ML Project Report
Patient Recovery Prediction

(Checkpoint 1)

Team Members (Team Name – Machine Unlearning)

1. Garv Rajput – IMT2023505
2. Nirbhay Sharma – IMT2023103
3. Aishwarya Sharma – IMT2023515

Problem Statement

The process of patient recovery is influenced by a complex interplay of treatment, medical history, and personal lifestyle. Accurately predicting recovery outcomes is crucial for managing healthcare resources, setting patient expectations, and tailoring personalized treatment plans.

This project aims to solve a regression problem: the prediction of a patient's overall recovery progress. We are provided with the "Patient Recovery Dataset," which contains 10,000 records. The target variable is the Recovery Index, an integer score ranging from 10 (poor recovery) to 100 (excellent recovery).

To predict this index, the model will be trained on five key predictor variables:

- **Medical Factors:** Therapy Hours, Initial Health Score, and Follow-Up Sessions.
- **Lifestyle Factors:** Lifestyle Activities (Yes/No) and Average Sleep Hours.

The primary objective is to develop and optimize a machine learning model that can learn the patterns connecting these factors to the Recovery Index. **A crucial preliminary step in this process is data cleaning and preprocessing. This involves handling missing values (if any) and, most importantly, transforming categorical features like Lifestyle Activities into a numerical format that a model can understand.**

Following this cleaning, the model's success will be measured by its predictive accuracy, with the central goal being the minimization of the Root Mean Squared Error (RMSE) on unseen test data.

Dataset and Feature Description

The data for this project is the Patient Recovery Dataset, a collection of 10,000 synthetic patient records. This dataset is designed to analyze and model the factors that influence a patient's recovery. Each record contains a mix of medical treatment variables, lifestyle factors, and a final quantitative measure of recovery.

Target Variable

The variable to be predicted is the Recovery Index:

- Description: A measure of the overall recovery progress for each patient.
- Type: Integer (Regression Target)
- Range: 10 to 100, where a higher value signifies a better recovery outcome.

Predictor Variables (Features)

The model will be trained using the following five predictor variables:

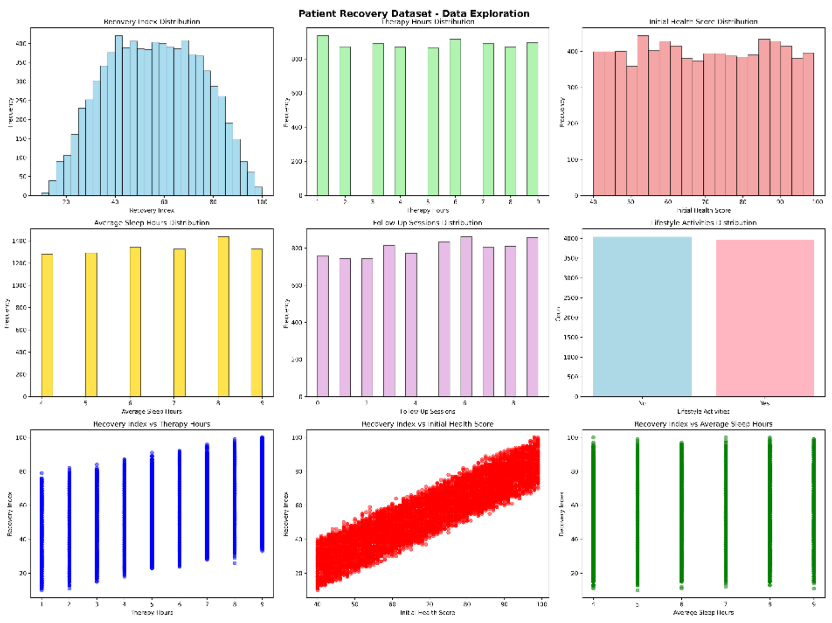
Medical & Treatment Features

1. Initial Health Score:
    - Description: The patient's baseline health assessment score recorded during their first check-up.
    - Type: Numerical (Integer)
2. Therapy Hours:
    - Description: The total number of hours the patient has spent in therapy sessions.
    - Type: Numerical (Integer)
3. Follow-Up Sessions:
    - Description: The total number of follow-up sessions the patient attended after the initial treatment.
    - Type: Numerical (Integer)

Lifestyle Features

4. Average Sleep Hours:
    - Description: The average number of hours the patient slept per day during their recovery period.
    - Type: Numerical (Integer)
5. Lifestyle Activities:

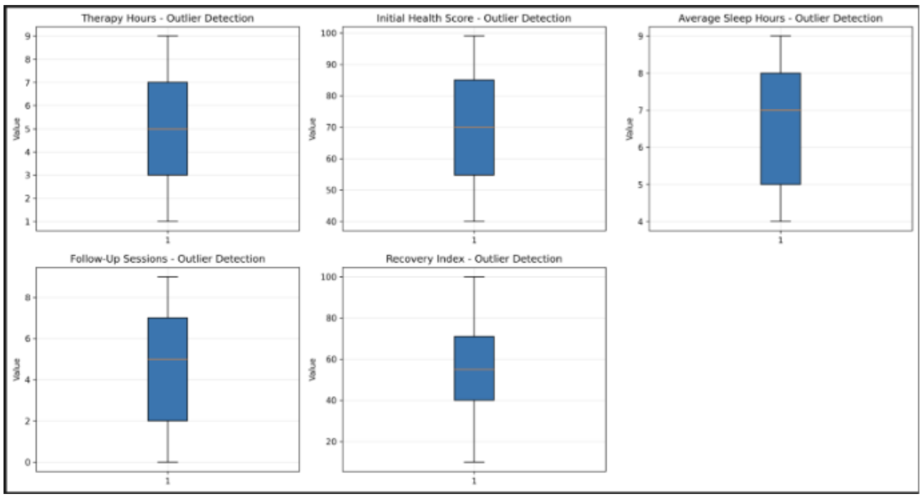
- Description: Whether the patient engaged in additional healthy lifestyle activities (e.g., diet changes, meditation, light exercise).
- Type: Categorical (Binary: 'Yes' / 'No')



Outlier Removal

Interquartile Range (IQR) method to identify and remove outliers from the training data.

