# Predicting Medical Costs with Gradient Descent and Machine Learning

Carlos Iván Fonseca Mondragón | A01771689

**ABSTRACT** This report presents the analysis and implementation of a linear regression model for predicting medical costs for an insurance company, based on a dataset with demographic and lifestyle information about its users. The methodology includes data cleaning, categorical variable encoding, outlier treatment, and feature normalization. The model works through an implementation of gradient descent, and its performance is graded using the mean squared error and $R^2$. This approach yields an $R^2$ value of 0.75 on the test set, meaning that it can explain 75% of the variance in the costs. When switching over to a decision tree regression, the model can achieve an $R^2$ of 0.86 in its train set, a 14% increase in predictive capabilities over the manual implementation.

## 1. INTRODUCTION

The rising costs of healthcare have certainly become a factor in the research and development of models that can help estimate the medical expenses of an individual based on demographic and behavioral variables. Accurate estimations not only help insurance companies optimized their pricing behavior, but also support the public health planning sector. With this context, linear regression provides a basis for modelling a relationship between independent variables (such as age, sex, body bass index, smoking status and region) with a healthcare charge.

This study aims to implement and evaluate a linear regression model for predicting medical costs using a publicly available dataset from the book Machine Learning with R by Brett Lantz, that includes individual-level health and demographic information. The regression model currently presented has been constructed using Python, without the use of any machine learning frameworks, providing a manual application of gradient descent in order to optimize the algorithm. The benefit to this method is a granular view into the learning process, error convergence and parameter adjusting.

Before training, the dataset undergoes several preprocessing steps to improve model performance and reliability. These include the transformation of the BMI variable into categorical classifications, encoding of categorical features into numerical format, winsorization of the target variable to mitigate the impact of outliers, and feature scaling via min-max normalization. The results that this implementation achieves a satisfactory level of prediction accuracy on new data, explaining over 70% of the variance in medical charges.

## 2. DESCRIBING THE DATASET

"*Medical Cost Personal Datasets"* is a fictional dataset provided in the book Machine Learning with R by Brett Lantz, it contains 1338 columns with the *age, sex, bmi, children, smoking habits, region, and* a *charge* in the insurance. Next, the columns will be explained in greater detail:

- **age:** The age of the individual, that typically ranges from 18 to 64 years in this dataset. Important because healthcare costs generally increase with age.
- **sex:** The biological sex of the individual: 'male' or 'female'. May correlate with certain medical risks or cost patterns.
- **bmi:** Body Mass Index — a standardized measure of weight relative to height. High BMI often indicates overweight or obesity, which are risk factors for higher medical costs.
- **children:** The number of children covered by the insurance plan.
- **region**: The U.S. geographic region where the person resides: 'northeast', 'northwest', 'southeast', or 'southwest'. Regional cost variations and healthcare access disparities can impact charges.
- **charges:** The total medical insurance charges billed to the individual. This will be the value to predict.

## 3. DATA EXTRACTION

Extraction refers to the first step of the ETL (Extract, Transform, Load) process, where we gather our information. Its objective is to recollect relevant data for transforming and using it for analysis.

### *3.1 Loading the dataset*

Data from the file '*insurance.csv*' was first converted into a **pandas** dataframe, with the *df* name. After analyzing the contents of this dataframe, there was no evidence of missing values anywhere in the file, nor were there any corrupted or incorrect values after a structural review (describe(), info(), and nunique()). After this initial validation, various preprocessing techniques were applied in order to reach a better regression model, as explained below.

### *3.2 Correlation matrix usage*

A correlation map (heatmap) was also generated to identify linear relationships between predictors and the target variable, helping justify the inclusion of all available features without additional elimination.

The values range from -1 (perfect negative linear relationship) through 0 (no linear relationship) to +1 (perfect positive linear relationship).

