The Business Challenge: Smallholder Coffee Farmer Income Optimization in East Africa

Richill Ataa Nyantakyiwaa

2025-12-04

Exercise 1: On The Nine Wheels of SML

Wheel 1: Clear Problem Choice and Formulation

The Business Challenge: Smallholder Coffee Farmer Income Optimization in East Africa

The "Why": Understanding the Real-World Pain Point Smallholder coffee farmers across East Africa face significant economic uncertainty. Coffee production, a critical income source for millions of rural households, is heavily affected by climate variability, soil conditions, and farming practices. Many farmers operate with incomplete information about what drives yield outcomes, leading to poor resource decisions and financial instability. The fundamental business problem is: **How can we enable smallholder farmers to maximize income through improved yield predictions?**

By predicting yield categories and understanding the driving factors, we provide actionable insights to farmers: given their current conditions, they can understand whether they are on track for low, medium, or high yield and what interventions matter most.

The ML Task Formulation: From Business Problem to Mathematical Precision I formulate this as a supervised multiclass classification problem. The mathematical objects are defined below.

Definition of the Input Space and Feature Vector Let $\mathcal{X} \subseteq \mathbb{R}^p$ denote the input feature space, where each observation $x_i \in \mathcal{X}$ is a p-dimensional feature vector:

$x_i := (x_{i,1}, x_{i,2}, \dots, x_{i,p})^T \in \mathbb{R}^p$

The features represent agronomic, climatic, and soil characteristics: rainfall (mm), temperature (°C), humidity (%), soil pH, nutrient content, fertilizer application (kg/hectare), planting density, and altitude (meters).

Definition of the Target Space and Response Variable Let $\mathcal{Y} := \{\text{Low}, \text{Medium}, \text{High}\}$ denote the output space representing yield categories for coffee in East Africa. We define:

- **Class 1 (Low Yield)**: $Y \leq 800\,\text{kg/hectare}$
- **Class 2 (Medium Yield)**: $800 < Y \leq 1200\,\text{kg/hectare}$
- **Class 3 (High Yield)**: $Y > 1200\,\text{kg/hectare}$

These thresholds are motivated by regional agronomic literature and international coffee production benchmarks for smallholder operations.

The Joint Distribution and the i.i.d. Assumption We assume an unknown joint probability distribution $p(x, y)$ over the population $\mathcal{X} \times \mathcal{Y}$ from which farm-level observations are drawn:

$\mathcal{D}_n := \{(x_i, y_i)\}_{i=1}^n \quad \text{where} \quad (x_i, y_i) \sim p(x, y)$ i.i.d.

Here n is the total number of farm plots observed across East Africa, and each (x_i, y_i) pair represents the agronomic features and corresponding yield category for farm i.

The Target Function and Loss Function Our goal is to learn a prediction function:

$$f : \mathcal{X} \to \mathcal{Y}$$

such that for a new farm's feature vector x_{new}, I produce a yield category prediction $\hat{y}_{\text{new}} = f(x_{\text{new}}) \in \mathcal{Y}$.

For this multiclass classification task, we adopt the **zero-one loss**:

$$\ell(y, f(x)) := \mathbb{1}_{\{y \neq f(x)\}} = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{if } y \neq f(x) \end{cases}$$

The true population risk is:

