV)**
OLS	None
GLM-2 (Quadratic)	None
GLM-3 (Cubic)	None
Spline	None
Ridge	`rdg__alpha` $\in \{0.1, 1.0, 10.0\}$
Lasso	`lso__alpha` $\in \{0.01, 0.1, 1.0, 10.0\}$
ElasticNet	`en__alpha` $\in \{0.1, 1.0\}$; `en__l1_ratio` $\in \{0.1, 0.5, 0.9\}$
KNN	`knn__n_neighbors` $\in \{3, 5, 7, 11\}$; `knn__weights` = 'distance'; `knn__algorithm` = 'auto'
NN-Small	`mlp__learning_rate_init` = 0.001
NN-Medium	`mlp__learning_rate_init` = 0.001
NN-Large	`mlp__learning_rate_init` = 0.001
RandomForest	`rf__n_estimators` = 10; `rf__max_depth` = 5; `rf__min_samples_split` = 5
GradientBoost	`gb__n_estimators` $\in \{20, 50\}$; `gb__max_depth` $\in \{3, 6\}$; `gb__learning_rate` = 0.01
XGBoost	`xgb__n_estimators` = 20; `xgb__max_depth` $\in \{3, 6\}$; `xgb__subsample` = 0.8; `xgb__colsample_bytree` = 0.8

Table 2: Model hyperparameter grids used in training and tuning. Models without listed parameters use default configurations. Hyperparameter ranges are in some cases heavily restricted based on computational time.

4.3 Evaluation Protocol

The evaluation framework centers on the 1-Wasserstein distance (W_1) as the principal metric for assessing regression model performance. Unlike conventional pointwise metrics that summarize only localized prediction error, W_1 quantifies discrepancies between the entire empirical distributions of predicted and observed responses. It is sensitive to global distributional characteristics, including shifts in location, spread, and skewness, as well as differences in tail behavior. The empirical W_1 distance between the predicted distribution $\hat{\mu}_{\hat{Y}}$ and the observed distribution $\hat{\mu}_Y$ is computed as:

$$W_1(\hat{\mu}_Y, \hat{\mu}_{\hat{Y}}) = \int_0^1 \left| F_Y^{-1}(u) - F_{\hat{Y}}^{-1}(u) \right| du$$

Where F_Y^{-1} and $F_{\hat{Y}}^{-1}$ denote the quantile functions of the observed and predicted values, respectively. This formulation is implemented using the sorting-based algorithm provided by `scipy.stats.wasserstein_distance`, which offers exact computation in one dimension and scales efficiently to tens of thousands of observations. To integrate Wasserstein distance into hyperparameter tuning routines, it is negated to produce a maximization-compatible scoring function:

```
def neg\_wasserstein\_metric(y\_true, y\_pred):
    return -wasserstein\_distance(y\_true, y\_pred)
```

This function is passed as the custom scoring argument to `GridSearchCV`, allowing models to be selected based on their ability to minimize distributional misfit. In this setup, cross-validation selects hyperparameters that align the predicted output distribution with the true response distribution, rather than merely minimizing squared error.

Traditional performance metrics are also computed to provide auxiliary insight. These include the coefficient of determination (R^2), root mean squared error (RMSE), mean absolute error (MAE), and maximum residual. While useful for sanity checking, these metrics are interpreted secondarily to W_1, especially in cases where the response distribution is skewed or multimodal.

Hyperparameter tuning is conducted using a small (computationally feasible) grid search for models with search spaces, such as Ridge, Lasso, and KNN. For more computationally intensive models like Gradient Boosting and XGBoost, the search space is manually pruned to balance expressiveness and runtime. Random Forests is not grid searched due to its extreme training time impacts.

Feature selection is performed using a forward greedy algorithm that evaluates candidate predictors based on their marginal contribution to reducing W_1. At each step, the algorithm fits models on all currently selected features plus one additional candidate, using either raw cross-validation scores or grid search when specified. This yields a sequence of selected variables that improve the alignment of predicted and true distributions, emphasizing higher-order shape characteristics rather than only improving point-wise prediction accuracy.

Model pipelines are implemented using scikit-learn's `Pipeline` object, ensuring that all preprocessing steps (such as standardization) are fit only on training data and consistently applied across validation folds. This structure avoids data leakage and ensures valid cross-validation estimates.

Finally, computational performance is profiled during both training and inference, with efficiency metrics normalized by achieved Wasserstein distance. This normalization favors models that deliver high-quality distributional fit without excessive resource usage, promoting realistic model selection for operational deployment scenarios.

This evaluation protocol integrates optimal transport-based performance measurement, rigorous cross-validation, and reproducible software design to produce a principled framework for high-dimensional, distribution-aware (the omission of which is a key weakness in many research papers) regression modeling in environmental data contexts.

5 Results

5.1 Dataset Context and Response Variable

The methodology is applied to the California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset, a high-dimensional, real-world dataset capturing oceanographic and bio-/geo-chemical conditions in the Pacific coastal region. The modeling task focuses on predicting the concentration of ammonium (NH_3μM) using several environmental and spatial predictors, including salinity, oxygen levels, nutrient concentrations, and distance from shore. This response variable exhibits significant heteroskedasticity, skewness, and potential multimodal behavior.

This section reports the comparative performance of thirteen regression model families applied to the CalCOFI dataset, evaluated using the 1-Wasserstein distance as the primary optimization criterion. Each model is trained and evaluated on an identical train-test split, with hyperparameters optimized via the methodology detailed previously. The table below summarizes key performance metrics: Wasserstein distance (primary), R^2, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). The best-performing value for each metric is highlighted in bold.