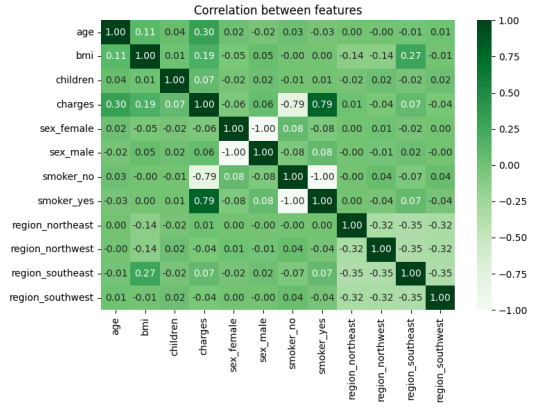


**Figure 1:** Correlation matrix

These are some key observations gathered from the matrix.

- **Smoker status on top**: smoker_yes shows strong positive correlation (~+0.79) with charges, while smoker_no mirrors it at -0.79 (redundant inverse dummies).
- **Age** exhibits moderate positive correlation (~+0.30).
- **BMI** has weak positive correlation (~+0.19).
- **Children** shows very weak correlation (~+0.07) but was retained for potential additive effects.
- **Sex dummies**: sex_female (-0.06) and sex_male (+0.06) are near-perfect inverses, creating multicollinearity.
- **Region dummies** display near-zero correlations (-0.04 to +0.07) but may capture small geographic shifts.

All features were retained to keep preprocessing simple and transparent for manual implementation. Although, we can see that the variable with the highest impact on our dependent variable is **smoker**.

## 4. DATA TRANSFORMATIONS

### *4.1 One-hot encoding*

One-hot encoding converts categorical variables into binary vectors, where each category is represented by a unique vector with one high (1) and the rest low (0).

This transformation was applied to the categorical variables sex, smoker, and region; it avoids introducing a non-existent ordinal relationship (for example, assigning numbers 0,1,2,3) and allows the linear model to interpret each category as an independent indicator. All new dummy columns remain on a 0–1 scale and require no additional normalization.

### *4.2 Rounding*

Global rounding to two decimal places was performed. While this operation is not strictly necessary for linear models and may eliminate small variations, it was maintained to standardize numerical representation and reduce possible minimal fluctuations in continuous variables like bmi or age (if present).

### *4.3 Winsorization*

Winsorization limits extreme values in data by capping them at a specific percentile. It reduces the influence of outliers without removing data points.

The charges variable was, initially, heavily skewed to the right, with several extreme values. To mitigate the influence of extreme outliers in the charges variable, a winsorization technique was applied at the 5th and 95th percentiles, following robust statistics recommendations (Wilcox, 2012; Tukey, 1962). This technique preserves most of the data's structure, while reducing the data's skewness caused by extreme values.

### *4.4 Data shuffling*

The dataset was randomly shuffled (using a randomization seed of **42**) to eliminate any positional patterns, and division into **training (60%), validation (15%)** and **test (25%)** subsets was performed, ensuring an honest evaluation of the manually implemented model's performance.

*4.5 Min-max scaling*

Min-max scaling rescales numeric data to a fixed range, typically [0, 1], preserving relative distances but reducing the impact of magnitude.

Only the numerical columns age and children were normalized (min–max scaling). This decision was based on:

- Reducing magnitude differences between these variables.
- Keeping one-hot columns intact (already binary). The target variable y (charges) was maintained in its original scale to facilitate direct interpretation of metrics.

The formula used in this transformation is the following:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

**Figure 2:** Min-max scaling formula used to normalize the range of a set of data entries.

## 5. MODEL IMPLEMENTATION

*5.1 Features used in the model*

After preprocessing, the model uses age, bmi, children, and one-hot encoded categorical variables (sex_female, sex_male, smoker_yes, smoker_no, region_northeast, region_northwest, region_southeast, region_southwest) as features, with charges as the target variable. Age and children undergo min-max scaling while binary dummy variables remain unchanged. No columns were dropped in the implementation of the model, rather, they were transformed into versions that helped increase the model's accuracy.

*5.2 Model Architecture and Assumptions*

Linear regression with gradient descent was chosen for these two reasons:

- It’s simple and interpretable.
- It shows how much each variable contributes to the final prediction.

This manual linear regression assumes an additive linear relationship between features and charges, with no interactions or non-linear terms. This simplifies learning but limits expressiveness for extremes. Bias is handled as a separate scalar parameter initialized to the target mean. All feature weights (theta) initialize to 0.0. The model uses MSE/2 as the loss function. Training runs for maximum 10,000 epochs with learning rate 0.002, stopping early when cost ≤ 0.01.

