# Patient Recovery Index Prediction

ML Project Report

1.Archit Jaju (IMT2023128)
2.Sanyam Verma (IMT2023040)

`github.com/ayhm23/Patient_Recovery_Prediction-ML`

October 26, 2025

## 1 Task

The objective of this project is to develop and evaluate machine learning models that can estimate a patient's recovery progress based on their treatment and lifestyle factors. This is a supervised regression problem where the target variable, `Recovery Index`, is a continuous score ranging from 10 to 100, indicating the extent of a patient's recovery.

The task involved:

- Performing exploratory data analysis (EDA) and preprocessing.
- Engineering new features to capture complex interactions and non-linear relationships.
- Training and systematically comparing multiple regression models.
- Tuning the final model using robust 10-fold cross-validation and submitting its predictions to Kaggle.

## 2 Dataset and Feature Description

The dataset consists of 10,000 patient records (8,000 for training, 2,000 for testing), each containing both clinical and behavioral predictors of recovery. The key variables are as follows:

- **Therapy Hours:** The total number of hours a patient spent in therapy sessions.
- **Initial Health Score:** The patient's health assessment score recorded during their first check-up.
- **Lifestyle Activities:** Whether the patient engaged in additional healthy lifestyle activities (Yes or No).
- **Average Sleep Hours:** The average number of hours the patient slept per day.
- **Follow-Up Sessions:** The number of follow-up sessions the patient attended.

The target variable, **Recovery Index**, is a measure of the overall recovery progress of each patient, with higher values indicating better recovery outcomes.

## 3 EDA and Preprocessing

### 3.1 Data Cleaning

The training and test datasets were first inspected for missing or duplicate values. The datasets were found to be complete, so no imputation was required.

The `Lifestyle Activities` column was converted from its categorical form (Yes/No) into a numeric form using `OneHotEncoder` to be compatible with all model types.

The `Id` column was removed from both datasets as it is not a predictive feature.

### 3.2 Base Feature Scaling

For models sensitive to feature magnitude (Linear, Ridge, Lasso, SVR, KNN), features were scaled using `StandardScaler`. This was integrated as the first step within a `Pipeline` to prevent data leakage during cross-validation. Tree-based models did not require scaling.

### 3.3 Feature Engineering

The project involved two main stages of feature engineering:

1. **Initial Feature Engineering (11 Features Total):** Based on preliminary EDA, 6 interaction and summation features were created from the original 5 base features: `Therapy_x_Health`, `Therapy_plus_Health`, `Sleep_x_Health`, `FollowUp_x_Health`, `Therapy_x_Sleep`, `Sleep_plus_Health`. This intermediate 11-feature set was used for the initial detailed model comparison described in Section 4.1.

2. **Advanced Feature Engineering (15 Features Total):** To further capture non-linearities, 4 additional features (ratios and logs) were created: `Therapy_Health_ratio`, `Sleep_Health_ratio`, `log_Therapy`, `log_Health`. This final 15-feature set (11 previous + 4 new) was used for the refined model selection and final submission (Section 4.2 onwards).

Rationale: The hypothesis was that the relationship between features was not purely additive or linear. Interaction terms model synergistic effects, summation terms capture combined impacts, ratios model proportional relationships, and logarithmic terms handle potential skewed distributions and diminishing returns.

Preprocessing Pipeline: For both feature sets, a `ColumnTransformer` was used to apply `StandardScaler` to all numerical features and `OneHotEncoder` to the categorical `Lifestyle Activities` feature.

## 4 Models Used for Training

The model selection process involved several stages:

### 4.1 Initial Model Comparison (11 Features)

The first major phase involved training and evaluating a wide range of regressors on the initial 11-feature set (5 base + 6 engineered) to understand the data's characteristics and establish performance baselines. The goal was to cast a wide net and identify promising model families.

- **Linear Regularized Models:**
  - *Ridge (L2):* Included for its effectiveness in handling potential multicollinearity introduced by engineered features and providing a robust baseline.
  - *Lasso (L1):* Tested for its ability to perform automatic feature selection by shrinking irrelevant coefficients to zero.

  - *Others Evaluated:* Linear Regression (unregularized), ElasticNet (balancing L1/L2), Bayesian Ridge (probabilistic approach).

- **Tree-Based Ensembles:**
  - *Random Forest Regressor:* Used to capture potential non-linear interactions missed by linear models, with bagging to reduce variance compared to single Decision Trees.

- **Boosting Ensembles:**
  - *XGBoost Regressor:* An efficient, regularized implementation of gradient boosting, often a top performer.
  - *Gradient Boosting Regressor:* Standard gradient boosting implementation, useful for comparison.
  - *Others Evaluated:* AdaBoost (focusing on hard-to-predict samples), LightGBM (fast gradient boosting).

