Patient Recovery Prediction Challenge
Team: Shivek Ranjan

Shivek Ranjan (IMT2023042) Navish Malik (IMT2023060)

Digvijaysinh Pawar (IMT2023099)

October 26, 2025

GitHub Repository:
github.com/ImNotFound7/Shivek-Ranjan-Patient-Recovery-Prediction-Challenge.git

1 Introduction

This report documents the methodology and findings for the Patient Recovery Prediction Challenge, a Kaggle competition aimed at forecasting patient recovery outcomes. The goal was to build a machine learning pipeline to accurately predict a patient's **Recovery Index**—a key metric of their progress—based on a set of medical and lifestyle features. The project involved a systematic process of data exploration, feature engineering, model training, and rigorous evaluation to identify the most robust predictive model.

The final selected model, a **Linear Regression**, demonstrated superior performance on the hidden test set, highlighting a crucial lesson in machine learning: complexity does not always yield better results.

2 Data Description and Exploratory Analysis

The dataset consists of 10,000 patient records, with a training set of 8,000 and a public test set of 2,000. Each record contains five predictor variables and one target variable.

Predictor Variables

- **Therapy Hours**: The total number of hours a patient spent in therapy sessions.
- **Initial Health Score**: The patient's health score at their first check-up.
- **Lifestyle Activities**: A binary indicator (Yes/No) of engagement in healthy lifestyle activities.
- **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.

Target Variable

- **Recovery Index**: An integer score from 10 to 100 indicating overall recovery progress.

Initial Findings

- The dataset was remarkably clean, with no missing values requiring imputation.
- The ‘Lifestyle Activities‘ column was the only categorical feature and was converted to a binary (0/1) format.
- The ‘Recovery Index‘ target variable exhibited a near-normal distribution with a mean of 55.31 and a standard deviation of 19.20, making Root Mean Squared Error (RMSE) an appropriate evaluation metric.

3 Data Preprocessing and Feature Engineering

A consistent preprocessing pipeline was established to ensure that all models were trained and evaluated under identical conditions.

- **Encoding**: The ‘Lifestyle Activities‘ feature was mapped to ‘1‘ for "Yes" and ‘0‘ for "No".
- **Scaling**: All numeric features were standardized using ‘StandardScaler‘ to normalize their ranges, which is crucial for linear models and distance-based algorithms.
- **Feature Engineering**: To capture potential non-linear relationships and interactions, the feature set was expanded from 5 to 30. This was achieved by creating:
  - **Interaction Terms**: Products of every pair of numeric features (e.g., ‘Therapy Hours‘ × ‘Initial Health Score‘).
  - **Polynomial Terms**: Squared values of each numeric feature (e.g., ‘Therapy Hours‘^{2}).
  - **Ratio Terms**: Ratios between pairs of numeric features.

4 Modeling and Evaluation

A wide array of regression models was trained and evaluated to benchmark performance comprehensively. The primary metric for evaluation was **Root Mean Squared Error (RMSE)**, calculated using 5-fold cross-validation on the training data and then validated against the public leaderboard score on the test set.

Models Tested

- **Linear Models**: Linear Regression, Ridge, Lasso, ElasticNet, Bayesian Ridge
- **Non-Linear Models**: Polynomial Regression (degrees 2 and 3), Decision Tree
- **Ensemble Models**: XGBoost, AdaBoost, Stacking Ensemble (combining top-performing linear models)

4.1 Cross-Validation Performance

Cross-validation provided a robust estimate of each model's performance on unseen data. The results showed that regularized linear models performed exceptionally well, with Lasso achieving the lowest CV RMSE.

Table 1: Cross-Validation RMSE on Training Data

Model	CV RMSE
Lasso	**2.0478**
Stacking Ensemble	2.0483
ElasticNet	2.0491
Ridge	2.0500
Bayesian Ridge	2.0500
Linear Regression (degree=1)	2.0501
Polynomial LR (degree=2)	2.0829
XGBoost	2.1246
Polynomial LR (degree=3)	2.2220
AdaBoost	2.5597
Decision Tree	2.5973

4.2 Kaggle Leaderboard Performance

The ultimate test of a model's generalization ability is its performance on a completely hidden dataset. When submitted to the Kaggle competition, the simple **Linear Regression (degree=1)** and **Ridge** models achieved the best public score, outperforming more complex models that had slightly better CV scores.