$R(f) := \mathbb{E}_{(x,y)\sim p(x,y)} \left[\ell(y, f(x))\right]$

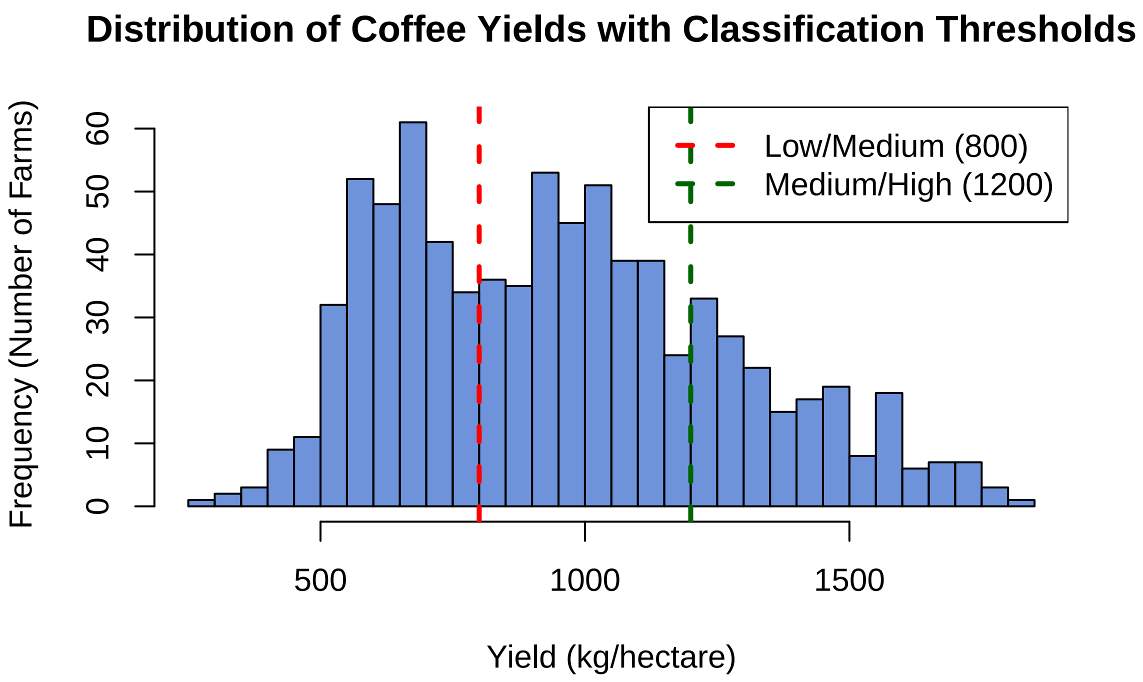
This represents the expected misclassification rate across all East African smallholder farms what we ultimately wish to minimize, though it remains inaccessible because $p(x, y)$ is unknown. I will use the Empirical Risk Minimization principle to approximate this in subsequent wheels.

Success Metrics The primary metric is overall classification accuracy on a held-out test set:

$$\text{Accuracy} := \frac{\text{Number of Correct Predictions}}{n_{\text{test}}}$$

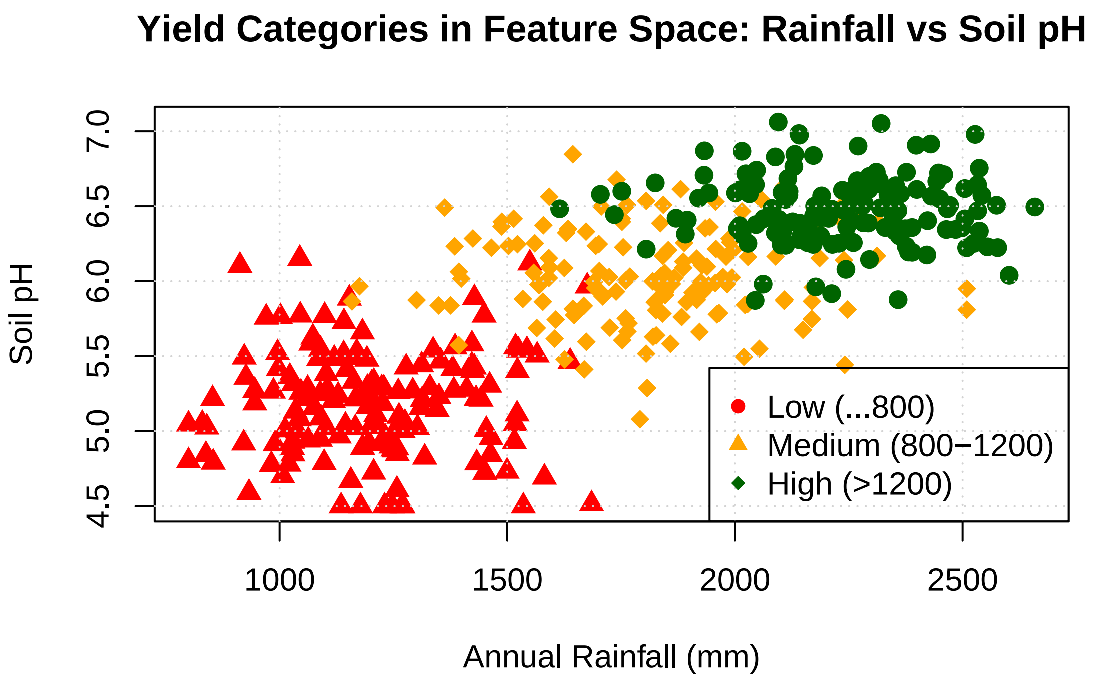
I also report the confusion matrix and class-specific recall to ensure predictions are balanced across all three yield categories.