The model calculates a weighted sum of all the input variables, plus a **bias**. These weights (or parameters) start at zero and are updated on each epoch (or run of the gradient descent), except for the initial bias, which starts with the mean of the **y** set (values to predict).

*5.3 The hypothesis_theta() function*

The *hypothesis_theta()* function calculates predictions through an explicit dot product

calculation. For each sample, it iterates through all features, multiplying each feature value by its corresponding weight, then adds the bias term.

This function uses the following variables:

- row_values: List of feature values for one sample (e.g., [0.5, 0.8, 2, 1, 0, 1, 0, 0, 1, 0])
- theta: List of learned weights for each feature (e.g., [100, 200, 50, ...])
- bias: Scalar intercept term (initialized to target mean)
- pred: Accumulated dot product of features and weights
- Returns: Final prediction after adding bias

*5.4 The mse() function*

The mse() function (which stands for "Mean Squared Error" or also known as the "loss function") measures the discrepancy between model predictions and actual values, quantifying our model's accuracy for medical charge predictions. This error calculation is fundamental to training, as it provides the metric we seek to minimize.

The division by 2 used in the code simplifies gradient calculations and provides smooth convergence properties for optimization.

The formula to calculate our Mean Squared Error is the following:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

**Figure 3:** MSE formula

This function uses the following variables:

- train_data: DataFrame containing all training features
- theta: Current weight parameters
- bias: Current bias parameter
- y: Series of actual target values
- total: Accumulated sum of squared errors
- rows: Number of training samples
- diff: Residual (prediction error) for each sample
- Returns: Mean squared error divided by 2

Steps taken by the MSE function go as follows:

1. Sample Processing: For each patient record (i = 0, 1, ..., 936), the function processes features like age, BMI, children, and encoded categorical variables for sex, smoking status, and region.
2. Prediction Generation: Calls hypothesis_theta() with current parameters to predict *charges* for each patient.
3. Error Calculation: Calculates the residual (diff = predicted_charge - actual_charge) for each patient.
4. Squared Error Accumulation: Squares each residual and adds to the running total, emphasizing larger prediction errors.
5. Final MSE: Divides accumulated total by (2 × number_of_samples) to obtain the average squared prediction error.

*5.5 Gradient Descent implementation*

The update_weights() function performs batch gradient descent by accumulating gradients across all samples before updating parameters. For each training example, it computes the prediction error and accumulates partial derivatives.

Parameter updates apply the accumulated gradients scaled by the previously defined learning rate. This approach ensures stable convergence by using all the dataset's information for each parameter update.

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

**Figure 4:** Final gradient descent equation

Variables used in this function are as follows:

- rows: Number of training samples for averaging gradients
- features: Number of features (length of theta)
- theta_updated: Copy of current weights for updating
- theta_grad: List accumulating gradients for each feature weight
- bias_grad: Scalar accumulating bias gradient
- row_vals: Feature values for current sample
- err: Prediction error for current sample
- learning_rate: Step size for parameter updates (0.002)
- Returns: Updated theta and bias parameters

There are three main steps in calculating the gradient:

1. Gradient Accumulation: For each patient, the function:
    a. Calculates prediction error (err = predicted_charges - actual_charges)
    b. Accumulates gradients for each feature: theta_grad[j] += err * feature_value[j]
    c. Accumulates bias gradient: bias_grad += err

2. Feature-Specific Updates: Each feature weight is adjusted based on its contribution to prediction errors:
    a. Age/BMI weights: Accumulate errors weighted by normalized age/BMI values
    b. Smoking indicators: Accumulate full errors for smokers, zero for non-smokers
    c. Region indicators: Only accumulate errors for patients in specific regions

3. Parameter Updates: After processing all samples:
    a. Average gradients by dividing by number of samples
    b. Apply learning rate: new_weight = old_weight - learning_rate * average_gradient
    c. Update bias similarly using accumulated bias gradients

This process repeats each epoch, allowing the model to progressively minimize prediction errors on medical charges.