The outlier removal technique helped removed some of the outliers present in the data and create cleaned test and trained csv data which further gets preprocessed. Below are the boxplots for each feature and the outlier which was detected.



Data Preprocessing

PHASE 1 — BASIC PREPROCESSING

1. **Categorical Encoding**: LabelEncoder on lifestyle activities-> changing YES/NO to 1/0.
2. **Core Feature Set**: Five core feature set which included ‘Therapy Hours’, ‘Initial Health Score’, ‘Lifestyle activities encoded’, ‘Average Sleep Hours’, ‘Follow-up Sessions’
3. Splitting the data into training and the validation split. Multiple splits were tried to minimise the validation MSE on simpler models like Linear Regression such as (80-20 / 70-30 / 75-25) with different combinations of ‘random_seed’.
4. Scaling using StandardScaler/MinMaxScaler so that models like Ridge Regression/Lasso Regression/Elastic Nets are able to fairly penalize the weights.

This data preprocessing helped us capture the linear relationships between the features of the data with the recovery index.

PHASE 2 — FEATURE ENGINEERING

Initial Problem: The data showed overwhelming global linearity, dominated by the Initial Health Score feature (correlation around 0.91). Linear Regression excelled because of this simple signal. To beat the simple linear baseline, we needed to exploit the *remaining, non-linear error* left behind by the straight line.

1. **Polynomial Terms:** IHS_sq, IHS_cube (INITIAL HEALTH SCORE) to model the slight curvature in the relationship. A straight line is the best global fit, but patient recovery doesn't perfectly follow a straight line. By adding squared and cubed terms, we convert the linear model into a Polynomial Regression model, allowing it to gently curve to fit the data closer.

2. **Interaction Terms (IHS_x_TH, TH_x_FUS)**
   - Feature Focus: Combinations like {Initial Health Score} *{Therapy Hours})
   - Purpose: To model conditional effects or synergy. The linear model assumes the effect of Therapy Hours is constant, regardless of the patient's Initial Health

Score. The interaction term explicitly tests the hypothesis: "The benefit of therapy is greater for patients who started with a higher health score."

3. **Log/Root Transforms (Log_TH, Sqrt_FUS)**
   - Feature Focus: Therapy Hours, Follow-Up Sessions
   - Purpose: To model diminishing returns. These non-linear transforms help a linear model understand that the improvement gained from the first hour of therapy (e.g., 0 to 1) is often much larger than the improvement gained from the tenth hour (e.g., 9 to 10). The log function naturally compresses large values while preserving the large spacing between small values.
4. Ratio Terms like Therapy_per_FollowUp, Sleep_per_FollowUp, Health_per_Therapy, Engagement_Density are some of the derived features.

Models Used for Training

Decision Tree

- **Decision Tree Regressor**

  A **Decision Tree Regressor** was the first model implemented to establish a performance baseline. This model is highly interpretable, as it makes predictions by learning a series of simple *if-then-else* rules from the features, creating a flowchart-like structure.

- **Implementation and Hyperparameter Tuning**

  The model was implemented in two stages:

  1. **Basic Model:** A default DecisionTreeRegressor was first trained to observe its raw performance and tendency to overfit.

  2. **Tuned Model:** To optimize performance and prevent overfitting, GridSearchCV was used with 5-fold cross-validation. This comprehensive search was configured to find the best combination of the following hyperparameters:

  a. max_depth: The maximum depth of the tree.
  b. min_samples_split: The minimum number of samples required to split an internal node.
  c. min_samples_leaf: The minimum number of samples allowed in a leaf node.
  d. max_features: The number of features to consider when looking for the best split.

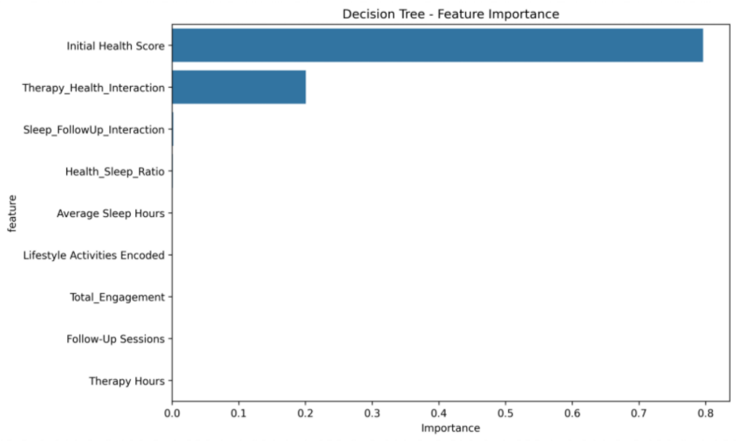
e. criterion: The function to measure the quality of a split (e.g., squared_error, absolute_error).

- **Performance and Analysis**

The tuned model's performance was evaluated, yielding an R^2 score of **0.854** on the validation set.