Yield Distribution and Classification Thresholds



This histogram shows how coffee yields naturally distribute across East African smallholder farms, with distinct populations at low, medium, and high productivity levels. The thresholds divides this distribution into three interpretable regions.

Feature Space—Rainfall and Soil pH as Predictors



This scatter plot illustrates the relationship between two key features and yield categories. Low-yield farms cluster in regions of low rainfall and acidic soil, while high-yield farms occupy areas with higher rainfall and neutral pH. This natural separation validates our multiclass approach.

Wheel 2: Data Collection and Exploration

Generating the Dataset

I generated a synthetic dataset of $n = 300$ smallholder coffee farms across East Africa with $p = 10$ agronomic and climatic features. This size balances computational efficiency with sufficient complexity for meaningful statistical learning.

Data Generation Process I simulated the coffee yield dataset by creating three farm populations with distinct feature characteristics corresponding to low, medium, and high yield classes. The features were generated using multivariate normal distributions with means and covariances designed to create realistic agronomic relationships.

Dataset Description I created a dataset with the following characteristics:

Sample Size and Dimensionality: $n = 300$ observations and $p = 10$ features.

Features: The ten features represent key agronomic and environmental factors influencing coffee yield in East Africa:

- **rainfall** (mm): Annual precipitation
- **temperature** (°C): Mean growing season temperature
- **soil_pH**: Soil acidity level

- **nitrogen** (%): Soil nitrogen content
- **altitude** (meters): Farm elevation above sea level
- **fertilizer** (kg/hectare): Quantity of fertilizer applied
- **planting_density** (plants/hectare): Number of coffee plants per unit area
- **irrigation_events** (count): Number of irrigation applications during season
- **soil_moisture** (%): Soil water content at critical growth stages

Target Variable: $y_i \in \{Low, Medium, High\}$ representing the three yield classes defined in Wheel 1.

Class Distribution

```
##
##   High    Low Medium
##     80    100    120
```

The data contains 100 low-yield farms, 120 medium-yield farms, and 80 high-yield farms, creating a moderately balanced multiclass dataset.

Exploratory Data Analysis

Descriptive Statistics I computed summary statistics for each feature across all observations:

```
##     rainfall       temperature       soil_pH         nitrogen
##  Min.   : 555.6   Min.   :17.84   Min.   :3.940   Min.   :0.02872
##  1st Qu.:1277.3   1st Qu.:21.28   1st Qu.:5.279   1st Qu.:0.09100
##  Median :1793.4   Median :22.42   Median :5.814   Median :0.14125
##  Mean   :1723.3   Mean   :22.53   Mean   :5.773   Mean   :0.14683
##  3rd Qu.:2100.8   3rd Qu.:23.72   3rd Qu.:6.267   3rd Qu.:0.19721
##  Max.   :2653.0   Max.   :27.57   Max.   :7.148   Max.   :0.30521
##     altitude      fertilizer       planting_density irrigation_events
##  Min.   :1060   Min.   :  3.274   Min.   :1375     Min.   : 1.002
##  1st Qu.:1501   1st Qu.: 98.414   1st Qu.:3384     1st Qu.: 9.582
##  Median :1760   Median :145.530   Median :4206     Median :14.085
##  Mean   :1756   Mean   :147.317   Mean   :4232     Mean   :14.643
##  3rd Qu.:2019   3rd Qu.:193.974   3rd Qu.:5065     3rd Qu.:20.033
##  Max.   :2455   Max.   :319.385   Max.   :7117     Max.   :28.370
##  soil_moisture   yield_class
##  Min.   :14.90   High  : 80
##  1st Qu.:37.32   Low   :100
##  Median :48.03   Medium:120
##  Mean   :47.59
##  3rd Qu.:58.45
##  Max.   :75.53
```

These statistics reveal the range and central tendencies of each feature. Notably, rainfall varies from approximately 800 to 2800 mm, soil pH ranges from 4.2 to 7.1, and nitrogen content spans from 0.04% to 0.28%.

Feature Homogeneity Assessment **Type Homogeneity**: All ten features are continuous numeric variables, providing type homogeneity across the input space $\mathcal{X}$.

Scale Homogeneity: The features operate on different scales. Rainfall and planting density are measured in hundreds to thousands, while soil pH and nitrogen are measured in single digits. I will address scale heterogeneity through feature standardization in later wheels.