## 6. MODEL EVALUATION

The performance of the model will be graded with the $R^2$ score achieved at the end of an execution. $R^2$ (coefficient of determination) measures the proportion of variance in the target variable that the model explains. Values range from 0 to 1, where 1 indicates perfect prediction and 0 means the model performs no better than simply predicting the mean. It's calculated as **$R^2$ = 1 - (SS_residual / SS_total)**, showing model explanatory power.

The model's predictive accuracy was also assessed using Mean Squared Error (MSE). This value represents the average squared difference between the predicted and actual costs. Although it is generally not easily interpreted by a human, a lower value generally indicates better performance.

### *6.1 Evaluating using the different splits*

For this first implementation, without the use of any machine learning frameworks, the dataset was divided into *train, validation* and *test*, with a 60/15/25 split.

The *train* dataset allows the model to learn and adjust its parameters after each epoch.

The validation dataset serves as a continuous check on model generalization during the 10,000 training epochs. While the training set is used to update model parameters, the validation set provides independent performance feedback without influencing parameter updates. This allows the monitoring of both the training and validation MSE at the same time, ensuring that the models performance stays consistent across data subsets and, at the same time, avoiding an overfit to specific patterns that could result in a lower $R^2$ during the test period.

The *test* dataset, as the name suggest, is used to evaluate whether the model is sufficiently prepared to give an accurate prediction on unseen data. The resulting $R^2$ is a measure of how well the model performed, comparing predictions from the model to the values on the dataset.

## 7. VARIANCE, BIAS, AND MODEL FITNESS

Bias and variance are fundamental sources of prediction error in machine learning models. Bias measures systematic errors that stem from highly simplistic model assumptions that consistently miss the true relationship between features and targets. Variance captures a model's sensitivity to small changes in training data, reflecting how much predictions fluctuate when trained on different samples from the same population.

The bias-variance trade-off is crucial because it determines model performance and generalization ability. High bias leads to underfitting, where models underperform by failing to capture core data patterns. High variance causes overfitting, where models perform excellently on training data but poorly on unknown data due to memorizing noise rather than learning valid relationships. Understanding this helps with crucial choices regarding feature engineering, model selection, and hyperparameter tuning because achieving the best predictive performance requires balancing

these forces rather than minimizing either one alone.

To achieve better predictions, a combination of the following approaches can be used:

- **Bias reduction** by increasing model complexity, adding feature interactions, or using more sophisticated algorithms
- **Variance reduction** through the collection of more data, applying regularization, or using ensemble methods.
- **Reaching an equilibrium point** with cross-validation, learning curves, and experimentation to diagnose whether the model primarily suffers from bias or variance issues.

## 8. MODEL RESULTS

Following model training and comprehensive testing, the results demonstrate that the model achieved performance measured by the coefficient of determination ($R^2$) with the following outcomes:

- Train $R^2$: 0.7716
- Validation $R^2$: 0.8147
- Test $R^2$: 0.7534

```
Final Bias: 10767.223732005454
Train set MSE: 31431930.660498574
Train set R^2: 0.7716529380112371
Test set MSE: 30102108.873223968
Test set R^2: 0.7534475438002389
Validation set MSE: 18978095.536526363
Validation set R^2: 0.8147059693400158
```

**Figure 5**: Model performance metrics for the train, validation and test datasets.

These findings reveal that the model successfully explained approximately 77.16% of variance in the training dataset, 81.47% in the validation dataset, and 75.34% in the test dataset. The converged parameters show a bias term of 10,767.22, with feature weights spanning from -12,920.20 to 10,419.44.

Smoking-related indicators demonstrate the most amount of influence on medical charge predictions, reflecting the impact that this behavior has on healthcare costs. This aligns with medical research showing smokers typically incur higher medical expenses, in particular, Darden (2022) found that smokers typically incur higher medical expenses.

The validation $R^2$ of 0.8147 represents the highest performance across all datasets, which is unusually high compared to both training and test results. This pattern suggests the possibility of validation set contamination, or indicates that the validation set may not be fully representative of the overall data distribution.

