A Study of Gradient Boosting

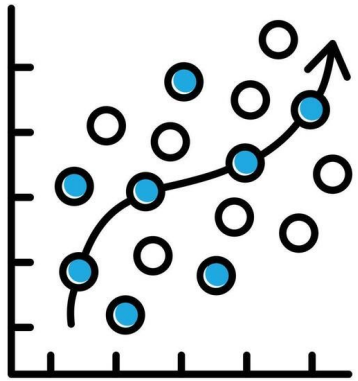
Minh Ngo and Ian DeLano

University of Florida

STA 4241 Final Project

Why Predictive Modeling?

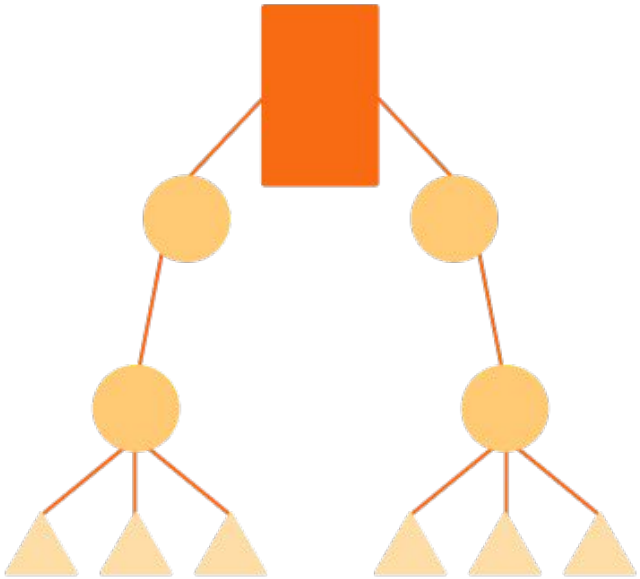
- Predictive modeling helps with data-driven decisions in business, finance, and more.
- Real-world data is messy & noisy- models need to balance accuracy, flexibility, and interpretability
- Goal: Create models that can generalize well on unseen data
- Many predictive models available:
  - Linear models, SVMs, k-NN, Random Forests, etc…
  - But today, we'll focus on Decision Trees — and how to boost them.



Why Decision Trees?

- Very common and interpretable ML model.
- Can handle classification and regression tasks.
- Simple and easy to use.
- But… comes with limitations:
  - Prone to overfitting.
  - Weak as a stand alone model, high variance.

This is where **Gradient Boosting** comes in.

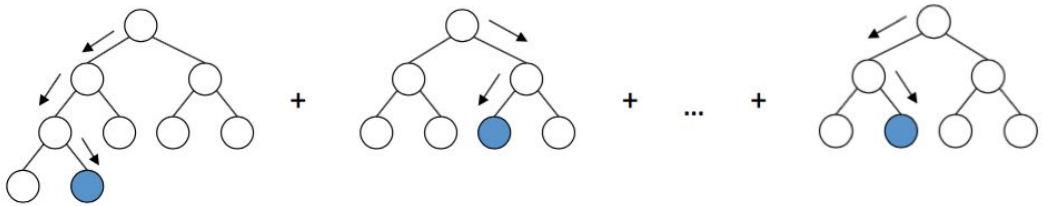


Why Gradient Boosting?

- Decision trees underperform on complex datasets.
- Gradient Boosting improves on decision trees:
  - Build an series of weak learners (trees).
  - Each tree learns from the errors of the previous one.
  - Iterative training enables learning from mistakes.