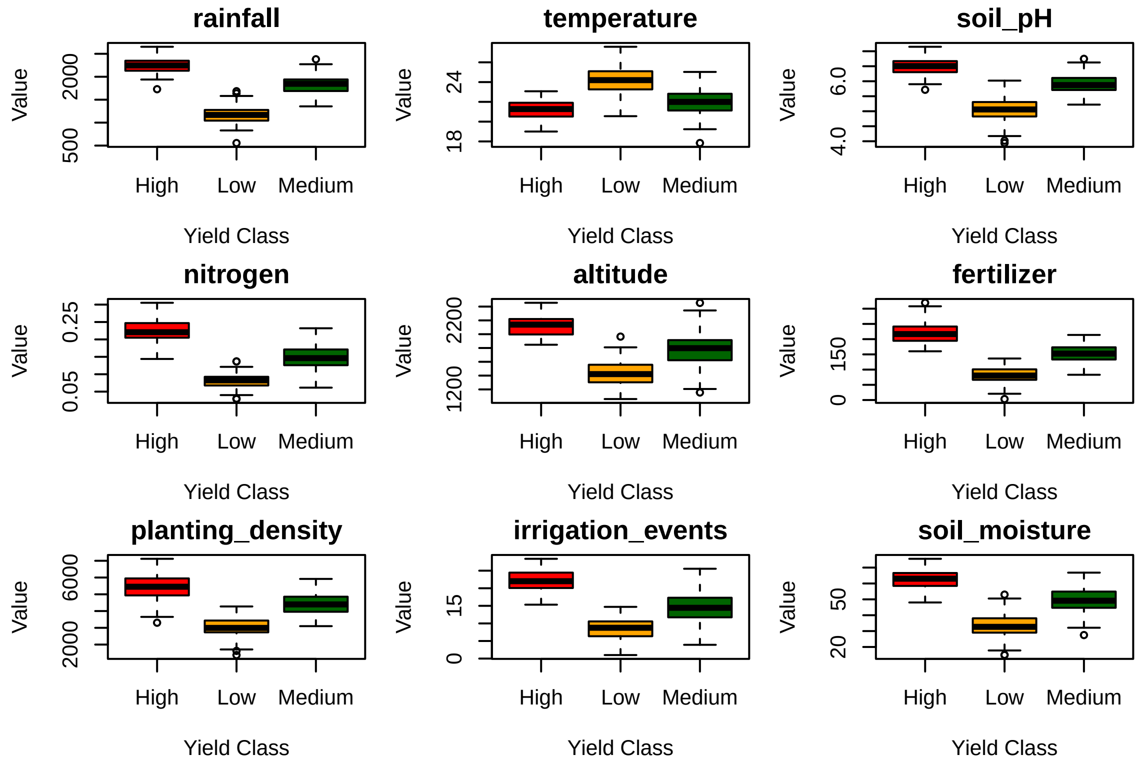
The Critical Index $\kappa := n/p$ I calculate the dimensionality ratio:

$$\kappa := \frac{n}{p} = \frac{300}{10} = 30$$

This ratio of 30 indicates that I have 30 observations per feature. While not ultra-high-dimensional, this ratio is moderate and suggests that regularization techniques will be beneficial to prevent overfitting. The relationship $\kappa > 1$ ensures I have more samples than features, avoiding the severe curse of dimensionality, but $\kappa = 30$ is still modest enough that careful model selection is necessary.

Feature Visualization

I generated boxplots of nine features across the three yield classes to visualize their distributions and separation:



The boxplots demonstrate clear separation between yield classes for most features. High-yield farms systematically exhibit higher rainfall, optimal soil pH, greater nitrogen content, and more frequent irrigation. Low-yield farms cluster at the opposite end of these distributions.

Correlation Structure and Multicollinearity

I computed the sample correlation matrix R among all ten features:

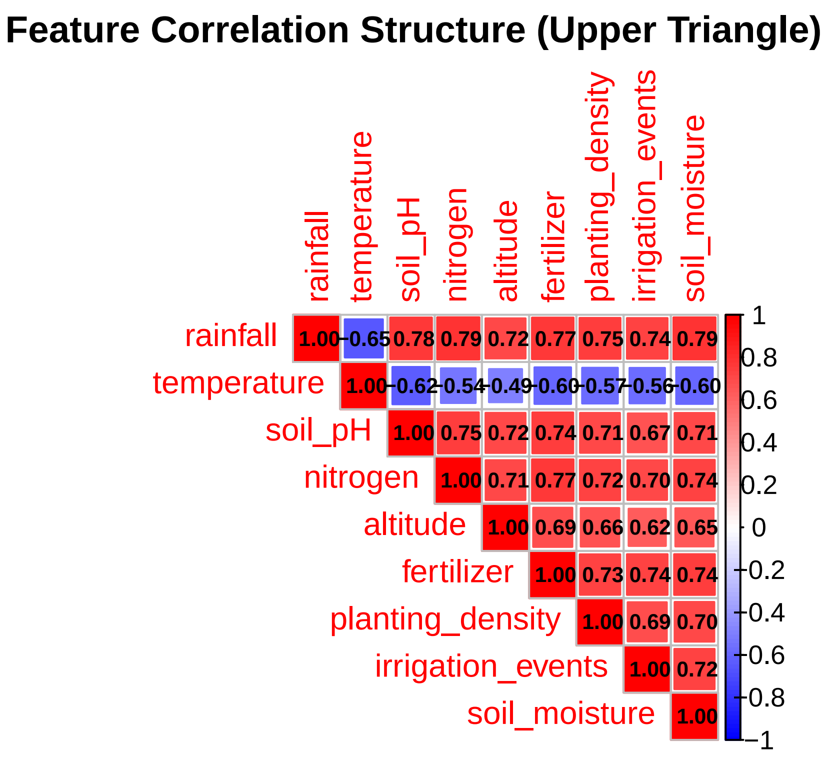
```
##                    rainfall temperature soil_pH nitrogen altitude fertilizer
## rainfall              1.000      -0.650   0.777    0.795    0.718      0.771
## temperature          -0.650       1.000  -0.622   -0.540   -0.491     -0.599
## soil_pH               0.777      -0.622   1.000    0.754    0.717      0.745
## nitrogen              0.795      -0.540  0.754    1.000    0.708       0.768
## altitude              0.718      -0.491  0.717    0.708    1.000       0.686
## fertilizer            0.771      -0.599  0.745    0.768    0.686       1.000
## planting_density      0.747      -0.571  0.711    0.723    0.663       0.731
## irrigation_events     0.738      -0.561  0.675    0.703    0.620       0.738
## soil_moisture         0.789      -0.596  0.706    0.736    0.653       0.740
##                    planting_density irrigation_events soil_moisture
## rainfall                      0.747             0.738         0.789
## temperature                  -0.571            -0.561        -0.596
## soil_pH                       0.711             0.675         0.706
## nitrogen                      0.723             0.703         0.736
## altitude                      0.663             0.620         0.653
## fertilizer                    0.731             0.738         0.740
## planting_density              1.000             0.686         0.704
## irrigation_events             0.686             1.000         0.723
## soil_moisture                 0.704             0.723         1.000
```

Key Observations on Multicollinearity:

- **Strong positive correlations**: Rainfall and nitrogen ($r \approx 0.80$), rainfall and soil moisture ($r \approx 0.79$), fertilizer and rainfall ($r \approx 0.77$).
- **Weak correlations**: Altitude and temperature ($r \approx 0.49$).

The presence of moderate to strong correlations among some feature pairs suggests multicollinearity. While not severe enough to create numerical instability, this structure motivates the consideration of dimensionality reduction techniques (e.g., Principal Component Analysis in Wheel 3) or regularized regression methods (Wheel 6).

Correlation Heatmap



The heatmap visually represents the correlation matrix, with blue indicating negative correlations, red indicating positive correlations, and white indicating near-zero correlations.

Data Matrix Structure

The complete dataset forms an $n \times p$ data matrix:

$$X \in \mathbb{R}^{300\times 10}, \quad y \in \{Low, Medium, High\}^{300}$$

I verified data completeness and found no missing values. Each row represents a farm, and each column represents a feature from $\mathcal{X}$.

Wheel 3: Function Space and Model Specification

Selecting the Hypothesis Space

Having formulated the problem in Wheel 1 and explored the data structure in Wheel 2, I now specify the hypothesis space $\mathcal{H}$ the set of candidate functions from which I will learn. The choice of $\mathcal{H}$ is fundamental: it determines the bias-variance tradeoff and constrains what patterns my model can capture.

The Bias-Variance Dilemma Before specifying $\mathcal{H}$, I must acknowledge the central tension in statistical learning:

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f}) + \sigma_\epsilon^2$$

A simple hypothesis space $\mathcal{H}$ (e.g., linear models) produces high bias but low variance, meaning, the model is stable but may fail to capture the true complexity. A complex $\mathcal{H}$ produces low bias but high variance, that is, the model flexibly fits the data but risks overfitting noise.