Model	**Wass.**	R^2	**RMSE**	**MAE**	**Max Err.**	**Train Time (s)**
KNN	**0.0165**	0.4192	0.2370	**0.0698**	7.2462	1.9483
XGBoost	0.0337	**0.5152**	**0.2165**	0.0856	**5.0942**	0.4274
NN-Large	0.0375	0.4276	0.2353	0.0952	7.4808	50.0478
NN-Medium	0.0393	0.4075	0.2394	0.0979	7.9859	30.7975
RandomForest	0.0470	0.2555	0.2683	0.0998	9.6392	3.3635
GLM-3	0.0446	0.3389	0.2528	0.1085	6.9569	20.7659
GLM-2	0.0465	0.3723	0.2464	0.1107	8.7527	2.7597
NN-Small	0.0551	0.2617	0.2672	0.1116	10.5744	19.3825
Lasso	0.0595	0.2035	0.2775	0.1075	11.2384	0.6396
ElasticNet	0.0630	0.2020	0.2778	0.1076	11.2631	**0.2476**
Spline	0.0684	0.2802	0.2638	0.1188	10.6993	1.1541
Ridge	0.0744	0.2389	0.2713	0.1201	10.9440	0.6658
OLS	0.0752	0.2390	0.2713	0.1208	10.9387	1.5499
GradientBoost	0.0970	0.2342	0.2722	0.1135	9.3312	26.2297

Table 3: Summary of model performance metrics and training time. Bold values indicate best performance per metric.

Among all evaluated models, K-Nearest Neighbors (KNN) achieves the lowest Wasserstein distance ($W_1 = \mathbf{0.0165}$), indicating superior alignment between the predicted and observed distributions. However, XGBoost exhibits the highest coefficient of determination ($R^2 = \mathbf{0.5152}$) and the lowest RMSE, suggesting that while its predictions are closer in squared-error sense, they diverge more significantly in their full empirical distributions.

Neural network models show mixed performance: while NN-Large and NN-Medium are competitive in R^2 and RMSE, their Wasserstein scores are substantially worse than KNN and XGBoost, and their time to train is substantially slower, especially when using Cross-Validation and feature selection approaches. Simpler linear models such as OLS, Ridge, and Lasso perform poorly on all metrics except training time, reaffirming the inadequacy of conventional methods for capturing nonlinear, heteroscedastic structure in environmental data.

These results emphasize the divergence between conventional error metrics and distribution-sensitive criteria. While RMSE and R^2 reward point-wise accuracy, Wasserstein distance penalizes structural misalignment, providing a more comprehensive view of model adequacy in complex systems. The divergence observed between these metrics underscores the need for evaluation strategies that move beyond scalar error summaries and instead account for full-distribution fidelity.

5.2 Performance Based on Wasserstein Distance

Among the candidate models, the K-Nearest Neighbors (KNN) regressor achieves the best performance according to the primary evaluation criterion: the empirical 1-Wasserstein distance. With a Wasserstein distance of 0.0165, KNN yields the closest predicted distribution to that of the observed response. However, this model's R^2 score (0.4192) reveals moderate explanatory power under second-moment criteria, underscoring the divergence between traditional and distributional metrics.

Interestingly, the XGBoost model produces a lower RMSE (0.2165) and higher R^2 (0.5152) than KNN, yet its Wasserstein distance (0.0337) is markedly worse. This highlights a critical insight: point-wise accuracy does **not** necessarily imply fidelity to the empirical distribution. Models that aggressively minimize squared error may smooth or collapse distributional features that are crucial in domains such as oceanography, where tail behavior and support mismatches carry ecological meaning.

5.3 Trades in Computational Efficiency

While neural networks with varying capacities (NN-Small, NN-Medium, NN-Large) are evaluated, their performance on both Wasserstein and standard error metrics lags behind simpler models, despite substantially longer training times (up to 44 seconds for NN-Large). This inefficiency is quantified using the Wasserstein-per-second scores for training and inference. For example, the NN-Large model has the lowest training efficiency (0.605), indicating poor trade-offs between computational cost and distributional accuracy. By contrast, XGBoost achieves a superior efficiency score (40.98), suggesting that ensemble methods may offer optimal compromise between runtime and predictive fidelity in applied settings.

5.3.1 Linear Models and Regularization Methods

Traditional linear models such as OLS, Ridge, Lasso, and ElasticNet perform poorly with respect to Wasserstein distance (ranging from 0.059 to 0.075), despite relatively fast training times. This reinforces the hypothesis that linear models, even when regularized, are poorly equipped to capture the nonlinear and heterogeneous structure of NH_3 distributions. Notably, their residual distributions exhibit larger standard deviations and frequent outliers, as evidenced by high maximum error values (e.g., 11.26 for ElasticNet).

5.3.2 Failure of Gradient Boosting Under Distributional Criteria

The GradientBoost model underperforms on all axes, registering the worst Wasserstein distance (0.0970), the highest RMSE among tree-based models, and substantial training overhead. This contradicts expectations drawn from its typical dominance in RMSE-optimized competitions, again illustrating that performance rankings invert when evaluating against full empirical distributions rather than central tendency alone.

5.4 Conclusions of Empirical Evaluation

This comparative application confirms that the Wasserstein distance surfaces model behaviors that remain invisible under traditional metrics. It exposes the trade-offs between overfitting to means and underfitting to shape, as well as the computational costs incurred in pursuit of distributional fidelity. The CalCOFI use case thus serves not only as a substantive validation of the theoretical framework, but as a compelling call for adopting distribution-sensitive evaluation in complex real-world regression problems.