The $R^2$ performance between the *train* and *test* (a small difference of 0.018 approximately) shows a reasonable level of consistency for a manual implementation of gradient descent, and a **low level of variance**. This similarity implies the absence of severe overfitting, since these types of models typically exhibit dramatically higher training performance, paired with poor test results.

A level of performance this close across datasets demonstrates that the model is not overly sensitive to training data

peculiarities, implying that the model doesn't tend to memorize noise but, instead, learns general patterns from the dataset.

Despite achieving the aforementioned $R^2$ values, this model's inability to explain more than 77% of the variance in the training data indicates a **medium level of bias**. The assumptions from this particular linear regression model prevent it from explaining more complex and non-linear relationships and feature interactions that are key for an accurate medical cost determination.

A scatter plot was generated to compare predicted and actual charges of all the patients in the test set. Even though the plot shows a good level of alignment between real and predicted values, there is still some divergence in more extreme cases.

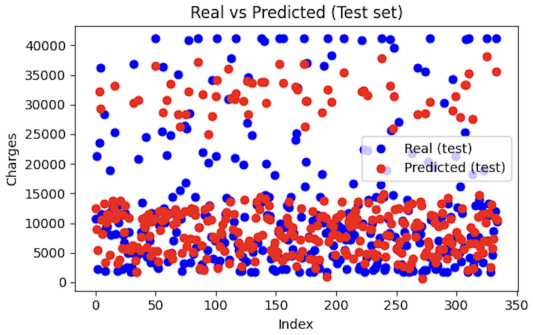


**Figure 6:** Real values plotted against the models predictions.

The cost function was plotted across all epochs to observe convergence behavior. The curve shows a clear decline in the mean square error over time, proving that the learning rate and epochs were set at appropriate values. The training finished at the full 10,000 iterations, since it couldn't reach the early stop criteria of *cost <= 0.01*.

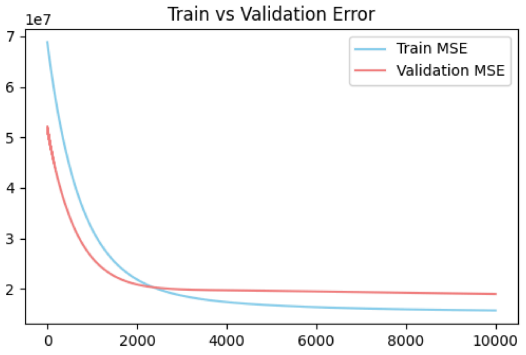


**Figure 7:** Error convergence plot.

These underestimation patterns, particularly in extreme cases, suggest that the model's simplistic linear structure cannot fully represent the actual data complexity.

This combination of medium bias and low variance clearly indicates **underfitting**. Even with good stability and generalization capabilities, this model lacks the complexity required to fully capture the patterns in medical insurance costs, like how age and smoking status might contradict each other's effect in calculating a cost.

The current model would greatly benefit from increased complexity through feature engineering, or more sophisticated algorithms, as will be shown later in this text. At the same time, it proves that, with a good enough data processing and use of optimization basics, even a simple model like gradient descent can yield meaningful insights.

## 9. FRAMEWORK IMPLEMENTATION

The model mentioned from this point, going forward, consists of a regression model based on *Bagging (Bootstrap Aggregation)*, which employs decision trees as base estimators. Compared to the manual implementation of gradient descent seen previously, this approach focuses on more complex, non-linear models, trained with the use of the *scikit-learn* library. All random states will be handled with a seed value of **42**.

*9.1 Opportunity areas*

As seen in the "VARIANCE, BIAS, AND MODEL FITNESS" section, the linear regression model showed a low variance and a medium level of bias, indicating a case of underfitting. In order to achieve a lower bias, without raising excessively the variance, is to employ an ensemble method, such as *Bootstrap Aggregation,* also known as *Bagging*.

Bagging improves the stability and accuracy of weak models (such as decision trees) by training multiple instances of the model on random subsets of the training set, and then averaging their predictions. This method reduces variance without significantly increasing bias.

*9.2 Changes in data preparation*

The general preprocessing remains identical to that described in previous sections of this paper, there is, however, an important exception: **no winsorization or min-max normalization was implemented**, since decision trees do not require variables on the same scale nor are they sensitive to extreme values.