- **Other Model Types:**
  - *Support Vector Regressor (SVR):* Explored with both linear and non-linear (RBF) kernels to capture different relationship patterns.
  - *K-Nearest Neighbors (KNN):* A non-parametric approach tested for capturing local relationships.
  - *Decision Tree:* Single tree evaluated as a simple non-linear baseline.

Results: Cross-validation consistently showed that **Ridge Regression** achieved the best performance (RMSE $\sim$2.0435) among all models tested on this 11-feature set. Lasso performed similarly, while Random Forest (RMSE $\sim$2.21), XGBoost, and Gradient Boosting (RMSE $\sim$2.09) performed worse. This indicated that the underlying relationship remained strongly linear even with the initial engineered features, and that more complex models were overfitting or failing to find additional signal. This solidified the decision to focus on refining linear models with more advanced feature engineering.

### 4.2 Refined Model Selection (15 Features)

Based on the success of Ridge in the initial phase and the hypothesis that further feature engineering could capture remaining non-linearities for a linear model, the focus shifted. The 4 additional ratio and log features were added (creating the final 15-feature set), and the primary models re-evaluated were:

- **Ridge Regression (L2):** Re-trained on the 15 features as the new baseline, leveraging its proven stability.
- **ElasticNet:** Selected as the final model candidate specifically for its ability to balance L1 (feature selection for potentially redundant features) and L2 (stability for correlated features) regularization, making it well-suited for the richer, potentially more correlated 15-feature set.

Random Forest was also briefly re-tested on the 15 features but continued to perform significantly worse than the linear models.

## 5 Hyperparameter Tuning

All models underwent hyperparameter tuning using K-fold cross-validation (typically 5 or 10 folds) to optimize performance, primarily using `GridSearchCV` or specialized CV estimators like `ElasticNetCV`. The scoring metric was consistently `neg_root_mean_squared_error`.

- **Ridge Regression (11 & 15 Features):** Tuned the `alpha` (regularization strength) parameter. For the 11-feature stage, a grid search across values like $[0.1, 1.0, 10.0, 20.0, 50.0, 100.0]$ identified $\alpha \approx 22.5$ as optimal. For the 15-feature stage, a similar search found $\alpha = 1.0$ to be best.
- **Lasso (11 Features):** Tuned the `alpha` parameter using a grid search over a logarithmic scale to find the optimal sparsity level.
- **Random Forest (11 Features):** Tuned key tree parameters using `GridSearchCV`:
  - `n_estimators`: Number of trees (e.g., $[50, 100, 150]$).
  - `max_depth`: Maximum depth of each tree (e.g., $[10, 20, \text{None}]$).
  - `min_samples_leaf`: Minimum samples per leaf node (e.g., $[1, 5, 10]$).
- **Gradient Boosting & XGBoost (11 Features):** Tuned parameters such as:
  - `n_estimators`: Number of boosting stages/trees.
  - `learning_rate`: Step size shrinkage.
  - `max_depth`: Maximum depth of individual trees.
  - (XGBoost specific regularization: `reg_alpha`, `reg_lambda`).
- **ElasticNetCV (15 Features):** This model automatically performed an efficient internal cross-validation search over a predefined grid during the `.fit()` call. The search space included:
  - **Alphas:** 60 values automatically generated on a log-scale within the model, covering a wide range (equivalent to searching `np.logspace(-5, 1, 60)`).
  - **L1 Ratios:** Explicitly provided list `[0.1, 0.3, 0.5, 0.7, 0.85, 0.9]` to test various blends of L1 and L2 penalties.
  - **Cross-Validation Strategy:** Employed a 10-fold `StratifiedKFold` (binned on 5 quantiles of `y`) internally to ensure robust evaluation across different target value ranges.

  The best combination ($\alpha \approx 0.0028$, `l1_ratio=0.9`) was automatically selected by the estimator.

## 6 Performance Discussion

### 6.1 Model Progression

The modeling process was iterative, improving performance at each stage:

1. **Initial 11-Feature Baseline:** Models were trained on the 5 base + 6 engineered features. Ridge Regression emerged as the best performer with a CV RMSE ~2.0435, achieving a Kaggle score of **2.066**. This confirmed the data's strong linear nature.
2. **Advanced Feature Engineering (15 Features):** Adding 4 more features (ratios, logs) significantly improved the Ridge model's Kaggle score to **1.982**. This validated the benefit of capturing more subtle non-linear effects for the linear model.
3. **Final Model Refinement:** Replacing Ridge with `ElasticNetCV` on the 15-feature set allowed for optimal L1/L2 regularization, providing both feature selection and stability. This final model achieved the best Kaggle score of **1.980**.