★ Advantages

- ➢ Better accuracy
- ➢ More generalization
- ➢ Captures nonlinearity and feature interactions



Project Overview

- Goal: Understand gradient boosting and how improves on decision trees.
- Dataset: Wine Quality Dataset (Kaggle)
  - Features: pH, alcohol, acidity, sulfate, etc.
  - Target: Quality score (0-10)
- Modeling Tasks:
  - Regression on continuous quality score
  - Classification to predict discrete quality class
- Compare Models' Performance:
  - Gradient Boosting (XGBoost) vs Single Decision Tree

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	fixed acidi	volatile ac	citric acid	residual s	chlorides	free sulfur	total sulfu	density	pH	sulphates	alcohol	quality	Id
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	0
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5	1
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5	2
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6	3
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	4
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5	5
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5	6
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7	7

The Additive Model

- Framework
  - Ensemble method
    - Sequentially combining weak learns for powerful model
    - Each model corrects errors of previous one
  - Minimizes a loss function
- Contrasts models like Random forest that is parallel in nature
- Additive Model Formula:
  - Creates new model at iteration m
  - Adds new weak learner from previous iteration

$$F_m(x) = F_{m-1}(x) + h_m(x)$$

Weak Learner

- At each iteration of the model, we train a new weak learner $h_m(x)$.
- This is done by training $h_m(x)$ to fit the “residuals” of the loss function:

$$h_m(x) = \arg\min_h \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + h(x_i)\right)$$

- The chosen loss function ***L*** is used to guide each learner.

Loss Functions

- Loss functions are used as a guide
- For regression problems, we often use squared error:

$$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

- For classification, log loss is commonly used:

$$L(y_i, \hat{p}_i) = -\log \hat{p}_{i,y_i}$$

- At each step, the new model is trained to reduce this loss — by focusing on the parts the current model is getting wrong.
- In practice, this means the new model is trained on the negative gradient.

Final Formula

- Caution must be brought to fitting new weak learners
  - Adding new trees too aggressively can cause overfitting.
- A learning rate(η) is employed to adjust with smaller and safer steps.
- Additive formula becomes:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$$

- Encourages steadier improvements:
  - Smaller η Requires more trees and longer computation time.
  - Typically leads to stronger final performance.

Gradient Boosting Algorithm

- Putting it all together you have the Gradient Boosting Algorithm

1. **Initialize:** $F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$
2. **For** $m = 1$ to M:
   (a) Compute pseudo-residuals: $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$
   (b) Fit weak learner $h_m(x)$ to $\{x_i, r_{im}\}$.
   (c) Update model: $F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$
3. **Return:** Final model $F_M(x)$

➔ Pseudo-residuals are a computation of a negative gradient.

➔ Weak learner is trained on pseudo-residuals to approximate the negative gradient.

➔ Model is updated by adding the weak learner using a fixed learning rate eta.