The same predictor variables were used: age, bmi, children, and categorical variables encoded using one-hot encoding for *sex*, *smoker*, and *region*.

The data split was also adjusted to a simpler two-way split:

- 75% train
- 25% test

Compared to the last implementation, this approach eliminates the need for a separate validation set since **cross-validation is used for model evaluation** and Bagging provides built-in Out-of-Bag (OOB) error estimation for validation purposes.

*9.3 Base model: Decision Tree*

Before applying Bagging, a decision tree was evaluated as the base model.

Decision trees are employed to make predictions through learning a set of binary decision rules, effectively splitting the data based on the values of each feature. The algorithm creates a tree structure where each node represents a decision rule, and leaf nodes contain the final predictions. In a regression use case, they help reduce variance within splits.

Although trees can handle non-linear relationships quite well, they tend to overfit the train data, with small changes in features leading to a dramatic change in the tree's structure. This instability, in turn, make decision trees great candidates for ensemble methods like Bagging, which combines multiple trees to lower variance, while still capturing complex patterns.

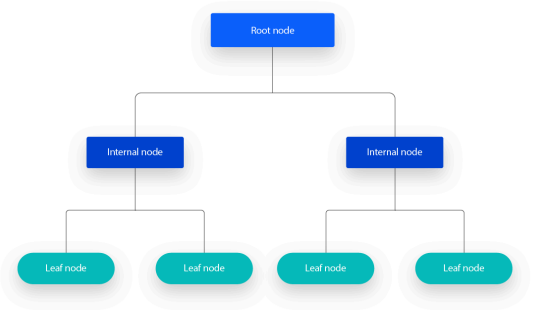


**Figure 8:** Decision tree structure (IBM, n.d.)

**Cross-validation** is a technique that divided data into multiple folds. According to Brownlee (2023), k-fold cross-validation involves splitting the dataset into k subsets and iteratively training on k−1 folds while validating on the remaining fold. Cross-validation provides more reliable performance estimates than single train-test splits because it tests the model on multiple different data subsets, giving a better overall picture of how well the model performs.

**7-Fold cross-validation** was used on the training set to obtain a robust estimate of baseline performance, with the average MSE as the comparison metric. The hyperparameters were:

- max_depth=10
- min_samples_leaf=4

Since there are no epochs in decision trees, these hyperparameters provide help in regularizing the model, stopping the tree from growing infinitely and without a clear direction.

This tree will serve as a direct comparison point to measure the improvement achieved by the Bagging model, with cross-validation providing a more reliable baseline estimate than a simple train-test evaluation.

*9.4 Bagging model implementation*

The Bagging model was implemented using the BaggingRegressor class from scikit-learn, which automates the bootstrap aggregation process. Configuration parameters were selected as follows:

- n_estimators=50
- Base estimator: Each tree uses the same hyperparameters (max_depth=10, min_samples_leaf=4), ensuring a fair comparison with the baseline tree
- Bootstrap sampling: Each tree is trained on a different bootstrap sample (random sampling with replacement) from the training set, containing around ~63% unique samples

Bagging trains 50 decision trees independently on different bootstrap samples. This randomness helps the ensemble learn varied patterns from the data, improving robustness.

For regression, Bagging averages the predictions from all trees. This simple aggregation reduces variance, as individual tree errors offset each other.

*9.5 Analysis on the number of estimators*

To assess how ensemble size affects performance, the Bagging regressor was trained with increasing numbers of estimators: [5, 10, 15, …, 200]. The error

curve reveals that most performance gains occur early: the test MSE drops sharply up to around 30 estimators, then flattens out. After 50 estimators, additional trees bring minimal improvement.

This suggests that 50 estimators offer an optimal trade-off between accuracy and computational cost. The test MSE remains consistently close to the training MSE across all values, confirming Bagging's stability and low risk of overfitting—even with larger ensembles.