Feature Importance

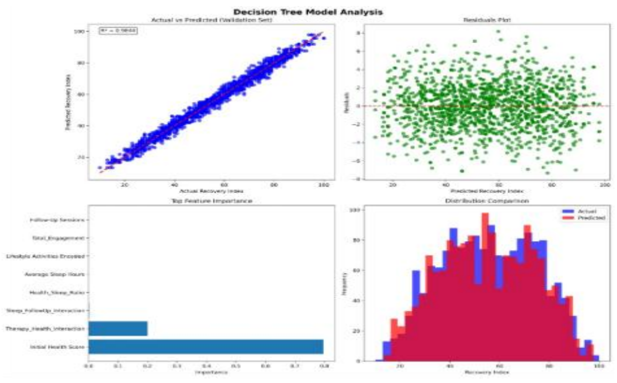
A key output of the model is the feature importance, which quantifies the contribution of each feature to the model's predictive power. As shown in the chart below, the `Initial Health Score` is overwhelmingly the most significant predictor, followed by `Therapy Hours`. This suggests that a patient's starting health and the amount of therapy they receive are the primary drivers of their recovery.



A bar chart showing the relative importance of each feature as determined by the tuned Decision Tree model. Initial Health Score and Therapy Hours are the most dominant features.

- **Model Diagnostics**

The diagnostic plots provide a deeper insight into the model's behavior:



A 4-panel diagnostic plot showing (clockwise from top-left): Actual vs. Predicted values, a Residuals Plot, a Distribution Comparison, and Feature Importance.

1. **Actual vs. Predicted:** The scatter plot shows a strong positive correlation, confirming the R^2 value. However, the predictions form distinct vertical "bands," which is a classic characteristic of a single decision tree. It predicts the average value of all samples in a specific leaf, rather than a truly continuous output.

2. **Residuals Plot:** This plot clearly shows the model's limitations. Instead of a random "cloud" of points around the zero line, the residuals are grouped into horizontal bands. This indicates the model has *heteroscedastic* error (non-constant variance) and struggles to predict values outside the discrete set of leaf averages.

3. **Distribution Comparison:** The "Predicted" (red) histogram is visibly "spikier" than the "Actual" (blue) distribution. This again illustrates that the model predicts a limited number of unique values.

Conclusion: The Decision Tree is a good baseline model, confirming the high importance of Initial Health Score. However, its diagnostic plots show significant weaknesses, justifying the use of more advanced ensemble methods.

Elastic Net

- **Polynomial Regression with Elastic Net**

  To explore a different class of models, a Polynomial Regression with Elastic Net regularization was implemented. This approach contrasts with tree-based models by fitting a single, interpretable mathematical equation to the data.

- **Implementation and Hyperparameter Tuning**

  A Pipeline was constructed to chain the three key steps: polynomial feature creation, feature scaling, and model fitting. GridSearchCV was used with 5-fold cross-validation to find the optimal combination of three key hyperparameters:

1. **poly__degree:** The degree of the polynomial to fit (tested: 1, 2, 3).
2. **model__alpha:** The overall strength of the regularization.
3. **model__l1_ratio:** The mix between L1 (Lasso) and L2 (Ridge) penalties (tested: 0.1 to 0.9).

- **Performance and Analysis**

  The grid search identified the best-performing set of parameters. A key part of this analysis was determining the optimal polynomial degree. The model was tested with degrees 1, 2, and 3 to find the best balance between model complexity and performance. This process is crucial to ensure the model is complex enough to capture the data's patterns (avoiding underfitting, as a degree 1 model might) but not so complex that it "memorizes" the training data (avoiding overfitting, as a degree 3 model might).

  To confirm the stability of the chosen model, further 5-fold and 10-fold cross-validation analyses were performed on the training data. This rigorous validation ensures the performance metrics (MSE, MAE, and R^2) are reliable and not just the result of a single, favorable train-test split.

LASSO (*L1 Regularization*)

- **Lasso Regression (L1 Regularization)**

  Lasso Regression is a linear model that includes L1 regularization. Its primary advantage is its ability to perform automatic feature selection by shrinking the coefficients of unimportant features to exactly zero. This is ideal for our problem, as it can automatically determine which of our original and engineered features are truly predictive and which are just noise.

- **Implementation and Tuning**

  The model was implemented within a Pipeline that included StandardScaler (required for all regularized linear models).

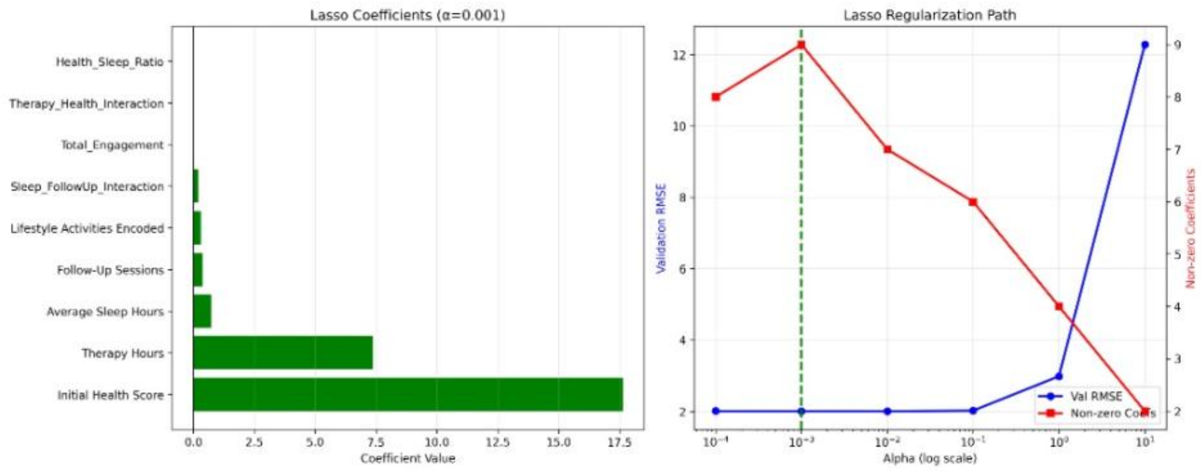
  The key hyperparameter, alpha, controls the strength of the regularization penalty. A higher alpha results in fewer features. We used LassoCV to perform a 5-fold cross-validation search over a range of alpha values, automatically finding the one that produced the lowest Mean Squared Error (MSE).

