Foundations of Machine Learning

Sakshi Badole - CS24MTECH11008

November 22, 2024

ASSIGNMENT - 4

Questions: Theory

---

1 VC-Dimension

VC-Dimension of the Hypothesis Space H

The VC-dimension of the hypothesis space H is 2. Here's a detailed explanation:

Definition

The VC-dimension of a hypothesis space H is the maximum number of points that can be shattered by H. A set of points is said to be *shattered* by H if, for every possible labeling of those points, there exists a hypothesis in H that can perfectly classify them.

Hypothesis Space H

In this scenario, the hypothesis space H consists of classifiers defined by intervals (p, q) on the real line R^1. A point x is classified as 1 if it falls within the interval (p, q) and 0 otherwise.

Shattering 2 Points

Consider two points x_1 and x_2 on the real line, where $x_1 < x_2$. No matter how these points are labeled (both 0, both 1, or one of each), it is always possible to find an interval (p, q) such that:

- Both points are classified as 1: Choose $p < x_1$ and $q > x_2$.
- Only x_1 is classified as 1: Choose $p < x_1 < q$ and $x_2 > q$.
- Only x_2 is classified as 1: Choose $p < x_2 < q$ and $p > x_1$.
- Neither point is classified as 1: Choose $p > x_2$ or $q < x_1$.

Since H can perfectly classify any labeling of 2 points, it can *shatter* 2 points.

Cannot Shatter 3 Points

For three points $x_1 < x_2 < x_3$, there are some labelings that H cannot achieve. For instance, if the points are labeled 0, 1, 0 from left to right, there is no interval (p, q) that can enclose only the middle point x_2 while excluding x_1 and x_3. Thus, H cannot shatter 3 points.

Conclusion

Since H can shatter 2 points but not 3, the VC-dimension of H is:

$$\text{VC-dimension}(H) = 2.$$

---

2 Regularizer

The Regularizing Effect of Gaussian Noise on Input Data

- **Gaussian Noise:** Assume that independent Gaussian noise, denoted as $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$, is added to each input variable x_k. Each noise term ϵ_k has a mean of 0 and a variance of σ^2.
- **Noisy Input:** The noisy input data can be expressed as:
$$x'_k = x_k + \epsilon_k.$$
- **Linear Model with Noisy Input:** Plugging the noisy input into the linear model gives us:
$$y(x', w) = w_0 + \sum_{k=1}^{D} w_k(x_k + \epsilon_k).$$
- **Sum-of-Squares Error with Noisy Data:** The sum-of-squares error function using the noisy input data is:
$$\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \left(y(x'_n, w) - t_n\right)^2.$$

Connecting to Regularization Using Expectation

To find the relation between minimizing the error with noisy data and the standard error, we take the expectation over the noise distribution:

$$E_\epsilon[\tilde{E}(w)] = E_\epsilon \left[\frac{1}{2} \sum_{n=1}^{N} \left(y(x'_n, w) - t_n\right)^2 \right].$$

Expanding and Simplifying

Expanding the square inside the expectation:

$$E_\epsilon[\tilde{E}(w)] = \frac{1}{2} \sum_{n=1}^{N} E_\epsilon \left[\left(w_0 + \sum_{k=1}^{D} w_k(x_{nk} + \epsilon_{nk}) - t_n \right)^2 \right].$$

Simplifying:

$$E_\epsilon[\tilde{E}(w)] = \frac{1}{2}\sum_{n=1}^{N}\left[E_\epsilon\left(\left(w_0 + \sum_{k=1}^{D} w_k x_{nk} - t_n + \sum_{k=1}^{D} w_k \epsilon_{nk}\right)^2\right)\right].$$

Expanding the square:

$$E_\epsilon[\tilde{E}(w)] = \frac{1}{2}\sum_{n=1}^{N}\left[\left(w_0 + \sum_{k=1}^{D} w_k x_{nk} - t_n\right)^2 + 2E_\epsilon\left[\left(w_0 + \sum_{k=1}^{D} w_k x_{nk} - t_n\right)\left(\sum_{k=1}^{D} w_k \epsilon_{nk}\right)\right] + E_\epsilon\left[\left(\sum_{k=1}^{D} w_k \epsilon_{nk}\right)^2\right]\right]$$