Table 2: Kaggle Public Leaderboard RMSE

Model	Public RMSE
Linear Regression (degree=1)	**1.980**
Ridge	**1.980**
ElasticNet	1.981
Bayesian Ridge	1.981
TOP3 Averaged	1.981
Lasso	1.982
Stacking Ensemble	1.982
Polynomial LR (degree=2)	1.992
XGBoost	2.042
Polynomial LR (degree=3)	2.072
AdaBoost	2.507
Decision Tree	2.518

5 Individual Model Insights

This section provides a detailed breakdown of how each model performed and what those results imply about the underlying data structure.

- **Linear Models (Linear, Ridge, Lasso, ElasticNet)**: This family of models was the clear winner. The standard **Linear Regression** and **Ridge** models achieved the best public RMSE of 1.980. **Lasso** had the best cross-validation score but was slightly edged out on the public leaderboard. This indicates that while some regularization was helpful during CV, the test set's data distribution was so similar to the training set's that an unregularized model generalized perfectly. Their success points to a strong, fundamentally linear relationship between the features and the Recovery Index.
- **Polynomial Regression**: Introducing non-linearity with a degree-2 polynomial provided a reasonable score (1.992) but did not beat the simpler linear models. Increasing complexity further to a degree-3 polynomial resulted in a significantly worse score (2.072), a clear sign of **overfitting**. The model began fitting to noise in the training data rather than the true underlying signal.
- **Tree-Based Models (Decision Tree, AdaBoost, XGBoost)**: These models performed significantly worse than their linear counterparts. **XGBoost** was the best of the three but still could not match the linear models' performance. This poor showing strongly suggests the absence of complex, non-linear interactions between features that tree-based models are designed to capture. The relationships in this dataset are global, not partitioned into local regions.
- **Ensemble Models (Stacking and Averaging)**: Both stacking the top linear models and averaging their predictions yielded very competitive scores (around 1.981-1.982). However, they failed to outperform the single best Linear Regression model. This is a classic example of an ensemble providing stability but not a significant performance boost because the base models were too highly correlated (i.e., not diverse enough).

6 Final Model Selection

Based on its top-ranking performance on the Kaggle public leaderboard, **Linear Regression (degree=1)** was selected as the final model. Its **Public RMSE of 1.980** was the lowest, proving its superior ability to generalize. This model is not only the most accurate but also the most interpretable and computationally efficient, making it an ideal choice.

7 Discussion and Lessons Learned

This project provided several critical insights into practical machine learning.

1. **Simplicity Wins Over Complexity**: The most striking lesson was that a basic Linear Regression model outperformed ensembles like XGBoost and stacking. This underscores the principle that the underlying data-generating process was

fundamentally linear. Adding unnecessary complexity introduced noise and led to overfitting, degrading performance on the test set.

2. **The Value of Cross-Validation and the Test Set**: There was a consistent, small, discrepancy between the cross-validation RMSE (around 2.05) and the public leaderboard RMSE (around 1.98). This suggests that while CV is an essential tool for model tuning and selection, the final decider must be a truly blind test set. The lower test RMSE may indicate that the test data distribution was slightly more uniform or less noisy than the training data. This gap highlights the importance of not overfitting to CV scores alone.

3. **Feature Engineering is a Double-Edged Sword**: While creating interaction and polynomial features is a standard technique to capture non-linearities, it did not prove universally beneficial here. For the linear models, these engineered features provided a slight boost. However, for tree-based models, they did not help, and for higher-degree polynomial regression, they caused catastrophic overfitting, as evidenced by the high RMSE scores.

4. **Ensembling Is Not a Panacea**: Both stacking and averaging top models yielded strong, reliable results. However, they failed to surpass the single best Linear Regression model. This shows that ensembling is most effective when base models are diverse and capture different aspects of the data; in this case, the top linear models were too similar for an ensemble to create significant new value.

8 Conclusion

The Patient Recovery Prediction Challenge was successfully addressed by employing a systematic approach to modeling. The final selection of a **Linear Regression model** was driven by empirical evidence from both cross-validation and leaderboard performance. The project reinforces: a deep understanding of the data, combined with a disciplined and simple modeling approach, often leads to the most robust and accurate solutions.

9 Appendix

Code Structure

- All experiments, from preprocessing to model training and prediction, are contained within the ‘ml-all-models.ipynb‘ notebook for full reproducibility. Predictions are stored in a folder named ”prediction-all-models”.

- Feature engineering, encoding, and scaling are applied consistently across all models for fair comparison.

Requirements

- Python 3.11

- pandas, numpy, scikit-learn, xgboost