Portfolio: Introduction into Statistical Learning Theory

Rassul Amantay
Student ID: 5520316

Abstract

This portfolio presents a practical and, ultimately, surprising exploration of core concepts from the Statistical Learning Theory course. The objective was to systematically compare regression models on a synthetic dataset designed with high dimensionality, non-linearity, and multicollinearity. The initial hypothesis was that complex, non-linear models would outperform simpler ones.

The results defied this expectation. The analysis revealed that regularized linear models-specifically Lasso, Gradient Descent with Early Stopping, and Ridge Regression-significantly outperformed their more complex, non-linear counterparts. The non-linear models performed even worse than the baseline OLS model, a classic demonstration of the "Curse of Dimensionality". Lasso regression achieved the best performance with a Mean Squared Error of 1.532. This work concludes that in high-dimensional, low-sample-size scenarios, a model's ability to regularize and simplify itself is far more critical than its theoretical flexibility.

1 Introduction

The field of statistical learning provides a formal framework for building models that can learn from data. A fundamental task within this field is regression, where the goal is to predict a continuous output variable based on a set of input features. While many models exist, choosing the right one depends heavily on the underlying structure of the data.

This portfolio documents a hands-on project designed to bridge the gap between the theoretical concepts discussed in the course and their practical application. Instead of analyzing models in isolation, this work follows a cohesive narrative: attempting to solve a single, challenging regression problem with an increasingly sophisticated toolkit.

To facilitate a clear comparison, a synthetic dataset was created with known properties. This allows us to understand not just "if" a model performs well, but "why", by linking its performance back to the specific data characteristics it was designed to handle. The journey will take us from the simple, interpretable OLS model to the complex, flexible Multi-Layer Perceptron, providing insights into the real-world trade-offs between model complexity, interpretability, and performance.

2 Methodology and Dataset

The foundation of this project is a synthetic dataset generated in Python. This approach was chosen over using a real-world dataset to allow for precise control over the challenges presented to the models.

2.1 Data Generation

The dataset consists of 150 samples, each with 25 input features ($d = 25$). The target variable, y, is determined by a function of only the first four features, with added Gaussian noise:

$$y = 2x_1 + \sin(2\pi x_2) + 0.5x_3^2 - 1.5x_4 + \mathcal{N}(0, 0.5^2)$$

This design introduces several key challenges:

- **Non-linearity:** The presence of $\sin(\cdot)$ and squared terms means that purely linear models will be fundamentally limited in their accuracy.
- **Irrelevant Features:** Features x_6 through x_{25} have no relationship with the target variable y. Models must be able to ignore this noise.
- **Multicollinearity:** Feature x_5 was intentionally made highly correlated with x_1 ($x_5 \approx 0.8x_1$) to test the stability of the models.

The data was split into a training set (100 samples) and a test set (50 samples). Feature scaling (standardization) was applied to the training data and then used to transform the test data.

2.2 Theoretical Framework: Risk Minimization

The goal of supervised learning is to find a function f that minimizes the **expected risk** (or generalization error), $R(f) = \mathbb{E}[l(Y, f(X))]$, where the expectation is over the true (and unknown) data distribution. The minimal possible risk is the **Bayes Risk**, $R^* = \inf_f R(f)$.

Since the true distribution is unknown, we cannot compute the expected risk directly. Instead, we use the training data $D = \{(x_j, y_j)\}_{j=1}^n$ to compute the **empirical risk**:

$$\hat{R}_n(f) = \frac{1}{n}\sum_{j=1}^{n} l(y_j, f(x_j))$$

The guiding principle for training our models is **Empirical Risk Minimization (ERM)**, where we seek a model $\hat{f}$ from a chosen family of functions $\mathcal{F}$ that minimizes this empirical risk. For regression with the squared error loss $l(y, \hat{y}) = (y - \hat{y})^2$, this becomes:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{j=1}^{n} (y_j - f(x_j))^2$$

All models in this portfolio are trained by attempting to find a parameter vector θ that minimizes this value. The ultimate goal is to achieve a low **excess risk**, $\mathbb{E}[R(\hat{f})] - R^*$.

3 Model Analysis and Results

Each model from the course was applied to the dataset in sequence.