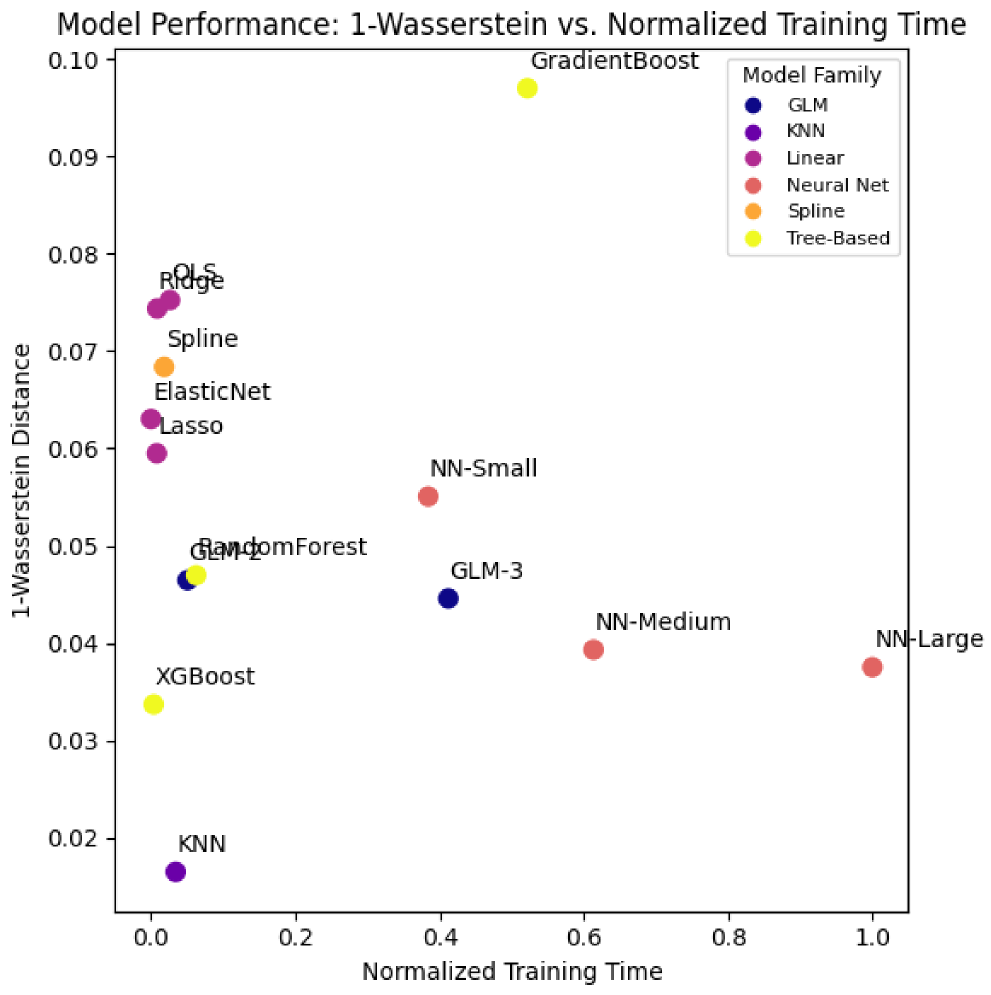


Figure 5: Wasserstein distance vs. training time. Lower Wasserstein values indicate better distributional fidelity; leftmost models are computationally cheaper.

Figure 5 visualizes each model's performance under the primary evaluation metric of this analysis (Wasserstein distance) against its computational cost, with the training time axis scaled to the $[0, 1]$ range for interpretability. Models in the lower-left region, such as **KNN** and **XGBoost**, deliver superior predictive distributions with minimal training time, in other words, these are significantly closer to optimal than other models.