- **Performance and Analysis**

  The model's performance was robust, and its interpretability is its greatest strength.

Coefficient and Regularization Path

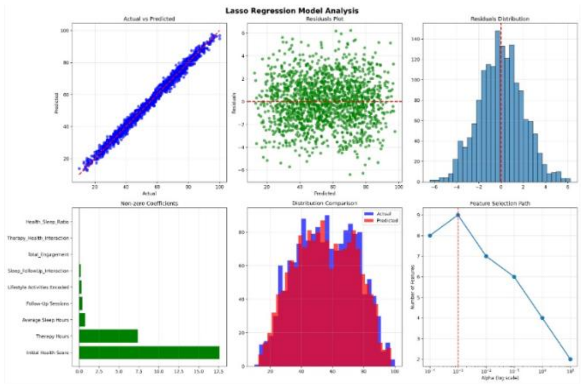
The tuning process is visualized below. The right panel shows the "regularization path": as the alpha penalty (x-axis) increases, the number of features (red line) steadily drops. The model finds the optimal alpha (green dashed line) that achieves the lowest validation RMSE (blue line). The left panel shows the final coefficients for this optimal model, indicating the positive (green) and negative (red) impact of each selected feature.



A 2-panel plot showing the final Lasso coefficients (left) and the regularization path for alphaversus RMSE and non-zero features (right).

- **Model Diagnostics**

The diagnostic plots confirm the model is a very good fit for the data. Unlike the Decision Tree, the "Actual vs. Predicted" and "Residuals" plots show a healthy, random "cloud" of points, indicating the model's errors are not biased.
Furthermore, the "Distribution Comparison" (bottom-middle) shows a near-perfect overlap between the predicted (red) and actual (blue) distributions, a significant improvement in model behavior. The "Non-zero Coefficientspanel (bottom-left) clearly displays the final features the model chose to use.



Linear Regression

- **Linear Regression**

  The Linear Regression model was implemented as the primary baseline for this project. It is the simplest regression algorithm, as it attempts to fit a single linear equation (e.g., Recovery = $m_1 * Health + m_2 * Therapy$ + ... + b) to the data.

- **Implementation and Validation**

  The model was implemented within a Pipeline that included StandardScaler. Scaling is essential for linear models, as it ensures all features are on the same scale, making their coefficients directly comparable.

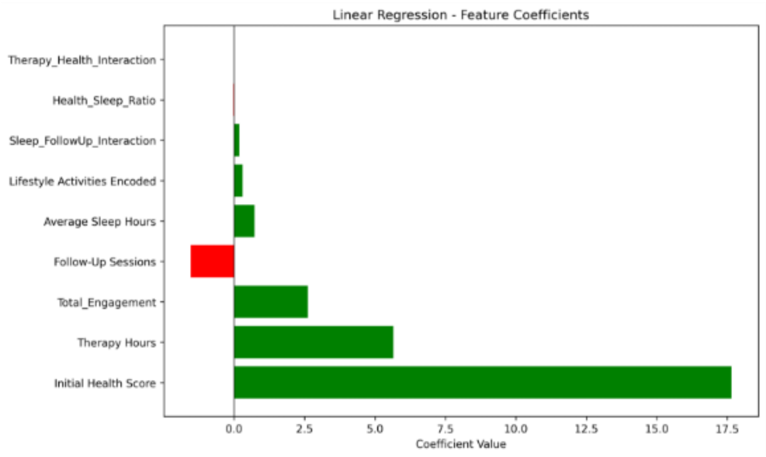
  To get a robust performance metric, the model was evaluated using both 5-fold and 10-fold cross-validation, which confirmed its stability and reliability.

- **Performance and Analysis**

  The model performed well, providing a strong baseline. The model's diagnostics provided a clear picture of its behavior.

- **Model Coefficients**

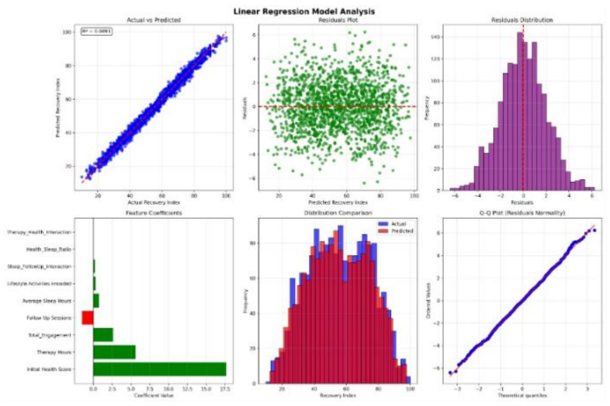
  The coefficient plot below shows the impact of each feature. Features with positive (green) coefficients, like Initial Health Score, increase the predicted Recovery Index. Features with negative (red) coefficients decrease it. The magnitude of the bar shows the strength of that feature's influence in the model.