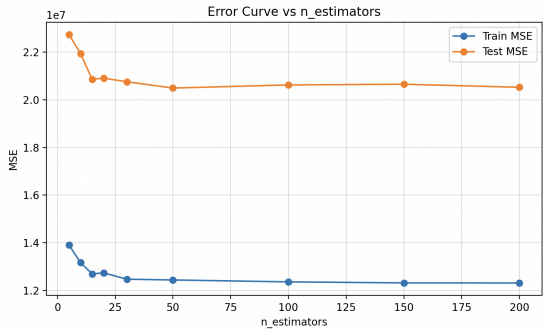


**Figure 9:** Error curve against estimator count

*9.6 Analysis on feature importance*

The feature importance analysis in Bagging offers a more reliable measure of variable significance by averaging importance scores across all 50 trees. This aggregation reduces the high variance seen in individual trees, where small data changes can skew rankings. Each tree estimates importance based on how much each feature reduces mean squared error during splits.

In this case, smoking status was the most important variable, followed by age and BMI, which makes sense given how these factors affect health costs. These results match what we already observed in the linear model, but with more consistency.

This analysis also helps to simplify the model. Features that consistently show low importance could be removed, whereas essential ones could be improved through transformation.

## 10. FRAMEWORK MODEL RESULTS

The results achieved demonstrate that Bagging significantly outperforms the base decision tree. The baseline decision tree achieved a cross-validation MSE of 26,277,684, with an $R^2$ of **0.803**, while the Bagging ensemble with 50 estimators reduced the test MSE to 20,491,365, this represents a 22% improvement in prediction accuracy. The level of error reduction achieved stems from the effectiveness of Bootstrap Aggregation in reducing variance, while maintaining predictive capabilities.

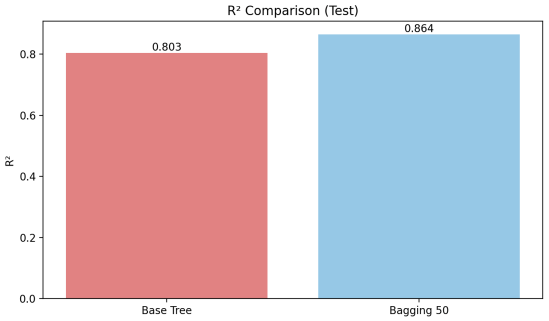


**Figure 10**: $R^2$ Comparison (Base Tree vs Bagging 50)

The Bagging model achieved the following $R^2$:

- Train $R^2$: 0.9143
- Test $R^2$: 0.8642

```
Baseline Bagging (50 estimators):
Train -> MSE: 12440512.79 | RMSE: 3527.11 | MAE: 1925.85 | R²: 0.9143
Test -> MSE: 20491365.25 | RMSE: 4526.74 | MAE: 2506.31 | R²: 0.8642
```

**Figure 11**: Model performance metrics for training and test datasets in a Bagging model

In practice, this means that the Bagging model can explain approximately 86% of the variance in medical insurance charges. The training performance, of 91% explicable variance, shows a gap of just 0.0501 in the $R^2$ value, indicating that the ensemble successfully avoids severe overfitting despite its complexity, showing it **being well-adjusted**, proving a **low grade of variance**, without signs of underfitting or severe overfitting.

The scatter plot comparing predicted vs. actual charges (top-left of Figure X) shows that the Bagging model closely follows the ideal prediction line, especially for intermediate values. This supports the strong $R^2$ and RMSE metrics, and confirms the model's ability to generalize well on unseen data.

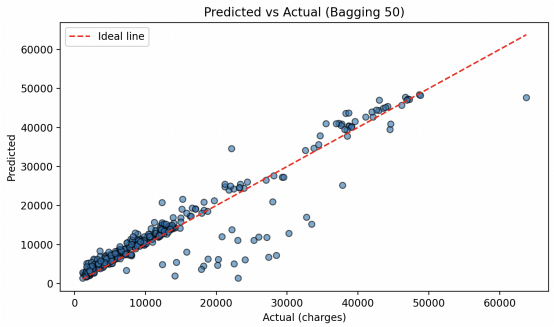


**Figure 12**: Predicted vs. actual medical charges

The feature importance scores calculated across the Bagging ensemble reveal a dominance of smoking status in predicting medical costs. The variables smoker_no and smoker_yes together account for over 66% of the total importance, confirming that smoking behavior is by far the strongest predictor in the model.