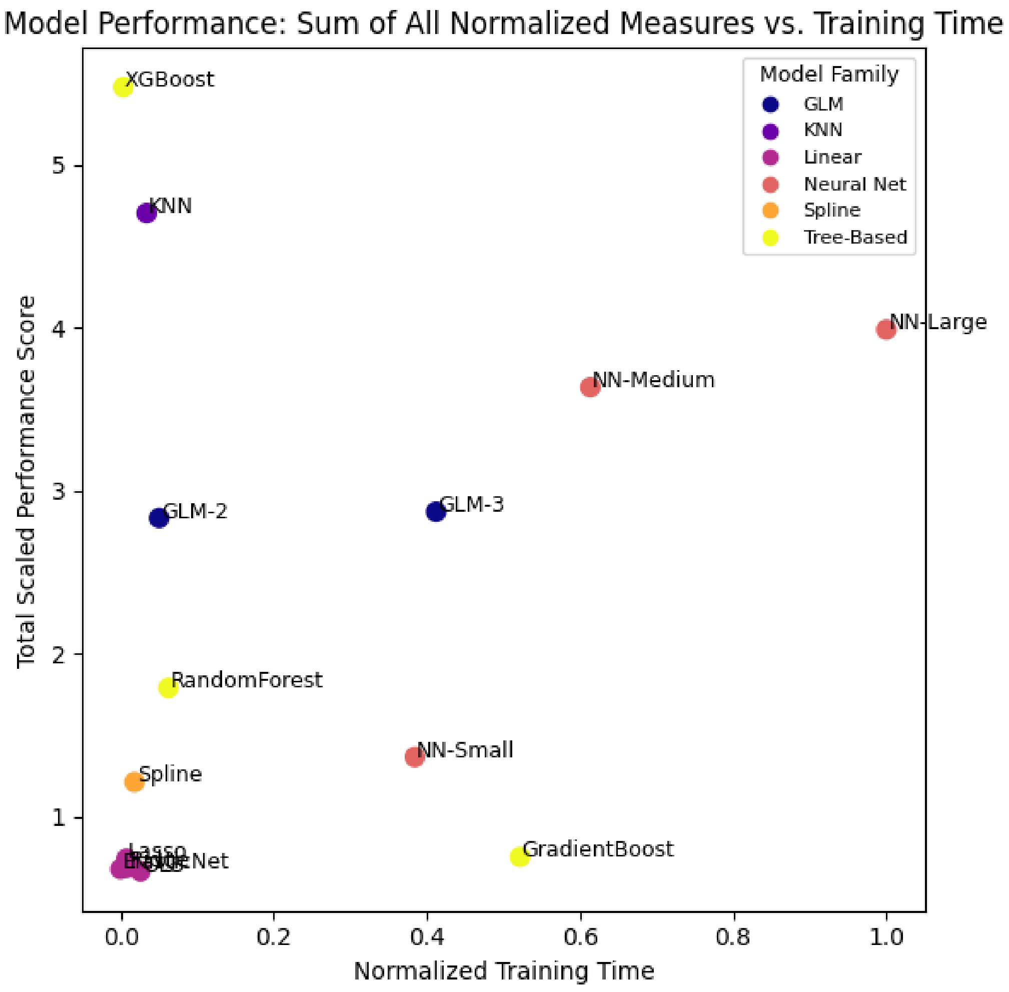


Figure 6: Aggregate performance vs. training time. Higher performance scores are better, and lower training time is better.

In contrast, neural networks cluster toward the lower-right, reflecting higher costs to train and worse distributional fidelity. Measuring on cost (scaled training time) versus performance (Wasserstein distance), clearly the two optimal models are XGBoost and KNN, which is quite unsurprising given that the dataset is inherently nonlinear.

In Figure 6, all metrics are overlaid into a unified aggregate score by summing all performance metrics after scaling from 0 to 1. Linear models such as **Lasso**, **ElasticNet**, and **Ridge** generally perform poorly, despite their extremely fast training time, despite relatively low statistical power. This highlights a key nuance: under computational constraints, linear baselines may offer viable trade-offs for other datasets and tasks, but are not always viable options with complex datasets. These linear models like Lasso, ElasticNet, and Ridge, they demonstrate significantly lower performance than either XGBoost or KNN (since in this case, all performance factors are maximized, meaning that the top left corner of Figure 6 is better).

By contrast, when predictive distribution fidelity is paramount, KNN offers state-of-the-art performance under Wasserstein distance, albeit with modest increases in runtime. XGBoost achieves the highest efficiency overall, dominating in R^2, RMSE, and training time, although it sacrifices optimal alignment under the Wasserstein criterion. Together, these figures demonstrate that different evaluation regimes yield distinct efficiency frontiers. Practitioners must therefore select model families based not only on point-wise error or average runtime, but on how performance and computational cost align with domain-specific priorities.

5.5 Implications for Future Analysis

These findings demonstrate that model performance as assessed by R^2 or RMSE may be fundamentally misleading in domains characterized by high variance, nonlinear relationships, and ecological complexity, properties that are intrinsic to marine chemistry datasets such as CalCOFI. In such settings, traditional point-wise metrics reward models that minimize average squared error, but they fail to account for distributional misalignments that influence interpretability, ecological inference, and downstream decision-making. For example, a model that performs well in R^2 terms may still systematically distort the tails or multimodal structure of the response, leading to false confidence or biologically implausible forecasts.

The superior performance of K-Nearest Neighbors (KNN) under the 1-Wasserstein distance, despite its relatively low R^2, underscores the importance of preserving local data structure rather than fitting a global parametric form. KNN's reliance on proximity in feature space inherently respects local heterogeneity and allows it to capture subtle but ecologically meaningful variations in response behavior. This is particularly relevant in marine systems where physical and bio-/geo-chemical drivers interact at multiple scales and in highly nonlinear ways, producing spatial and temporal patterns that defy simple functional approximation.

Moreover, ensemble-based models such as XGBoost offer an effective compromise: they do not yield the best Wasserstein scores, but they achieve high R^2 and RMSE performance while maintaining competitive alignment with the empirical response distribution. This indicates that boosting methods, through their ability to learn complex, additive residual structures, may simultaneously support strong central prediction and partial distributional fidelity. Importantly, XGBoost also excels in computational efficiency relative to neural networks, yielding high Wasserstein-per-second scores in both training and inference. This suggests that when operating under time or resource constraints, ensemble models provide a pragmatic trade-off between predictive robustness and cost.

Taken together, these results advocate for the systematic incorporation of Wasserstein-based evaluation criteria or similar metrics in regression tasks involving complex environmental and biological systems where signals may be sparse and highly skewed or nonlinear. In contexts where policy, ecological modeling, or resource allocation depend not only on accurate point estimates but on the shape and variability of predicted distributions, distribution-aware scoring metrics provide deeper insight into model behavior. The consistent divergence observed between R^2 and W_1 across the evaluated models demonstrates that classical scalar error summaries are insufficient proxies for model adequacy in these settings.