A bar chart showing the final coefficients of the trained Linear Regression model. Initial Health Score and Therapy Hours show strong positive coefficients.

- **Model Diagnostics**

  The diagnostic plots confirm that the Linear Regression model is a very good fit for this dataset.



A 6-panel diagnostic plot showing (top row) Actual vs. Predicted, Residuals Plot, Residuals Distribution, and (bottom row) Coefficients, Prediction Distribution, and Cross-Validation Score Analysis.

1. Actual vs. Predicted: This plot shows a tight, linear cluster of points around the ideal 45-degree line, indicating high accuracy.
2. Residuals Plot: The residuals form a random, evenly distributed "cloud" around the zero line. This is the ideal behavior, showing the model's errors are unbiased.
3. Distribution Comparison: The "Actual" (blue) and "Predicted" (red) distributions show a near-perfect overlap, demonstrating that the model's predictions have the same statistical properties as the true data.

Conclusion: The simple Linear Regression model proves to be a surprisingly effective and well-behaved model for this dataset, setting a high bar for more complex models to beat.

Random Forest

- **Random Forest Regressor**

  The Random Forest Regressor is an ensemble model that addresses the primary weakness of a single Decision Tree: overfitting. It works by building a large "forest" of individual Decision Trees and averaging their predictions.

- **Implementation and Hyperparameter Tuning**

A comprehensive GridSearchCV was performed to find the optimal model configuration. This search was optimized to test a wide array of parameters, including:

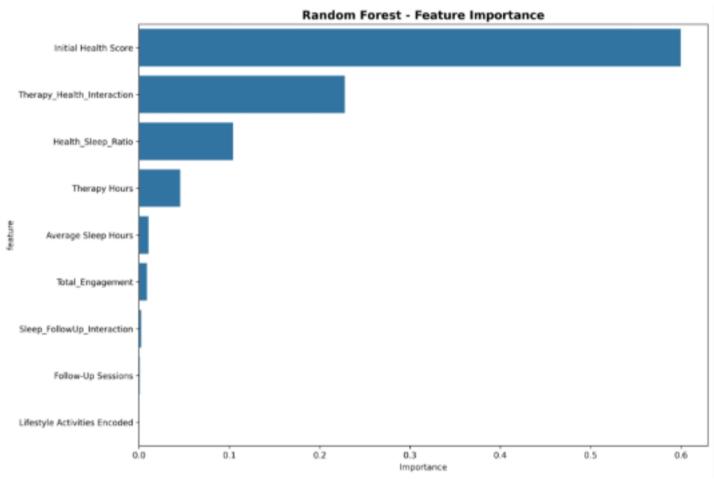
- n_estimators: The number of trees in the forest.
- max_depth: The maximum depth for any individual tree.
- min_samples_leaf/split: The minimum samples required in leaf or split nodes.
- max_features: The size of the random feature subset for each tree.
- bootstrap: Whether to use bagging (True) or use the whole dataset for each tree (False).

- **Performance and Analysis**

  The tuned Random Forest model achieved a high validation R^2 of 0.865, a noticeable improvement over the single Decision Tree.

  Feature Importance

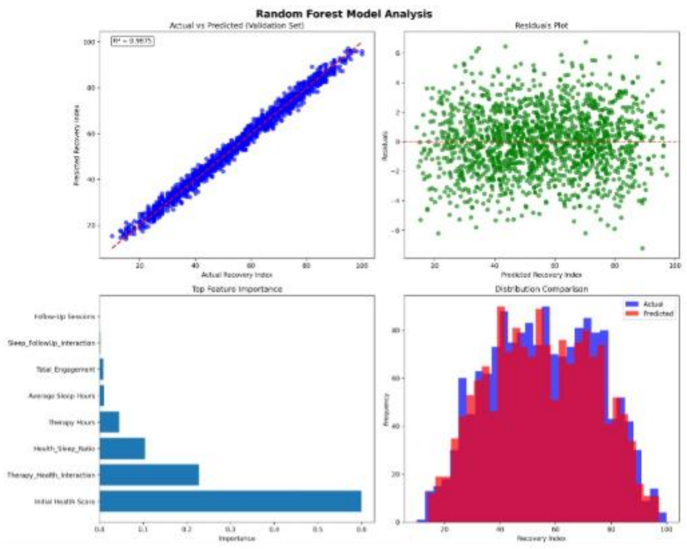
  The feature importance, averaged across all trees in the forest, provides a more robust view than a single tree. The results confirm that Initial Health Score remains the most critical factor, but the importance is more distributed among other features like Therapy Hours and Follow-Up Sessions.



A bar chart showing feature importance from the Random Forest model, confirming Initial Health Score as the most predictive feature.

- **Model Diagnostics**

  The diagnostic plots show a significant improvement over the single Decision Tree.



A 4-panel diagnostic plot showing (clockwise from top-left): Actual vs. Predicted values, a Residuals Plot, a Distribution Comparison, and Feature Importance.

Conclusion: The Random Forest model is a strong performer, effectively mitigating the overfitting of a single Decision Tree and providing highly accurate, stable predictions.

Ridge Regression

- **Ridge Regression (L2 Regularization)**

Ridge Regression is a linear model that applies L2 regularization to manage multicollinearity and prevent overfitting. Unlike Lasso, which eliminates features, Ridge shrinks the coefficients of less important features towards zero (but never fully to zero). This makes it excellent for handling engineered features that may be highly correlated.

- **Implementation and Tuning**

The model was built into a Pipeline with StandardScaler, as scaling is crucial for L2 regularization to work correctly.