Using the properties of Gaussian noise ($E[\epsilon_{nk}] = 0$) and independence, the middle term vanishes, leaving:

$$E_\epsilon[\tilde{E}(w)] = \frac{1}{2}\sum_{n=1}^{N}\left[\left(w_0 + \sum_{k=1}^{D} w_k x_{nk} - t_n\right)^2 + \sum_{k=1}^{D} w_k^2 E[\epsilon_{nk}^2]\right].$$

Since $E[\epsilon_{nk}^2] = \sigma^2$, we have:

$$E_\epsilon[\tilde{E}(w)] = \frac{1}{2}\sum_{n=1}^{N}\left[\left(w_0 + \sum_{k=1}^{D} w_k x_{nk} - t_n\right)^2 + \sigma^2 \sum_{k=1}^{D} w_k^2\right].$$

Recognizing the L2 Regularization Term

This can be rewritten as:

$$E_\epsilon[\tilde{E}(w)] = E(w) + \frac{N\sigma^2}{2}\sum_{k=1}^{D} w_k^2,$$

where:

- $E(w)$ is the sum-of-squares error with noise-free data.
- The term $\frac{N\sigma^2}{2}\sum_{k=1}^{D} w_k^2$ represents the L2 weight-decay regularization term with regularization strength $\lambda = N\sigma^2$.

Key Takeaways

This derivation shows that minimizing the sum-of-squares error on data with Gaussian noise is equivalent to minimizing the standard sum-of-squares error with an L2 regularization term, where the bias term w_0 is not included in the regularization. The regularization coefficient is given by:

$$\lambda = N\sigma^2.$$

In essence, adding Gaussian noise to the input data provides a regularizing effect similar to explicitly including an L2 regularization term. This helps in preventing overfitting by encouraging the model to learn more generalized patterns.

3 Hierarchical Clustering

Distance Matrix: The starting point is a distance matrix, which contains the pairwise distances between the data points (in this case, the 6 points labeled x_1 to x_6).
(a). Single Linkage Dendrogram :

- This method uses the *minimum distance* between any pair of points in the two clusters as the distance between the clusters.
- The algorithm starts by treating each data point as its own cluster.
- It then iteratively merges the two closest clusters until all data points belong to a single cluster.
- The distances between the merged clusters are shown on the y-axis of the dendrogram.

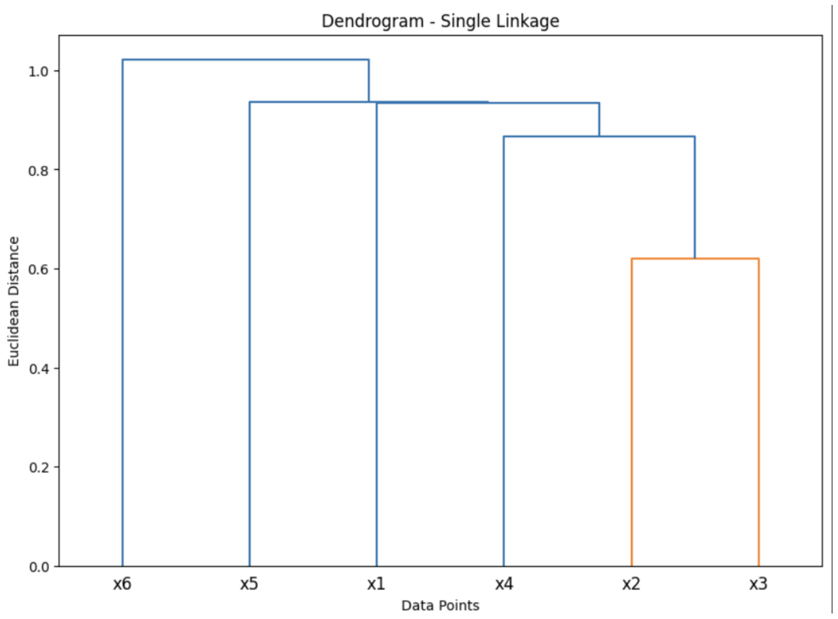


Figure 1: Single Linkage dendrogram

(b). Complete Linkage Dendrogram :
This method uses the *maximum distance* between any pair of points in the two clusters as the distance between the clusters.The algorithm follows a similar process to the single linkage, but it merges the two clusters with the smallest maximum distance between their points. The distances between the merged clusters are shown on the y-axis of the dendrogram.