3.1 Ordinary Least Squares (OLS)

Theoretical Foundation: OLS is the most direct application of ERM for linear models of the form $f(x) = \phi(x)^T\theta$. The OLS estimator, $\hat{\theta}_{OLS} = (\Phi^T\Phi)^{-1}\Phi^T Y$, is the one that minimizes the empirical risk. A key tool for analyzing its performance is the **bias-variance decomposition**. For any estimator $\hat{\theta}$, the expected excess risk is:

$$\mathbb{E}[R(\hat{\theta})] - R^* = \underbrace{\|\mathbb{E}[\hat{\theta}] - \theta^*\|_{\hat{\Sigma}}^2}_{\text{Bias}^2} + \underbrace{\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_{\hat{\Sigma}}^2]}_{\text{Variance}}$$

For the OLS estimator, it is unbiased ($\mathbb{E}[\hat{\theta}_{OLS}] = \theta^*$), so the bias term is zero. The entire error comes from the variance, leading to the well-known result:

$$\mathbb{E}[R(\hat{\theta}_{OLS})] - R^* = \frac{\sigma^2 d}{n}$$

This formula reveals a critical weakness: the error grows linearly with the number of dimensions d. This explains why OLS is expected to perform poorly on our high-dimensional dataset ($d = 25, n = 100$).

Results: The model was trained on the scaled training data. The final test MSE was **1.789**. This sets our baseline.

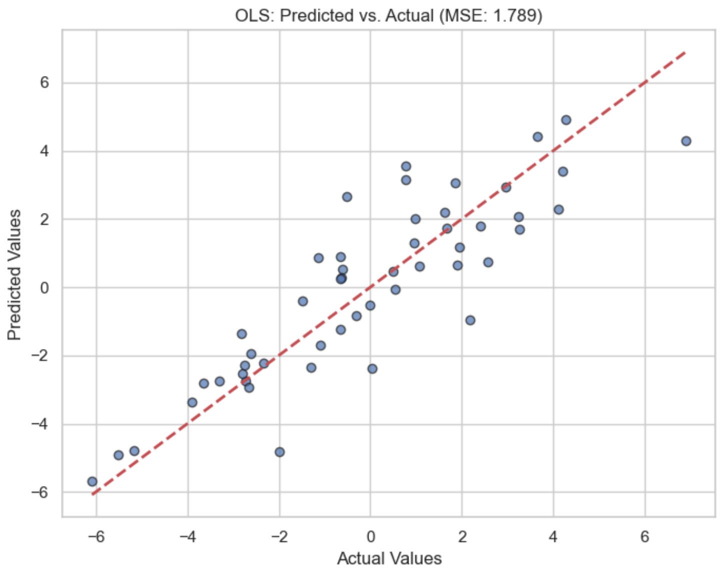

Figure 1: Predicted vs. Actual values for the OLS model on the test set.

Reflection: The high MSE was expected. With a high feature-to-sample ratio, the model overfits. The multicollinearity we introduced also likely made the matrix $\Phi^T\Phi$ ill-conditioned, further destabilizing the solution.

3.2 Ridge and Lasso Regression (L1 and L2 Regularization)

Theoretical Foundation: To combat overfitting, we add a penalty to the objective function.

- **Ridge Regression** uses an L2 penalty, minimizing: $\hat{R}_n(\theta) + \lambda\|\theta\|_2^2$. The estimator has a closed-form solution: $\hat{\theta}_\lambda = \frac{1}{n}(\hat{\Sigma} + \lambda I)^{-1}\Phi^\top y$. Unlike OLS, this estimator is biased. Its excess risk is:

$$\mathbb{E}[R(\hat{\theta}_\lambda)] - R^* = \underbrace{\lambda^2(\theta^*)^T(\hat{\Sigma} + \lambda I)^{-2}\hat{\Sigma}\theta^*}_{\text{Bias}^2\text{(increases with }\lambda)} + \underbrace{\frac{\sigma^2}{n}\text{Tr}[\hat{\Sigma}^2(\hat{\Sigma} + \lambda I)^{-2}]}_{\text{Variance (decreases with }\lambda)}$$

  This trade-off is key: we accept some bias to achieve a large reduction in variance, hoping to find a λ that minimizes their sum.