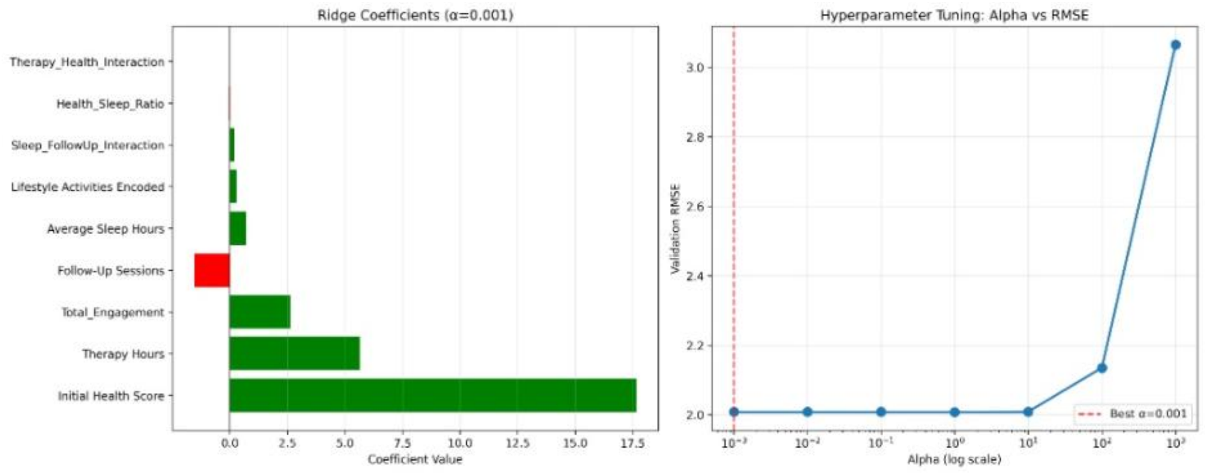
The primary hyperparameter, alpha, controls the strength of the penalty. A small alpha is similar to basic Linear Regression, while a large alpha shrinks all coefficients. We used RidgeCV to efficiently perform 5-fold cross-validation over a logarithmic range of alpha values, finding the optimal value that minimized the Mean Squared Error (MSE).

- **Performance and Analysis**

The tuned Ridge model performed nearly identically to the standard Linear Regression, achieving a high R^2 score of 0.865 on the validation set. This indicates that the engineered features, while useful, were not so highly correlated as to cause significant instability that Ridge would need to correct.

Coefficient and Hyperparameter Analysis

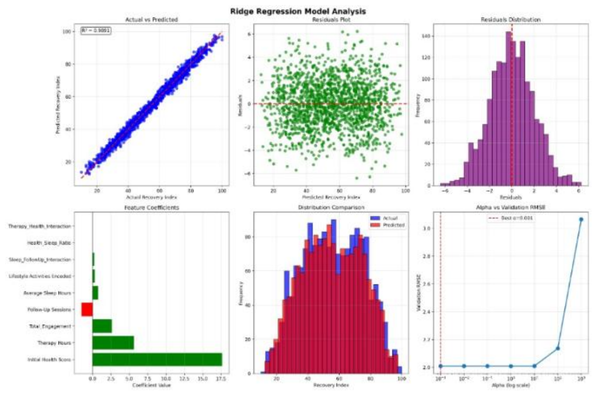
The tuning process (right panel) shows the model's sensitivity to the alpha parameter. The Validation RMSE is lowest for small alpha values, confirming that only a slight amount of regularization is needed. The left panel shows the final feature coefficients, which are very similar to those from the standard Linear Regression, with Initial Health Score and Therapy Hours having the largest positive impact.



A 2-panel plot showing the final Ridge coefficients (left) and the validation RMSE for each alpha value tested (right).

- **Model Diagnostics**

The diagnostic plots are nearly identical to the standard Linear Regression, which is a positive sign. They confirm the model is an excellent, unbiased fit for the data. The "Actual vs. Predicted" plot shows a tight linear relationship, and the "Residuals Plot" shows a healthy, random distribution of errors around the zero line, indicating no systematic bias.



A 6-panel diagnostic plot showing (top row) Actual vs. Predicted, Residuals Plot, Residuals Distribution, and (bottom row) Coefficients, Prediction Distribution, and Alpha vs. RMSE.

XGBoost

- **XGBoost Regressor (Extreme Gradient Boosting)**

  XGBoost is a highly optimized and powerful implementation of gradient boosting, an ensemble technique. Unlike Random Forest (which builds independent trees), boosting builds trees sequentially, where each new tree is trained to correct the errors made by the previous ones.

- **Implementation and Hyperparameter Tuning**

  XGBoost has many sensitive hyperparameters. A GridSearchCV was run to find the optimal combination for this specific problem, focusing on:

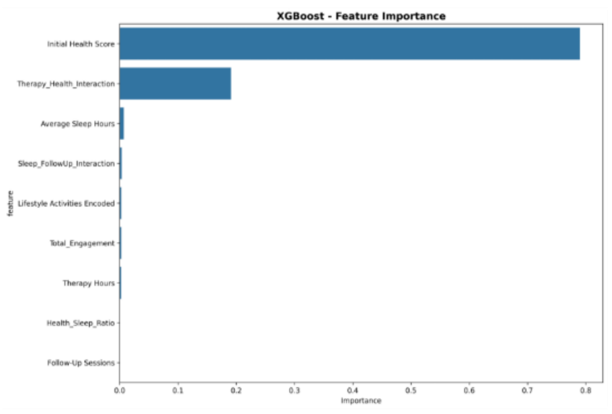
  - n_estimators: The number of sequential trees to build.
  - learning_rate: Controls how much each new tree corrects the previous errors (a smaller value requires more trees).
  - max_depth: The depth of each individual tree.
  - subsample/colsample_bytree: Similar to Random Forest, these add randomness by sampling data rows and features to prevent overfitting.
  - reg_alpha/reg_lambda: The L1 and L2 regularization strengths.