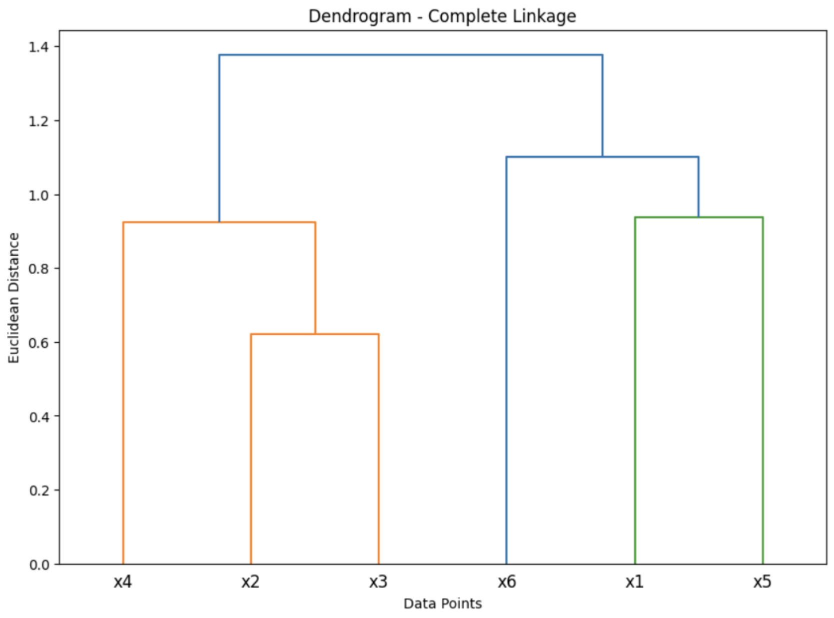


Figure 2: Complete Linkage Dendrogram

(c). Modifications to the distance matrix

Modification 1: x_5 to x_6 (from 0.67 to 0.15) This modification is aimed at aligning Iteration 3 of the clustering process. The figures following the explanation show that the single and complete linkage are same after the modifications.

- **Original Distance Matrix (Iteration 3):** In the original matrix, single-link clustering would merge $\{x_1, x_2\}$ with x_5 (distance 0.28), while complete-link clustering would merge $\{x_1, x_2\}$ with x_6 (smallest largest distance 0.61).
- **Modified Distance Matrix (Iteration 3):** By changing the distance between x_5 and x_6 to 0.15, the single-link method now prioritizes merging $\{x_1, x_2\}$ with x_6. This is because 0.15 is smaller than 0.28. The complete-link method also merges $\{x_1, x_2\}$ with x_6, as the largest distance within this new cluster becomes 0.34 (between x_1 and x_6), which is still smaller than other merging options.

Modification 2: x_4 to x_5 (from 0.45 to 0.60) This modification focuses on aligning Iteration 4 of the clustering process.

- **Original Distance Matrix (Iteration 4):** With the original matrix, single-link clustering would merge $\{x_3, x_4\}$ with x_6 (distance 0.20), while complete-link clustering would merge $\{x_3, x_4\}$ with x_5 (smallest largest distance 0.45).
- **Modified Distance Matrix (Iteration 4):** By increasing the distance between x_4 and x_5 to 0.60, the complete-link method now favors merging $\{x_3, x_4\}$ with x_6. This is because 0.60 is smaller than the largest distance between $\{x_3, x_4\}$ and any other cluster. Importantly, this change doesn't disrupt the single-link method, which still merges $\{x_3, x_4\}$ with x_6 due to the smallest distance of 0.20.