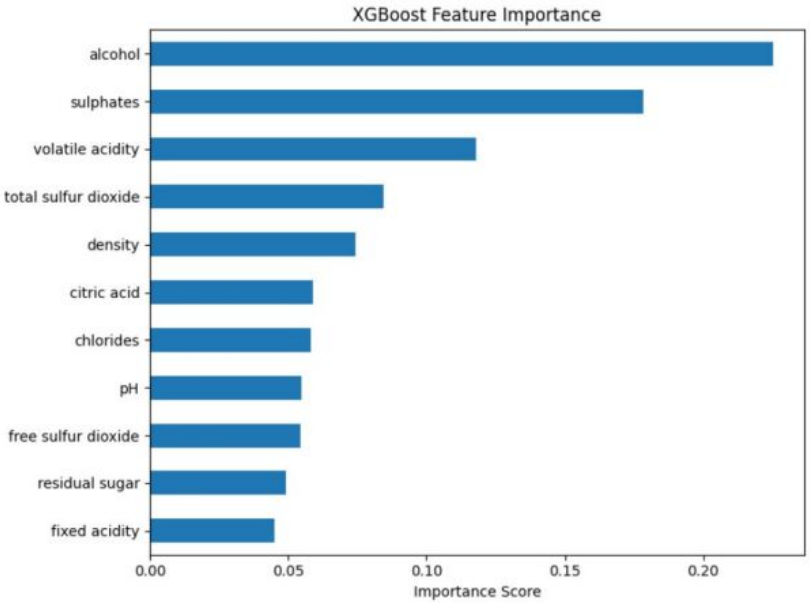
Implementation and Data Collection

- Red Wine Quality Dataset Used
  - Contains roughly 1600 samples, 11 features
  - Wine quality score as target
- Preprocessing
  - Removed non-predictive columns such as ID.
  - Split into feature matrix X (input) and label vector Y (target quality score)
  - Applied standard scaling to features for consistency.
- Modeling Tasks
  - Regression: Predict continuous quality score.
  - Classification: Predict categorical quality class (3-8)
- Algorithms Used
  - Decision Tree as a baseline.
  - XGBoost for Gradient Boosting
    - Tuned with GridSearchCV, 3-fold cross-validation

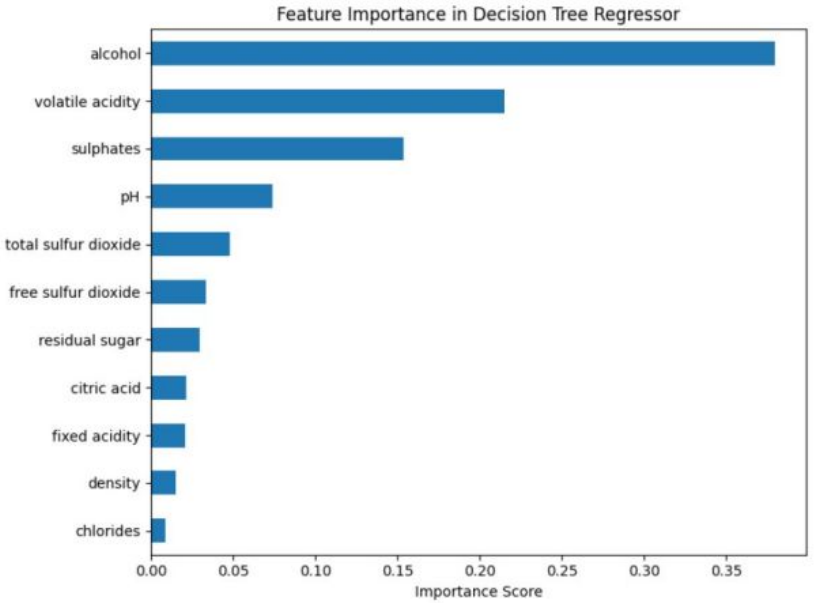
Regression Results: XGBoost vs Decision Tree

- Metrics used to evaluate performance:
  - Root Mean Squared Error (RMSE)
  - R-Squared (R^2)
- Decision Tree Regressor (depth = 5)
  - RMSE = 0.7103, R^2 = 0.1324
- XGBoost Regressor
  - Untuned: RMSE = 0.5848, R^2 = 0.4013
  - Tuned: RMSE = 0.5782, R^2 = 0.4251
- Summary:
  - XGBoost outperforms decision trees on both error and fit.
  - Improved generalization and error minimization
  - Tuning offers improved performance, although marginal.

Results - Feature Importance



(a) XGBoost Regressor



(b) Decision Tree Regressor

Figure 1: Feature Importance Comparison Between Models

Classification Results: Decision Tree vs XGBoost

- Setup
  - Converted continuous wine scores into discrete classes (3-8)
    - Extrema had lack of available data
  - Performanced measured by accuracy.
- Decision Tree Classifier
  - 59.38% accuracy
- XGBoost Classifier
  - Untuned
    - 68.12% accuracy
  - Tuned (via GridSearchCV cross-validation)
    - 69.43% accuracy
- Summary:
  - XGBoost outperforms base model, tuning shows marginal improvement.