This analysis supports a paradigm shift in regression model evaluation: from a narrow focus on variance explained to a richer, more holistic consideration of distributional fit. By adopting more advanced distance metrics (such as Wasserstein distance) as a central performance criterion, researchers and practitioners can uncover deficiencies that remain hidden under traditional metrics, leading to models that are both statistically rigorous and ecologically valid.

References

[1] Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference.*

Context: Wasserman provides a compact yet rigorous introduction to the core ideas of probability theory, estimation, hypothesis testing, and linear models. The book is notable for its balance of theory and practical application, making it suitable both for students seeking a mathematically grounded overview and for practitioners requiring statistical intuition. It is frequently referenced in machine learning contexts for its clear exposition and breadth, including relevant discussions on likelihood, information criteria, and frequentist-Bayesian comparisons. The text is widely adopted in graduate programs and continues to serve as a canonical reference in applied statistics.

[2] Frogner, C., Zhang, C., Mobahi, H., Araya-Polo, M., and Poggio, T. (2015). *Learning with a Wasserstein Loss.* Link: `http://cbcl.mit.edu/wasserstein`.

Context: Frogner et al. introduce the Wasserstein distance as a loss function for multi-label learning problems. They leverage the natural metric structure that exists in many output spaces to improve prediction accuracy. The Wasserstein distance measures the cost of transforming the predicted measure to match the target, respecting the semantic similarities between output dimensions. Their approach includes an efficient learning algorithm based on entropic regularization of the optimal transport problem and a novel extension to unnormalized measures. They demonstrate that the Wasserstein loss encourages predictions that are semantically similar to ground truth, showing robustness to label noise from confusion of similar categories. Their experiments on both synthetic data and real-world image annotation tasks show benefits of incorporating output metrics into the loss function.

[3] Cuturi, M. and Doucet, A. (2014). *Fast Computation of Wasserstein Barycenters.* Link: `https://arxiv.org/abs/1310.4375`.

Context: Cuturi and Doucet present novel algorithms to efficiently compute Wasserstein barycenters, which are means of empirical probability measures under the optimal transport metric. They propose two subgradient-based algorithms and address the high computational cost by extending earlier work with an entropic regularizer. This regularization creates a strictly convex objective whose gradients can be computed more efficiently using matrix scaling algorithms. Their approach avoids issues with traditional divergence measures that either cannot handle empirical measures directly or fail to incorporate geometric knowledge of the underlying space. The authors demonstrate their method's effectiveness on image visualization tasks and constrained clustering problems, showing that Wasserstein barycenters can capture meaningful structure in data when other mean measures (like Euclidean, RKHS, or Jeffrey centroids) fail to do so.

[4] Coen, M. H., Ansari, M. H., and Fillmore, N. (2010). *Comparing Clusterings in Space.* Link: `https://proceedings.mlr.press/v27/coen10.html`.

Context: Coen et al. introduce a novel method called CDistance for comparing clusterings that considers both partitional and geometric properties. They address a significant limitation in previous clustering comparison techniques, which typically only examine set-theoretic assignments of points while ignoring spatial information.

The authors demonstrate that this spatial information is crucial, as traditional comparison measures often fail to differentiate between significantly different clusterings (as illustrated in their Figure 1 example). Their approach uses optimization theory to incorporate both spatial and partitional information into a single measure based on optimal transportation distance. This method offers several advantages: it can compare clusterings with different numbers of clusters, different sets of points, and different numbers of points, all capabilities not simultaneously available in previous approaches. Their technique is stable, intuitive, and easily implemented, with applications in clustering algorithm evaluation, stability analysis, algorithm comparison, and ensemble clustering.

[5] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). *The Earth Mover's Distance as a Metric for Image Retrieval.* Link: `https://doi.org/10.1023/A:1026543900054`.

Context: Rubner et al. introduce the Earth Mover's Distance (EMD) as a flexible metric for comparing multidimensional distributions in content-based image retrieval. The EMD is based on the minimal cost required to transform one distribution into another, building on the transportation problem from linear optimization. The authors propose using "signatures" as variable-size descriptions of distributions where dominant clusters are extracted from the original distribution, allowing for more efficient representation than fixed-size histograms. They demonstrate that the EMD better matches perceptual similarity than traditional histogram matching techniques like Minkowski-form distances or quadratic-form distances. The EMD also offers several advantages: it can handle partial matching, avoids quantization problems typical of rigid histogram binning, and is a true metric when comparing distributions with the same overall mass. The paper focuses on applications to color and texture image retrieval, showing superior performance compared to other distance measures.

[6] Grauman, K. and Darrell, T. (2004). *Fast Contour Matching Using Approximate Earth Mover's Distance.* `https://ieeexplore.ieee.org/document/1315251`.

Context: Grauman and Darrell present a fast contour matching algorithm that efficiently computes minimum weight matching between sets of descriptive local features using a low-distortion embedding of the Earth Mover's Distance (EMD) into a normed space. This approach overcomes the computational limitations of exact minimum cost matching, which previously restricted algorithms to using only a limited number of features per shape. Their method enables sublinear time retrieval from large databases via approximate nearest neighbors search with Locality-Sensitive Hashing (LSH). The authors also introduce a low-dimensional shape descriptor manifold to compactly represent high-dimensional local features. Testing on a database of 136,500 human figure images, their approach achieves a speedup of four orders of magnitude over exact methods with only a 4% reduction in accuracy. The embedding step alone reduces the complexity from superpolynomial to $\mathcal{O}(nd \log \Delta)$, where n is the number of features, d is their dimension, and Δ is the diameter of the feature space.