### 6.2 Validation Strategy and OOF Score

To get an unbiased estimate of the final model's performance, an Out-of-Fold (OOF) prediction set was generated using `cross_val_predict` with the final ElasticNet model and the 10-fold StratifiedKFold strategy. The RMSE on these OOF predictions was **2.0449**. This score is considered a very reliable estimate of how the model will perform on unseen data.

### 6.3 Final Model Comparison

Table 1: Model Performance Progression Summary

| Stage | Model | Best RMSE (CV/OOF) | Kaggle Score |
|---|---|---|---|
| *Initial (11 Features) Baseline* | | | |
| | Ridge | ~2.095 | - |
| | Lasso | ~2.095 | - |
| | ElasticNet | ~2.096 | - |
| | XGBoost | ~2.139 | - |
| | Gradient Boosting | ~2.143 | - |
| | Random Forest | ~2.205 | - |
| *Refined (15 Features)* | Ridge | ~2.0444 | 1.982 |
| *Refined (15 Features)* | **ElasticNet** | **~2.0449 (OOF)** | **1.980** |

### 6.4 Conclusion on Best Model

The final model selection process clearly demonstrates the value of careful feature engineering combined with appropriate regularization. The initial baseline evaluation on 11 features established that linear models, particularly Ridge (Kaggle score **2.066**), outperformed complex non-linear models for this dataset.

The most significant performance gain came from **advanced feature engineering** (expanding to 15 features including ratios and logs), which improved the Ridge score to **1.982**. This confirmed the hypothesis that even a predominantly linear relationship can benefit from features designed to capture subtle interaction and non-linear effects.

The final optimization step, switching from Ridge to **ElasticNet** (Kaggle score **1.980**), provided a further refinement. ElasticNet's combination of L1 and L2 penalties proved ideal for the 15-feature set:

- The dominant **L1 component** (`l1_ratio=0.9`) performed effective feature selection, shrinking several potentially irrelevant or noisy engineered feature coefficients (like `Average Sleep Hours`, `log_Therapy`) to exactly zero, simplifying the model.
- The small **L2 component** added stability, particularly useful for handling the multicollinearity introduced by correlated engineered features (e.g., `Therapy Hours` and `Therapy_plus_Health`).

Therefore, the final **ElasticNet** model represents the optimal solution identified: leveraging a rich, engineered feature set managed by a regularization strategy that balances feature selection and stability, yielding the best predictive performance.

### 6.5 Feature Importance

The coefficients from the final ElasticNet model highlight the impact of the engineered features. The model automatically performed feature selection, shrinking several coefficients to zero. Table 2 shows the full list of coefficients.

## 7 Final Model and Reproducibility

### 7.1 Final Training and Prediction

The final selected model is the **ElasticNet** ($\alpha = 0.00276$, `l1_ratio=0.9`) trained on the full 15-feature dataset. A `Pipeline` encapsulating the `ColumnTransformer` (scaling, encoding) and the `ElasticNetCV` model was used. Predictions on the test set were generated using this pipeline and clipped to the known target range of $[10, 100]$.

s

Table 2: Full Feature Importances (Coefficients) from Final ElasticNet Model

| Feature | Coefficient |
|---|---|
| Therapy_plus_Health | 7.1272 |
| Initial Health Score | 5.2301 |
| Therapy Hours | 4.1095 |
| Sleep_plus_Health | 3.5845 |
| log_Health | 1.6685 |
| Therapy_x_Health | 1.2643 |
| Therapy_Health_ratio | 1.1996 |
| Follow-Up Sessions | 0.5463 |
| Sleep_x_Health | 0.3717 |
| Lifestyle Activities_No | -0.2991 |
| Lifestyle Activities_Yes | 0.2991 |
| Sleep_Health_ratio | 0.2239 |
| Therapy_x_Sleep | 0.0955 |
| Average Sleep Hours | 0.0000 |
| FollowUp_x_Health | 0.0000 |
| log_Therapy | 0.0000 |

### 7.2 Reproducibility

- All analysis scripts and notebooks are available at the GitHub repository.
- A consistent `random_state=42` was used throughout.
- Scikit-learn `Pipeline` and `ColumnTransformer` objects ensured consistent preprocessing and prevented data leakage. s
- A `StratifiedKFold` (10 splits, binned on `y`) was used for robust cross-validation during tuning and OOF prediction generation.
- The final trained `Pipeline` object was saved as a `.joblib` file, and the final feature coefficients were exported to `elasticnet_coefficients.csv`.