Other notable features include BMI (18.6%) and age (12.8%). In contrast, categorical variables like region and sex, which were one-hot encoded, contribute minimally to the model's decisions—each representing well under 1% of total importance. This shows that lifestyle-related characteristics are the most important determinant of insurance costs, even though demographic and geographic factors may also be important.

```
Top feature importances:
smoker_no          0.396439
smoker_yes         0.267391
bmi                0.185941
age                0.128291
children           0.011496
region_northeast   0.003412
region_northwest   0.001633
sex_male           0.001473
region_southeast   0.001417
sex_female         0.001336
region_southwest   0.001171
```

**Figure 13**: Importance per feature

## 11. CORRECTIONS

During an intermediate code revision, the teacher pointed out that there was no need to do categorical encoding for the **bmi** column, which was previously converted into one of six classes, ranging from underweight to class 3 obesity, in response, this change was dropped in favor of keeping the numerical bmi column. In addition, columns such as **smoker, region** and **sex**, which were previously also

categorically encoded, were instead one-hot encoded.

Automatic hyperparameter establishment in the Bagging model was dropped in favor of a manual adjustment.

## 12. CONCLUSION

This paper presents a comprehensive comparison between a manual gradient descent linear regression and an ensemble-based Bagging method for predicting medical costs, uncovering significant differences in their predictive capabilities and bias-variance characteristics.

The results prove a significant performance gap between the two approaches. The manual gradient descent linear regression achieved a test $R^2$ of 0.7534, while the Bagging ensemble achieved a test $R^2$ of 0.8642, representing a **14.7%** improvement in explanatory power over the previous model. This translates to significantly better predictive accuracy, with the ensemble method explaining 86% of the variance in medical charges compared to the linear model's 75%. The following table compares results from both models:

| **Dataset** | **Model** | **MSE** | **$R^2$** |
|---|---|---|---|
| **Train** | GD | 31431930.6605 | 0.7717 |
| | Bagging | 12440512.7900 | 0.9143 |
| **Validation** | GD | 18978095.5365 | 0.8147 |
| **Test** | GD | 3010210 | 0.7534 |
| | | 8.8732 | |
| | Bagging | 20491365.2500 | 0.8642 |

Linear regression exhibited low variance with consistent performance across datasets, however, it suffered from medium bias due to its inability to capture non-linear relationships. The Bagging ensemble addresses these limitations by reducing both bias and variance simultaneously, capturing complex patterns while ensuring reasonable generalization (training-test $R^2$ gap of 0.0501).

Both prediction attempts consistently identified smoking-related indicators as the top cost drivers, however, Bagging provided a more robust validation thanks to the use of cross validation and OOB estimation, while the validation set of the gradient descent displayed contamination issues with its abnormally high $R^2$ of 0.8147.

The Bagging ensemble method proves to be superior thanks to its ability to handle complex, non-linear relationships. The 14.7% improvement over the manual model has direct implications for insurance pricing in real life, making the model with machine learning the preferred choice for complex regression problems.

## 13. REFERENCES


a. Tukey, J. W. (1962). *The future of data analysis. The Annals of Mathematical Statistics*, 33(1), 1–67.

b. Darden, M. E. (2022). *Smoking, selection, and medical care expenditures*. National Bureau of Economic Research. https://www.nber.org/system/files/working_papers/w29885/w29885.pdf
c. Breiman, L. (2004). *Out-of-bag estimates*. University of California, Berkeley. https://www.stat.berkeley.edu/~breiman/OOBestimation.pdf
d. Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81-106.
e. IBM. (n.d.). *What is a decision tree?* https://www.ibm.com/think/topics/decision-trees
f. Brownlee, J. (2023, October 4). *A gentle introduction to k-fold cross-validation*. Machine Learning Mastery. https://www.machinelearningmastery.com/k-fold-cross-validation/
g. Lyashenko, V., & Jha, A. (2025, April 25). *Cross-validation in machine learning: How to do it right.* Neptune AI. https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right
h. GeeksforGeeks. (2025, August 2). *Regularization in machine learning.* https://www.geeksforgeeks.org/machine-learning/regularization-in-machine-learning/