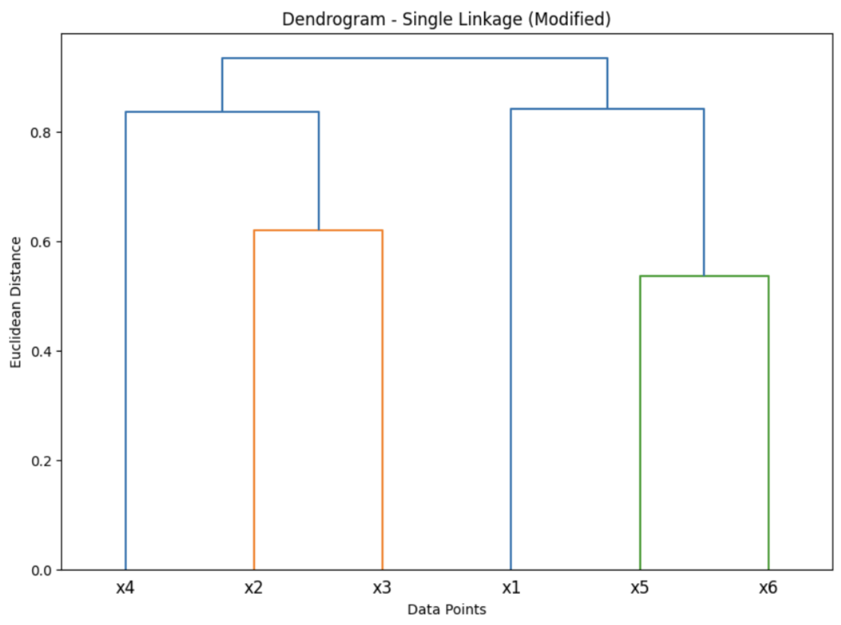


Figure 3: Single Linkage Dendrogram :Modified matrix

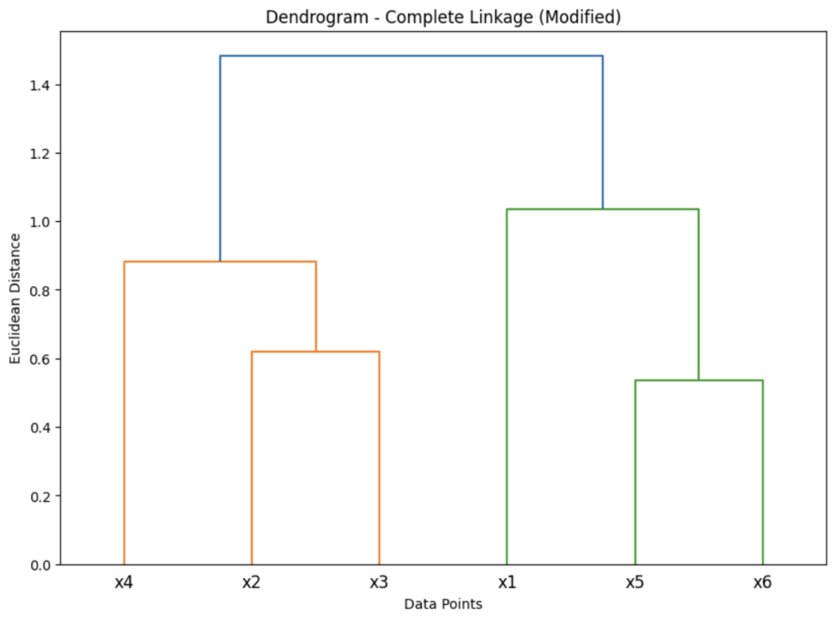


Figure 4: Complete Linkage Dendrogram : Modified Matrix

4 Principal Component Analysis

(a) Calculating the Covariance Matrix

The covariance matrix is defined as:

$$C = E[(x - E[x])(x - E[x])^T]$$

Calculate $E[x]$

Since k is uniformly distributed over $1, \ldots, M$, the probability of a being in any particular slot is $\frac{1}{M}$.

Therefore, the expected value of each element of x is $\frac{a}{M}$.

Thus, we have:

$$E[x] = \left(\frac{a}{M}, \frac{a}{M}, \ldots, \frac{a}{M}\right)^T$$

Calculate $(x - E[x])$

When a is in the i-th slot, $(x - E[x])$ is:

$$(x - E[x]) = \left(-\frac{a}{M}, \ldots, -\frac{a}{M}, a\left(1 - \frac{1}{M}\right), -\frac{a}{M}, \ldots, -\frac{a}{M}\right)^T$$

where $a\left(1 - \frac{1}{M}\right)$ is in the i-th slot.

Calculate $(x - E[x])(x - E[x])^T$

The resulting matrix will have the following elements:

- $a^2\left(1 - \frac{1}{M}\right)^2$ on the diagonal.
- $\frac{a^2}{M^2}$ on all off-diagonal elements.