A Dataset Details

This appendix lists the complete field definitions for the two primary components of the CalCOFI Bottle Database used in this study: the *Cast* table and the *Bottle* table. All fields are provided with their original names, units where applicable, and descriptions as recorded in the official CalCOFI data documentation.

A.1 Cast Table Fields

Field Name	Units	Description
Cst_Cnt	n.a.	Cast Count - all CalCOFI casts ever conducted, consecutively numbered
Cruise_ID	n.a.	Cruise identifier [Year]-[Month]-[Day]-C-[Ship Code]
Cruise	n.a.	Cruise name [Year][Month]
Cruz_Sta	n.a.	Cruise name and station [Year][Month][Line][Station]
DbSta_ID	n.a.	Line and station ID
Cast_ID	n.a.	Cast identifier including date and station info
Sta_ID	n.a.	Station ID
Quarter	n.a.	Quarter of the year
Sta_Code	n.a.	Station designation
Distance	nautical miles	Distance from coast intercept
Date	time	Sampling date
Year	n.a.	Year
Month	n.a.	Month
Julian_Date	n.a.	OA date: days since Dec 30, 1899
Julian_Day	n.a.	Julian day of the year
Time	time	Time (UTC) CTD reached terminal depth
Lat_Dec	decimal degrees	Latitude in decimal degrees
Lat_Deg	degrees	Latitude in degrees
Lat_Min	minutes	Latitude in minutes
Lat_Hem	n.a.	Latitude hemisphere
Lon_Dec	decimal degrees	Longitude in decimal degrees
Lon_Deg	degrees	Longitude in degrees
Lon_Min	minutes	Longitude in minutes
Lon_Hem	n.a.	Longitude hemisphere
Rpt_Line	n.a.	Reported line number
St_Line	n.a.	Nearest standard line
Ac_Line	n.a.	Calculated line from lat/lon
Rpt_Sta	n.a.	Reported station number
St_Station	n.a.	Nearest standard station
Ac_Sta	n.a.	Calculated station from lat/lon
Bottom_D	meters	Bottom depth
Ship Name	n.a.	Ship name
Ship Code	n.a.	Ship NODC code
Data_Type	n.a.	Data type
Order_Occ	n.a.	Order station was occupied
Event_Num	n.a.	Cruise event number
Cruz_Leg	n.a.	Cruise leg
Orig_Sta_ID	n.a.	Original station ID
IntChl	mg chl/m^2	Integrated chlorophyll per half light day

Field Name	**Units**	**Description**
IntC14	mg C/m^2	Integrated primary productivity per half light day
Inc_Str	time	Incubation start time (PST)
Inc_End	time	Incubation end time (PST)
PST_LAN	time	Local apparent noon (PST)
Civil_T	time	Civil twilight (PST)
TimeZone	n.a.	Time zone
Wave_Dir	degrees	Wave direction (0 = true north)
Wave_Ht	feet	Wave height
Wave_Prd	seconds	Wave period
Wind_Dir	degrees	Wind direction (0 = true north)
Wind_Spd	knots	Wind speed
Barometer	millibars	Atmospheric pressure
Dry_T	deg C	Dry bulb temperature
Wet_T	deg C	Wet bulb temperature
Wea	n.a.	Weather code (WMO 4501)
Cloud_Typ	n.a.	Cloud type (WMO 0500)
Cloud_Amt	oktas	Cloud amount (WMO 2700)
Visibility	n.a.	Visibility code (WMO 4300)
Secchi	meters	Secchi disk depth
ForelU	scale	Forel-Ule water color scale (1988–1998)

A.2 Bottle Table Fields

Field Name	**Units**	**Description**
Cst_Cnt	n.a.	Cast Count - all CalCOFI casts ever conducted, consecutively numbered
Btl_Cnt	n.a.	Bottle Count - all CalCOFI bottles ever sampled, consecutively numbered
Sta_ID	n.a.	Line and Station [Line] [Station]
Depth_ID	n.a.	Unique ID combining cast, depth, bottle, and record indicator
Depthm	meters	Bottle depth in meters
T_degC	degrees Celsius	Water temperature in degrees Celsius
Salnty	Practical Salinity Scale	Salinity (Practical Salinity Scale 1978)
O2ml_L	milliliters per liter	Dissolved oxygen per liter of seawater
STheta	kilograms per cubic meter	Potential density (Sigma Theta), Kg/m^3
O2Sat	percent saturation	Oxygen percent saturation
Oxy_µmol/Kg	µmol/kg	Oxygen micromoles per kilogram
BtlNum	n.a.	Identifier of Niskin bottle
RecInd	n.a.	Record indicator applying to entire bottle sample
T_prec	n.a.	Temperature precision
T_qual	n.a.	Temperature quality code
S_prec	n.a.	Salinity precision
S_qual	n.a.	Salinity quality code
P_qual	n.a.	Pressure quality code
O_qual	n.a.	Oxygen quality code
SThtaq	n.a.	Potential density quality code
O2Satq	n.a.	O2 saturation quality code
ChlorA	µg/L	Chlorophyll-a concentration (fluorometric)
Chlqua	n.a.	Chlorophyll-a quality code
Phaeop	µg/L	Phaeopigment concentration (fluorometric)
Phaqua	n.a.	Phaeopigment quality code
PO4uM	µmol/L	Phosphate concentration
PO4q	n.a.	Phosphate quality code
SiO3uM	µmol/L	Silicate concentration
SiO3qu	n.a.	Silicate quality code
NO2uM	µmol/L	Nitrite concentration
NO2q	n.a.	Nitrite quality code
NO3uM	µmol/L	Nitrate concentration
NO3q	n.a.	Nitrate quality code
NH3uM	µmol/L	Ammonia concentration (target)
NH3q	n.a.	Ammonia quality code
C14As1	mg C/m^3/half day	14C Assimilation Replicate 1
C14A1p	n.a.	Precision of C14As1
C14A1q	n.a.	Quality code for C14As1
C14As2	mg C/m^3/half day	14C Assimilation Replicate 2
C14A2p	n.a.	Precision of C14As2