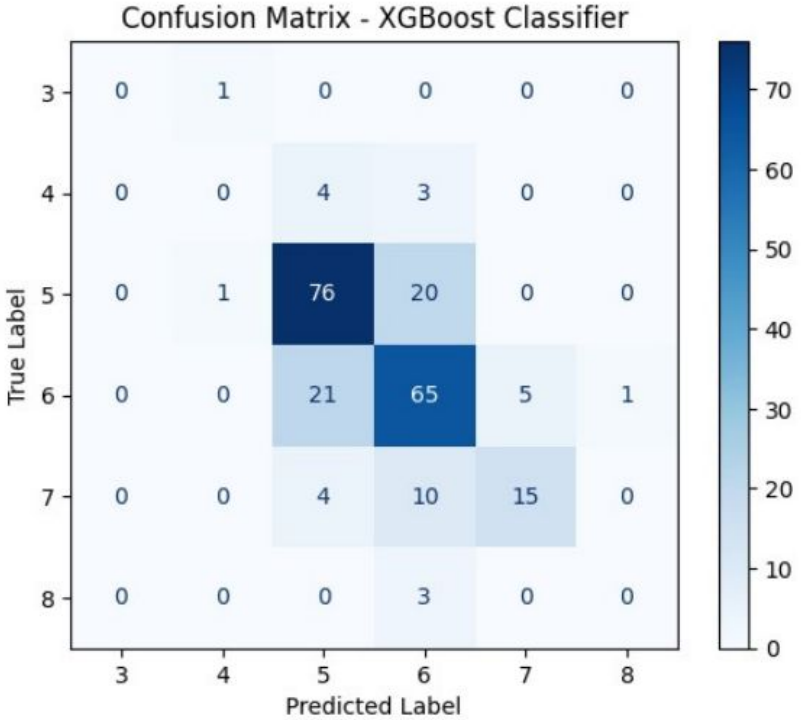
Results - Classification Limitation

- Further Inspection of Data:
  - Found majority of scores clustered between 5-7
  - Sparsity in extreme classes
- Found model overfitting to dominant classes
- As shown in Histogram:
  - The imbalanced labels skewed model performance
  - Thus limiting generalization

(a) Distribution of Wine Quality Classes

Results - Confusion Matrix

- Imbalance further shown with a confusion matrix
  - Caused over-prediction of common labels
  - High misclassification rate for underrepresented classes
- This highlights that classifier can be weak in cases of class imbalances
- Limitations of model to not predict entire range of classes



(b) Confusion Matrix for XGBoost Classifier

Conclusion

- We found that Gradient Boosting via XGBoost, consistently outperformed decision trees in both regression and classification.
- Our study demonstrated significant improvements in generalization and predictive power, even with minimal tuning.
- Gradient Boostings' sequential approach, proved to be a flexible and powerful modeling strategy compared to single decision trees.
- There were limitations in regards to class imbalances where minority classes were often misclassified.
- Gradient Boosting stands to be a reliable, scalable, and extremely effective technique for predictive modeling tasks with the input of structured data.

References

[1] Analytics Vidhya. (2021). Gradient Boosting Algorithm: A Complete Guide for Beginners. Retrieved from https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-completeguide-for-beginners/h-what-is-a-gradient-boosting-algorithm

[2] Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). The Elements of Statistical Learning (2nd ed.). New

York: Springer. Chapter 10: Boosting and Additive Trees, pp. 337–384. ISBN: 978-0-387-84857-0.

[3] IBM. (n.d.). What is Gradient Descent? IBM Think. Retrieved from

https://www.ibm.com/think/topics/gradient-descent

[4] GeeksforGeeks. ML — Gradient Boosting. Retrieved from https://www.geeksforgeeks.org/ml-gradientboosting/

[5] Friedman, J. H. (February 1999). Evaluation of Predictive Models in Imbalanced Classification Tasks.

Retrieved from https://web.archive.org/web/20191101082737/http://statweb.stanford.edu/ jhf/ftp/trebst.pdf

[6] Yasser H. (2018). Wine Quality Dataset. Retrieved from https://www.kaggle.com/datasets/yasserh/winequality-dataset

Questions?