Calculate $E[(x - E[x])(x - E[x])^T]$

Since the probability of a being in any slot is $\frac{1}{M}$, the expected value of the matrix is:

$$C_{ij} = \lambda + \mu\delta_{ij}$$

where:

$$\lambda = E\left[\frac{a^2}{M^2}\right], \quad \mu = E\left[a^2\left(1 - \frac{1}{M}\right)^2\right] - E\left[\frac{a^2}{M^2}\right]$$

and δ_{ij} is the Kronecker delta (equal to 1 if $i = j$, and 0 otherwise).

(b) Eigenvectors and Eigenvalues

Proof that the covariance matrix C, with $C_{ij} = \lambda + \mu\delta_{ij}$, has one eigenvector of the form $(1, \dots, 1)$ and the other eigenvectors all have the same eigenvalue.

Recall that an eigenvector v of a matrix C satisfies the equation:

$$Cv = ev,$$

where e is the eigenvalue corresponding to the eigenvector v.

Step 1: Show $(1, \dots, 1)$ is an eigenvector.

Let $v = (1, \dots, 1)^T$.

Then the i-th element of Cv is:

$$(Cv)_i = \sum_{j=1}^{M} C_{ij} v_j = \sum_{j=1}^{M} (\lambda + \mu\delta_{ij}).$$

The Kronecker delta δ_{ij} equals 1 when $i = j$ and 0 otherwise. Therefore:

$$(Cv)_i = M\lambda + \mu.$$

This is a constant times the i-th element of v. Therefore, v is an eigenvector of C with eigenvalue $e = M\lambda + \mu$.

Step 2: Show that other eigenvectors have the same eigenvalue.

Consider any vector u orthogonal to v. This means the dot product of u and v is zero:

$$u^T v = \sum_{i=1}^{M} u_i = 0.$$

The i-th element of Cu is:

$$(Cu)_i = \sum_{j=1}^{M} C_{ij} u_j = \sum_{j=1}^{M} (\lambda + \mu\delta_{ij}) u_j = \lambda \sum_{j=1}^{M} u_j + \mu u_i.$$

Since $u^T v = 0$, we have:

$$\sum_{j=1}^{M} u_j = 0.$$

Therefore:

$$(Cu)_i = \mu u_i.$$

This shows that $Cu = \mu u$. Therefore, u is an eigenvector of C with eigenvalue $e = \mu$.

The covariance matrix $\mathbf{C}$ has the form:

$$C_{ij} = \lambda + \mu\delta_{ij}$$

where $\lambda = \frac{E[a^2]}{M}$ and $\mu = 0$. The eigenvectors and eigenvalues are:

1. Eigenvector $(1, 1, \ldots, 1)^T$:

$$\mathbf{C} \cdot (1, 1, \ldots, 1)^T = M\lambda(1, 1, \ldots, 1)^T$$

This vector has an eigenvalue $M\lambda$.

2. Other Eigenvectors: Any vector orthogonal to $(1, 1, \ldots, 1)^T$ has an eigenvalue λ, due to the structure of $\mathbf{C}$.

(c) PCA Suitability

For this problem: - The first principal component corresponds to $(1, 1, \ldots, 1)^T$ and captures most of the variance. - The remaining $M - 1$ eigenvectors have the same eigenvalue λ, meaning PCA cannot effectively distinguish between them. - Since k is uniformly distributed and each data point has only one non-zero element, PCA will not yield meaningful feature reduction. The data points align with the coordinate axes, making PCA less effective for feature selection.

Conclusion: Due to the uniform distribution and the data structure where only one feature is active at a time, PCA is not suitable for this problem. It cannot provide meaningful variance-based feature extraction or dimension reduction.

Programming Questions

5 Logistic Regression

Model Equation

The linear model is defined as:

$$f_\theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{1}$$

(a). Logistic Function

The logistic (sigmoid) function is:

$$P(\hat{y} = 1 | x_1, x_2) = \sigma(f_\theta(x_1, x_2)) = \frac{1}{1 + \exp(-f_\theta(x_1, x_2))} \tag{2}$$

Initial Parameters

- $\theta_0 = -1$
- $\theta_1 = 1.5$
- $\theta_2 = 0.5$
- Learning Rate: 0.1

Cross-Entropy Error Function

The cross-entropy error function is:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

where m is the number of training samples, y_i are true labels, and $\hat{y}_i$ are predicted probabilities.