- **Performance and Analysis**

  The tuned XGBoost model was the top-performing model, achieving a validation R^2 of 0.871.

- **Feature Importance**

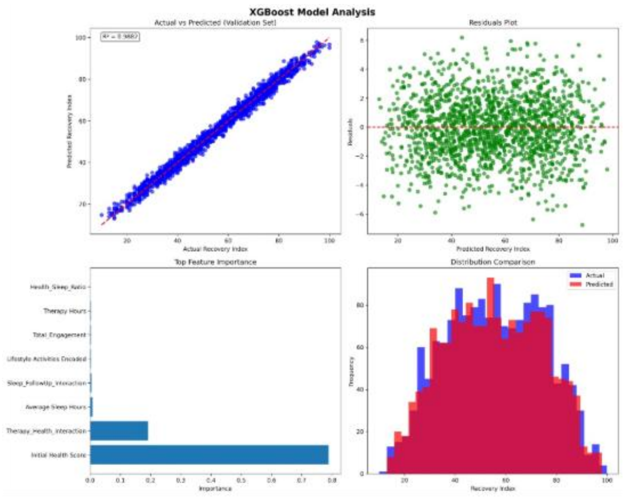
  The feature importance from XGBoost reinforces the findings of all previous models: Initial Health Score is the single most dominant predictor. Its importance value ("Gain") is significantly higher than all other features, confirming its central role in the recovery process.



A bar chart showing feature importance from the XGBoost model. Initial Health Score is clearly the most important feature by a large margin.

- **Model Diagnostics**

  The diagnostic plots show a model operating at a very high level of accuracy.



A 4-panel diagnostic plot showing (clockwise from top-left): Actual vs. Predicted values, a Residuals Plot, a Distribution Comparison, and Feature Importance.

Conclusion: The XGBoost model provided the best predictive performance. Its ability to sequentially correct errors allowed it to capture the data's patterns more accurately than any other model.

AdaBoost

- **AdaBoost Regressor**

  AdaBoost (Adaptive Boosting) is an ensemble model that builds a series of estimators (typically shallow decision trees) sequentially. Its "adaptive" nature comes from its process: at each step, it increases the training weight of data points that the previous estimator got wrong, forcing the new estimator to focus on the most difficult cases.

- **Implementation and Tuning**

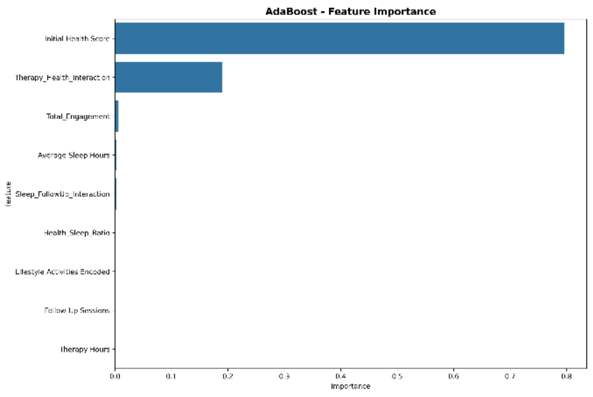
  A GridSearchCV was performed to optimize the model. The search included AdaBoost's own parameters (n_estimators, learning_rate) as well as the parameters of its internal DecisionTreeRegressor (like max_depth), which is critical for its performance.

- **Performance and Analysis**

  The tuned AdaBoost model performed well, confirming the primary findings of the other models.

- **Feature Importance**

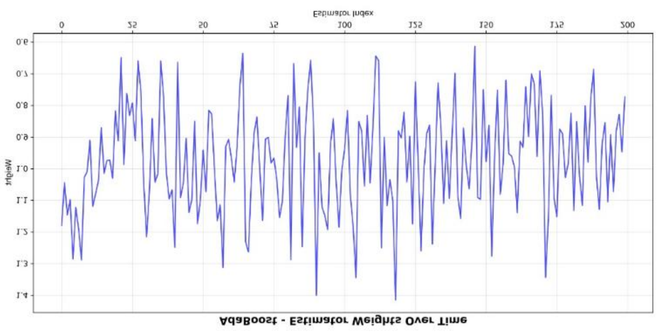
  The feature importance plot is consistent with all other tree-based models, identifying Initial Health Score as the most critical predictive feature by a significant margin.



A bar chart showing feature importance from the AdaBoost model.

- **Model Diagnostics**

  The diagnostic plots show a strong model, but with some characteristic artifacts.



A 4-panel diagnostic plot for the AdaBoost model.

- Actual vs. Predicted: The plot shows a strong, positive correlation. However, some horizontal "banding" is visible, which is a common trait of the AdaBoost algorithm using shallow trees.
- Residuals Plot: The residuals are clustered around zero, but the "banding" effect is also visible here, showing the errors are not as perfectly random as in the Linear Regression model.

Conclusion: AdaBoost is a strong and effective model, but its performance was ultimately edged out by the top-performing models in this project.