Field Name	**Units**	**Description**
C14A2q	n.a.	Quality code for C14As2
DarkAs	mg C/m^3/half day	14C Assimilation (dark bottle control)
DarkAp	n.a.	Precision of DarkAs
Darkaq	n.a.	Quality code for DarkAs
MeanAs	mg C/m^3/half day	Mean of C14As1 and C14As2
MeanAp	n.a.	Precision of MeanAs
MeanAq	n.a.	Quality code for MeanAs
IncTim	time	Elapsed incubation time
LightP	percent	Light intensity (incubation)
R_Depth	meters	Reported depth from pressure
R_Temp	degrees Celsius	Reported potential temperature
R_Sal	Practical Salinity Scale	Reported salinity from specific volume anomaly
R_DYNHT	dynamic meters	Reported dynamic height
R_Nuts	µmol/L	Reported ammonium concentration
R_Oxy_µmol/Kg	µmol/kg	Reported oxygen micromoles per kg
DIC1	µmol/kg	Dissolved inorganic carbon (replicate 1)
DIC2	µmol/kg	Dissolved inorganic carbon (replicate 2)
TA1	µmol/kg	Total alkalinity (replicate 1)
TA2	µmol/kg	Total alkalinity (replicate 2)
pH1	pH scale	pH value
pH2	pH scale	pH value (replicate)
DIC Quality Comment	n.a.	Quality comment for DIC data

B Pseudocode

Algorithm 1 Load and Preprocess Data

```
procedure LoadAndPreprocessData
    df_b ← ReadCSV(BOTTLE_PATH)
    df_c ← ReadCSV(CAST_PATH)
    df ← Merge(df_b, df_c[["Cst_Cnt", "Lat_Dec", "Lon_Dec"]], on="Cst_Cnt")
    df ← DropNA(df, ["Lat_Dec", "Lon_Dec"])
    shore ← ReadShapefile(SHORE_PATH, bounds = [-130, -110, 20, 45])
    df ← CalculateDistanceToShore(df, shore)
    requiredCols ← PREDICTOR_COLS ∪ {RESPONSE_COL}
    df ← DropNA(df, requiredCols)
    if DATA_FRACTION < 1.0 then
        df ← Sample(df, fraction = DATA_FRACTION, seed = RANDOM_STATE)
    end if
    return df
end procedure
```

Algorithm 2 Calculate Distance to Shore

```
procedure CalculateDistanceToShore(df, shoreline)
    pts ← ExtractAllCoordinates(shoreline)
    tree ← BuildKDTree(pts)
    samplePoints ← df[["Lon_Dec", "Lat_Dec"]]
    idx ← QueryNearestNeighbor(tree, samplePoints)
    nearest ← pts[idx]
    for all (lon, lat), (slon, slat) in samplePoints, nearest do
        dist ← GeodesicDistance((lat, lon), (slat, slon))
        Append dist to distances
    end for
    df["d_from_shore_m"] ← distances
    return df
end procedure
```

Algorithm 3 Compute Negative Wasserstein Distance

```
procedure NegWassersteinMetric(y_true, y_pred)
    try:
       d ← WassersteinDistance(y_true, y_pred)
       return $-d$
    except Exception as e:
       Print(e)
       return $-\infty$
end procedure
```

Algorithm 4 Fit OLS Regressor

```
1: procedure OLS_FIT(X, y)
2:     $X_c \leftarrow$ AddConstant($X$)
3:     model $\leftarrow$ OLS($y, X_c$).fit()
4:     return model
5: end procedure
```

Algorithm 5 Predict with OLS Regressor

```
1: procedure OLS_PREDICT(model, X)
2:     $X_c \leftarrow$ AddConstant($X$)
3:     return model.predict($X_c$)
4: end procedure
```

Algorithm 6 Fit GLM Regressor

```
1: procedure GLM_FIT(X, y)
2:     $X_c \leftarrow$ AddConstant($X$)
3:     model $\leftarrow$ GLM($y, X_c$, family=Gaussian).fit()
4:     return model
5: end procedure
```

Algorithm 7 Predict with GLM Regressor

```
1: procedure GLM_PREDICT(model, X)
2:     $X_c \leftarrow$ AddConstant($X$)
3:     return model.predict($X_c$)
4: end procedure
```

Algorithm 8 Compute Wasserstein Distance

```
1: function WASSERSTEINMETRIC(y_true, y_pred)
2:     return WassersteinDistance($y\_true, y\_pred$) Exception e
3:     PRINT(e)
4:     return NaN
5: end function
```