(b). Gradient Descent Update

After one iteration, the updated logistic regression model becomes:

$$P(\hat{y}=1|x_1,x_2) = \frac{1}{1+\exp(-(-1.003+1.505x_1+0.502x_2))} \tag{3}$$

(c). Model Performance

After convergence, the model performance on the test dataset is:

Accuracy:	0.667
Precision:	0.600
Recall:	1.000

---

6 House Price Prediction

Introduction

The comprehensive analysis of two machine learning models developed for predicting house sale prices using the Ames Housing dataset. Both models employ advanced feature engineering, preprocessing, and ensemble techniques to improve prediction accuracy.

Model 1: Stacking Ensemble Approach

Preprocessing Pipeline

- **Feature Engineering:** Created composite features
  - `TotalSF`: Total square footage
  - `TotalBathrooms`: Comprehensive bathroom count
  - `TotalPorchSF`: Total porch square footage
  - `HouseAge`: Years since construction
  - `RemodAge`: Years since remodeling
  - `GarageAge`: Years since garage construction
- **Missing Value Handling**

  - Numerical features: Filled with mean values
  - Categorical features: Filled with 'None'
- **Categorical Encoding:** Target Encoding
- **Feature Transformation:** Log transformation for skewed features
- **Feature Scaling:** StandardScaler normalization

Ensemble Composition

Base models in the stacking ensemble:

- Gradient Boosting Regressor
- XGBoost Regressor
- LightGBM Regressor

Meta-model: Ridge Regression

Performance

- Root Mean Squared Error (RMSE): 0.13062
- Stacking approach used 5-fold cross-validation

Model 2: Weighted Ensemble Approach

Preprocessing Enhancements

- Additional feature engineering techniques
- Neighborhood price mapping
- Cyclic encoding of months
- Interaction features
  - `Qual_SF`: Quality * Living Area
  - `Cond_SF`: Condition * Total Square Footage

Feature Selection Strategy

- Correlation-based feature selection
- Threshold of 0.05 for feature correlation with target
- Features with absolute correlation below threshold were dropped

Preprocessing Details

- One-hot encoding for categorical variables
- Mean imputation for numerical features
- Log transformation of target variable

Ensemble Technique

- Weighted ensemble with optimized weights
- Optimization using SLSQP method
- Constrained weight optimization
  - Weights sum to 1
  - Individual weights between 0 and 1

Base Models

- Random Forest Regressor
- Gradient Boosting Regressor
- XGBoost Regressor
- LightGBM Regressor

Performance

- Root Mean Squared Error (RMSE): 0.12803
- Slightly improved performance compared to Model 1

Key Differences and Improvements

Feature Engineering

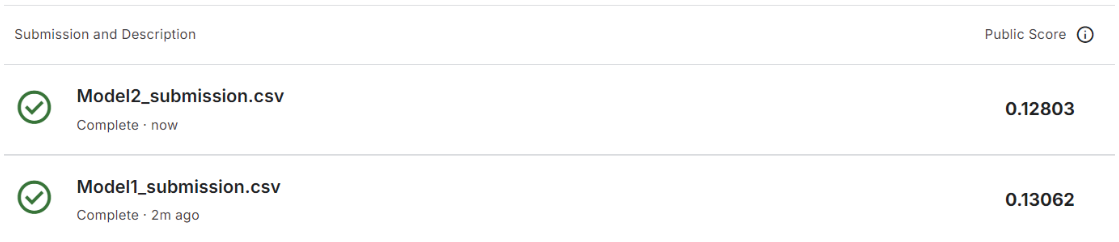
- Model 2 introduced more sophisticated feature interactions
- Cyclic encoding of months
- Neighbourhood price mapping

Feature Selection

- Correlation-based feature pruning
- Removed low-correlation features
- More targeted feature selection

Ensemble Technique

- Model 2 used weighted optimization
- Dynamically learned optimal model weights using scipy.optimize
- More flexible ensemble approach



Submission and Description	Public Score
✓ Model2_submission.csv Complete · now	0.12803
✓ Model1_submission.csv Complete · 2m ago	0.13062

Figure 5: Kaggle Submissions displaying Scores for the two models

Conclusion

The second model demonstrates marginal but significant improvements through advanced feature engineering, more sophisticated feature selection, and a flexible weighted ensemble approach.