- **Lasso Regression** uses an L1 penalty: $\hat{R}_n(\theta) + \lambda\|\theta\|_1$. The L1-norm produces sparse solutions, forcing coefficients of unimportant features to zero, thus performing automatic feature selection.

Results: Using cross-validation, the optimal λ was found for both models.

- For Ridge, the optimal alpha yielded a test MSE of **1.729**.
- For Lasso, the model achieved a significantly better test MSE of **1.532**.

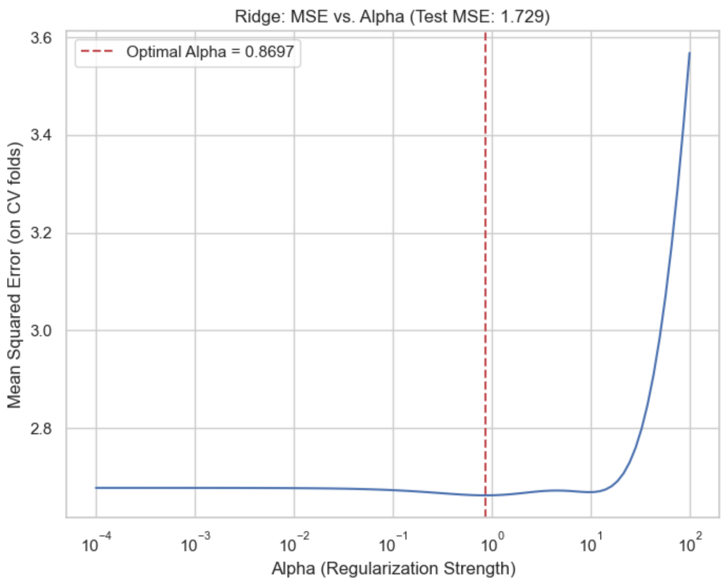

Figure 2: Finding the optimal alpha for Ridge regression.

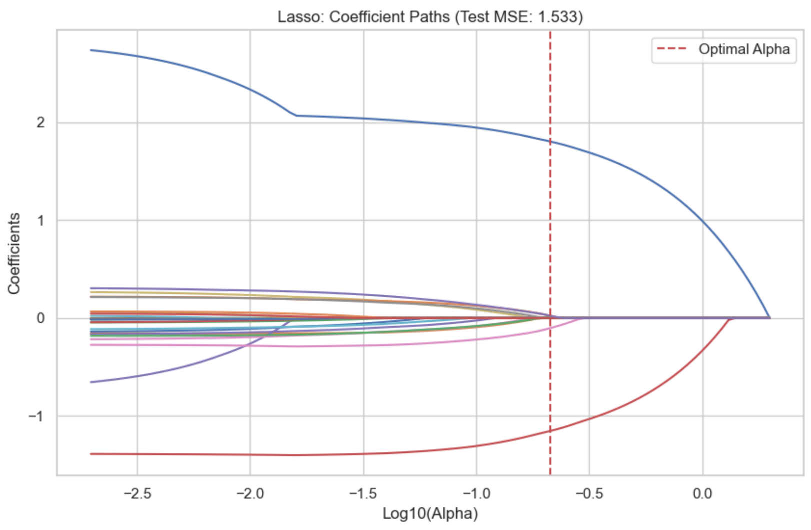

Figure 3: Coefficient paths for Lasso, showing feature selection in action.

Reflection: The results clearly show the benefit of regularization. The superior performance of Lasso suggests that for this problem, completely eliminating the 20+ noisy features was a more effective strategy than simply shrinking their weights.

3.3 Gradient Descent with Early Stopping

Theoretical Foundation: Regularization can also be achieved implicitly. By using an iterative algorithm like Gradient Descent and stopping it before it has fully converged, we prevent the model from perfectly fitting the training data noise. The GD update rule is:

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \gamma \nabla \hat{R}(\hat{\theta}_t)$$

The number of iterations T acts as a regularization parameter. The error decomposition shows that the bias is a decreasing function of T, while the variance is an increasing function of T. Early stopping aims to find the optimal T that minimizes this trade-off. For $T \to \infty$, the GD estimator converges to the OLS estimator.

Results: The learning curves showed the test error reaching a minimum at iteration 22 and then increasing. Stopping at this point, we achieved a test MSE of **1.644**.