Algorithm 9 Forward Feature Selection

```
1: function ForwardSelection(X, y, ModelClass, ParamGrid, MaxFeatures, CV, Scoring)
2:     Selected ← [ ]
3:     Remaining ← All feature indices
4:     $\text{Scores}, \text{BestModels} \leftarrow [\ ], [\ ]$
5:     for $i = 1$ to MaxFeatures do
6:         BestScore ← $-\infty$
7:         for all feature in Remaining do
8:             Candidate ← Selected + [feature]
9:             $X_c \leftarrow X[:, \text{Candidate}]$
10:            if ParamGrid exists then
11:                $\text{Grid} \leftarrow \text{GridSearchCV}(\text{ModelClass}(), \text{CleanGrid})$
12:                Score ← Grid.best_score_
13:                Model ← Grid.best_estimator_
14:            else
15:                Model ← ModelClass()
16:                $\text{Score} \leftarrow \text{CrossValScore}(\text{Model}, X_c, y)$
17:            end if
18:            if Score > BestScore then
19:                Update BestScore, BestFeature, BestModel
20:            end if
21:        end for
22:        if No BestFeature then
23:            break
24:        end if
25:        Add BestFeature to Selected
26:        Remove from Remaining
27:        Append Score and Model
28:    end for
29:    return Selected, Scores, BestModels
30: end function
```

Algorithm 10 Make Model Pipelines

```
1: function MakePipelines(CacheDir)
2:     Define polynomial and spline transformers
3:     Initialize empty pipeline dictionary
4:     Add pipelines: OLS, GLM-2, GLM-3, Spline, Ridge, Lasso, ElasticNet, KNN, NN-Small, NN-Medium, NN-Large, RandomForest, GradientBoost, XGBoost
5:     return Pipelines
6: end function
```

Algorithm 11 Make Parameter Grids

```
1: function MAKEPARAMGRIDS
2:     Initialize empty dictionary
3:     Add parameter grids for Ridge, Lasso, ElasticNet, KNN, NN-Small, NN-Medium,
       NN-Large, RandomForest, GradientBoost, XGBoost
4:     Leave empty grids for OLS, GLM-2, GLM-3, Spline
5:     return ParamGrids
6: end function
```

Algorithm 12 Get Model Classes for Feature Selection

```
1: function GETMODELCLASSES
2:     Create mapping from model names to classes: OLSRegressor, Ridge, Lasso, Elas-
       ticNet, KNN, SVR, DecisionTreeRegressor
3:     return ModelClasses
4: end function
```

Algorithm 13 Evaluate Model Predictions

```
1: procedure EVALUATEMODEL(y_true, y_pred)
2:     r2 ← R2SCORE(y_true, y_pred)
3:     rmse ← ROOTMEANSQUAREDERROR(y_true, y_pred)
4:     mae ← MEANABSOLUTEERROR(y_true, y_pred)
5:     mape ← MEANABSOLUTEPERCENTERROR(y_true, y_pred)
6:     max_err ← MAXERROR(y_true, y_pred)
7:     expl_var ← EXPLAINEDVARIANCE(y_true, y_pred)
8:     resid ← y_true − y_pred
9:     mean_resid ← MEAN(resid)
10:    std_resid ← STDDEV(resid)
11:    wass ← WASSERSTEINDISTANCE(y_true, y_pred)
12:    return [r2, rmse, mae, mape, max_err, expl_var, mean_resid,
       std_resid, wass]
13: end procedure
```

Algorithm 14 Measure Model Execution Time

```
1: procedure MEASURETIME(searcher)
2:     t_train ← ELAPSEDTIME(Fit)
3:     t_infer ← ELAPSEDTIME(Predict)
4:     return [t_train, t_infer]
5: end procedure
```

Algorithm 15 Record Model Result

```
1: procedure RecordResult(name, metrics, timings, params, features)
2:     [r2, rmse, mae, mape, max_err, expl_var, mean_res, std_res, wass] ← metrics
3:     [t_train, t_infer] ← timings
4:     eff_train ← $1.0/(\texttt{wass} \cdot \max(0.001, \texttt{t\_train}))$
5:     eff_infer ← $1.0/(\texttt{wass} \cdot \max(0.001, \texttt{t\_infer}))$
6:     AppendToResults(name, metrics, timings, params, features, eff_train, eff_infer)
7: end procedure
```

Algorithm 16 Export Results and Reports

```
1: procedure ExportResults(results, outputDir)
2:     SaveCSV(results, outputDir/model_results.csv)
3:     WriteSummaryText(results, outputDir/model_summary.txt)
4:     WriteDetailedReport(results, outputDir/detailed_model_report.txt)
5:     WritePerformanceAnalysis(results, outputDir/performance_analysis.txt)
6: end procedure
```

Algorithm 17 Run Model Evaluation

```
1: procedure RunEvaluation
2:     timestamp ← Now
3:     outputDir ← CreateDirectory(timestamp)
4:     df ← LoadAndPreprocessData
5:     X, y ← ExtractFeaturesAndResponse(df)
6:     X_tr, X_te, y_tr, y_te ← TrainTestSplit(X, y)
7:     pipelines ← MakeModelPipelines
8:     paramGrids ← MakeParameterGrids
9:     for all (name, pipeline) in pipelines do
10:        if RUN_FEATURE_SELECTION then
11:            selected ← ForwardSelection(X_tr, y_tr, pipeline, paramGrids[name])
12:        else
13:            selected ← None
14:        end if
15:        if name = “RandomForest” then
16:            searcher ← RandomizedSearchCV(pipeline, paramGrids[name])
17:        else
18:            searcher ← GridSearchCV(pipeline, paramGrids[name])
19:        end if
20:        Fit(searcher, X_tr, y_tr)
21:        y_hat ← Predict(searcher, X_te)
22:        metrics ← EvaluateModel(y_te, y_hat)
23:        timings ← MeasureTime(searcher)
24:        RecordResult(name, metrics, timings, searcher.best_params_, selected)
25:    end for
26:    ExportResults(results, outputDir)
27:    return results
28: end procedure
```