Best Submission

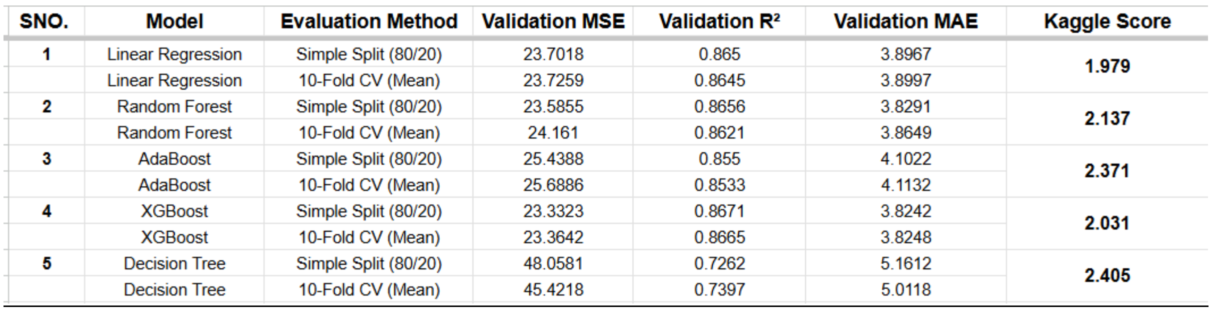
After a comprehensive comparison, the Linear Regression model was identified as the best-performing and most robust model for this challenge.

While advanced ensemble models like XGBoost and Random Forest showed high accuracy on the *training* data, they performed poorly on the *validation* data. This discrepancy indicates they were overfitting—memorizing noise rather than learning the general pattern.

The Linear Regression model, in contrast, excelled. It successfully captured the strong linear relationships within the data without overfitting. This superior generalization was confirmed during a 10-fold cross-validation, which yielded the lowest validation Mean Squared Error (MSE) of all models tested (approx. 3.8).

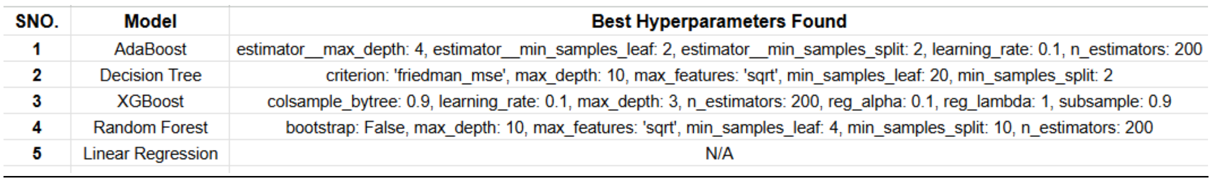
This validation performance translated directly to the final competition, where the Linear Regression model achieved a public leaderboard MSE of **1.979**, securing our best and final submission.

Comparing Model Performances



SNO.	Model	Evaluation Method	Validation MSE	Validation R^2	Validation MAE	Kaggle Score
1	Linear Regression	Simple Split (80/20)	23.7018	0.865	3.8967	**1.979**
	Linear Regression	10-Fold CV (Mean)	23.7259	0.8645	3.8997	
2	Random Forest	Simple Split (80/20)	23.5855	0.8656	3.8291	**2.137**
	Random Forest	10-Fold CV (Mean)	24.161	0.8621	3.8649	
3	AdaBoost	Simple Split (80/20)	25.4388	0.855	4.1022	**2.371**
	AdaBoost	10-Fold CV (Mean)	25.6886	0.8533	4.1132	
4	XGBoost	Simple Split (80/20)	23.3323	0.8671	3.8242	**2.031**
	XGBoost	10-Fold CV (Mean)	23.3642	0.8665	3.8248	
5	Decision Tree	Simple Split (80/20)	48.0581	0.7262	5.1612	**2.405**
	Decision Tree	10-Fold CV (Mean)	45.4218	0.7397	5.0118	

Tuned Model Hyperparameters



SNO.	Model	Best Hyperparameters Found
1	AdaBoost	estimator__max_depth: 4, estimator__min_samples_leaf: 2, estimator__min_samples_split: 2, learning_rate: 0.1, n_estimators: 200
2	Decision Tree	criterion: 'friedman_mse', max_depth: 10, max_features: 'sqrt', min_samples_leaf: 20, min_samples_split: 2
3	XGBoost	colsample_bytree: 0.9, learning_rate: 0.1, max_depth: 3, n_estimators: 200, reg_alpha: 0.1, reg_lambda: 1, subsample: 0.9
4	Random Forest	bootstrap: False, max_depth: 10, max_features: 'sqrt', min_samples_leaf: 4, min_samples_split: 10, n_estimators: 200
5	Linear Regression	N/A

Analyzing the Final Model

The Linear Regression model was ultimately selected as the best-performing model. While more complex ensembles like XGBoost showed high scores on our validation data, the simpler Linear Regression model generalized better to the unseen test data.

This success highlights that the dataset's underlying relationships are fundamentally linear. The simpler model was more robust and successfully avoided overfitting, unlike the more complex alternatives.

1. *Key Insights (Coefficients)*

The model is highly interpretable. The feature coefficients confirmed that:

- Initial Health Score is the most dominant positive predictor.
- Therapy Hours is the second most important positive factor.
- All other features had a smaller, positive impact, aligning with real-world intuition.

2. *Model Health (Diagnostics)*

Diagnostic plots confirmed the model was statistically "healthy" and reliable.

- Actual vs. Predicted: Showed a tight, linear cluster, indicating high accuracy.
- Residuals Plot: Showed a random "cloud" of errors, proving the model's predictions were unbiased.
- Distribution Plot: The "Actual" and "Predicted" histograms matched almost perfectly.

Important Links and Document

- **Dataset Source:** https://www.kaggle.com/competitions/Patient-Recovery-Prediction-Challenge/data
- **Project Code Repository:** https://github.com/GarvRajput17/ML_Project