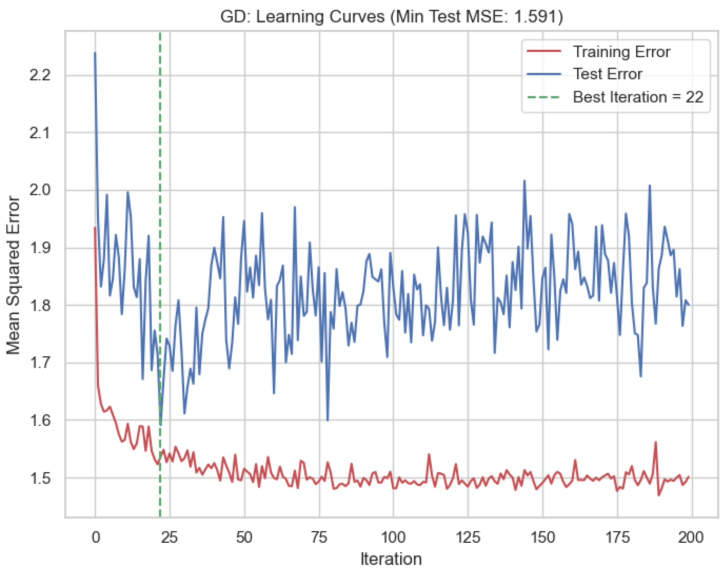

Figure 4: Training and Test error curves for Gradient Descent, demonstrating the benefit of early stopping.

Reflection: The performance of GD with early stopping is very close to that of the explicitly regularized models, confirming it as a valid and powerful regularization technique.

3.4 k-Nearest Neighbors (k-NN)

Theoretical Foundation: k-NN is a non-parametric local averaging method. It predicts the value for a point x by averaging the outputs of its closest neighbors in the feature space: $\hat{f}(x) = \frac{1}{k}\sum_{j\in N_k(x)} Y_j$. Its flexibility should, in theory, allow it to capture non-linearities.

Results: The results were deeply counter-intuitive. The optimal "k" was found to be 3, but it resulted in a catastrophic test MSE of **4.867**.

Reflection: This failure is a textbook example of the **Curse of Dimensionality**. With only 100 training points in a 25-dimensional space, the data is incredibly sparse. The distance metric becomes meaningless because the 21 noisy features overwhelm the 4 features that actually contain the signal.

3.5 Multi-Layer Perceptron (MLP)

Theoretical Foundation: The MLP is the most powerful model in our analysis, whose power stems from the **Universal Approximation Theorem**. The total error of a neural network can be seen as a sum of approximation, estimation, and optimization error. With its high number of parameters, an MLP has a very low approximation error, but its estimation error can be huge if not given enough data.

Results: Similar to k-NN, the MLP failed spectacularly, producing a test MSE of **4.058**.

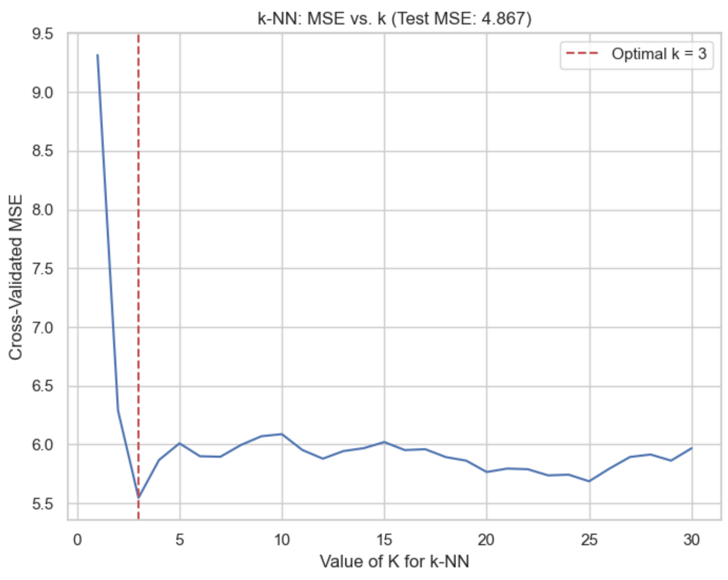

Figure 5: Finding the optimal number of neighbors for k-NN.

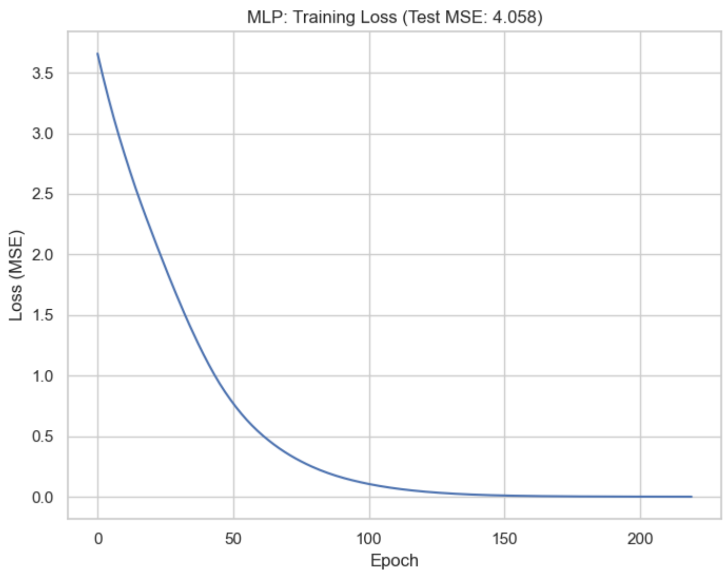

Figure 6: Training loss curve for the MLP Regressor.

Reflection: The MLP is an even more potent victim of the Curse of Dimensionality. A network with thousands of parameters has immense capacity. Trying to tune them with only 100 data points is a hopeless task. The model has too much freedom, causing the estimation error to explode and leading to extreme overfitting.

4 Summary and Conclusion

The performance of all six models is summarized in Table 1.

The final results from the initial experiment are a powerful lesson in the importance of model selection. The initial hypothesis that complex, non-linear models would perform best

Table 1: Final Test MSE for all evaluated models on 150 samples, sorted by performance.

Model	**Test MSE**
Lasso	**1.532**
GD Early Stop	1.644
Ridge	1.729
OLS	1.789
MLP	4.058
k-NN	4.867

was proven wrong. The story of this dataset is not about capturing non-linearity, but about surviving the curse of dimensionality. The regularized linear models, especially Lasso, won precisely because their inherent simplicity and strong regularization prevented them from being misled by the vast number of noisy features. The flexible, non-linear models failed because they lacked this strong structural prior and had too much freedom in a data-sparse environment.

5 Further Experiments: Scaling to 1500 Data Points

To investigate the impact of data volume on model performance, the same experimental setup was repeated with a significantly larger dataset of 1500 samples. The results, summarized in Table 2, were markedly different and aligned more closely with initial expectations.

Table 2: Final Test MSE for all models on 1500 samples.

Model	**Test MSE**
MLP	2.034773
Lasso	2.060630
Ridge	2.096483
OLS	2.098404
GD Early Stop	2.128839
k-NN (k=14)	3.999542

With a larger dataset, the MLP emerged as the top-performing model, closely followed by the regularized linear models. This reversal of fortunes highlights a key principle: the curse of dimensionality, which crippled the non-linear models in the low-sample regime, was overcome by providing a much richer dataset. With enough data, a flexible model like the MLP could finally leverage its high capacity to learn the underlying non-linear patterns, overcoming its high estimation error and outperforming the simpler linear models.

6 Personal Reflection

This project was an invaluable and surprising exercise. My initial expectation was a simple progression where model complexity would lead to better performance. The failure of k-NN and MLP on the small dataset was, at first, confusing, but it forced me to revisit the course material on the Curse of Dimensionality and truly understand its practical implications. It was a powerful lesson that a model's theoretical power does not guarantee practical success without sufficient data.

The follow-up experiment with 1500 data points provided the perfect counterpoint. Seeing the MLP's performance dramatically improve and take the lead confirmed that these complex models are not inherently flawed, but simply data-hungry. This two-part experiment has solidified my understanding that there is no single "best" model, instead, the choice is a trade-off between model complexity and data availability. For low-sample problems, regularization and simplicity are key, while for larger datasets, more complex models have the opportunity to shine.

7 Code Repository

The complete code for this project, including the data generation and model implementations, is available on GitHub:

https://github.com/RassulAmantay7/Introduction-into-Statistical-Learning-Theory