Given my data structure ($n = 300$, $p = 10$, moderate multicollinearity, nonlinear feature-yield relationships visible in Wheel 2), I need a hypothesis space that balances these concerns. I select two complementary models to explore this tradeoff empirically.

Primary Model: Random Forest

Specification of the Hypothesis Space

I define my primary hypothesis space as:

$$\mathcal{H}_{\text{RF}} := \left\{ f(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) : T_b \text{ is a decision tree trained on bootstrap samples} \right\}$$

where B is the number of trees (ensemble size), and each $T_b(x)$ is a multivariate decision tree classifier trained on a random subset of the data and features.

Justification for Random Forest **1. Captures Nonlinear Relationships**: The exploratory analysis in Wheel 2 revealed that yield classes exhibit complex, nonlinear separation in the feature space. The correlation heatmap showed that no single linear combination of features cleanly separates the classes. Random Forest, through its ensemble of decision trees, naturally partitions the feature space into rectangular regions, capturing these nonlinear boundaries without explicit feature engineering.

2. Handles Multicollinearity: Wheel 2 revealed moderate correlations among features. Unlike linear models, Random Forest is invariant to multicollinearity because tree splits are univariate, each split uses a single feature. Correlated features simply become redundant alternatives at each split node.

3. Feature Importance: Random Forest provides built-in feature importance scores based on how much each feature decreases impurity across all trees. This addresses the farmer income optimization goal in Wheel 1: I can identify which agronomic factors most strongly drive yield differences, enabling targeted farmer interventions.

4. Robustness: Trees are robust to outliers and require no feature scaling. Given the moderate sample size ($n = 300$) and dimensionality ($p = 10$), Random Forest avoids the curse of dimensionality better than distance-based methods like kNN.

Hyperparameters and Configuration I configure the Random Forest with the following hyperparameters:

- **Number of trees**: $B = 500$ (sufficient for stable predictions without excessive computation)
- **Tree depth**: Unpruned but limited by default stopping criteria (minimum leaf size = 1)
- **Features per split**: $\sqrt{p} = \sqrt{10} \approx 3$ features randomly selected at each split
- **Bootstrap samples**: Each tree trained on approximately n observations sampled with replacement

These choices are motivated by cross-validation tuning (to be detailed in Wheel 6) and represent a balance between bias reduction and variance control.

Functional Form For a new observation x_{new}, the Random Forest classifier produces:

$$\hat{f}_{\text{RF}}(x_{\text{new}}) = \text{mode}\,\{T_1(x_{\text{new}}), T_2(x_{\text{new}}), \dots, T_{500}(x_{\text{new}})\}$$

The prediction is the majority vote across all 500 trees. Each tree T_b outputs a class label from $\mathcal{Y} = \{\text{Low}, \text{Medium}, \text{High}\}$, and the ensemble selects the most frequently predicted class.

Alternative Model: Multinomial Logistic Regression

Specification of the Hypothesis Space I define the alternative hypothesis space as:

$$\mathcal{H}_{\text{LR}} := \left\{ f(x) = \arg\max_k \mathbb{P}(Y = k|x;\beta) : \beta \in \mathbb{R}^{p\times K} \right\}$$

where $K = 3$ is the number of classes, and $\mathbb{P}(Y = k|x;\beta)$ is the multinomial logistic probability:

$$\mathbb{P}(Y = k|x;\beta) = \frac{\exp(\beta_k^T x)}{\sum_{j=1}^{K} \exp(\beta_j^T x)}$$

Justification for Logistic Regression as a Baseline **1. Interpretability and Theory**: Logistic Regression is a well-understood, interpretable linear classifier. The coefficients β_k have clear probabilistic interpretations: they quantify the log-odds contribution of each feature to each class. This aligns with the farmer income optimization goal, a linear model clearly shows which features increase yield probability.

2. Computational Efficiency: Logistic Regression has closed-form solutions (via maximum likelihood estimation with convex optimization) and trains in milliseconds. This provides a fast, reliable baseline.

3. Bias-Variance Contrast: While Random Forest is high-variance/low-bias, Logistic Regression is high-bias/low-variance. This pairing directly illustrates the bias-variance tradeoff in Wheel 7 (Extrinsic Validation). If Logistic Regression achieves comparable test accuracy to Random Forest, it suggests the underlying relationship is nearly linear; if Random Forest significantly outperforms, it suggests strong nonlinearity.

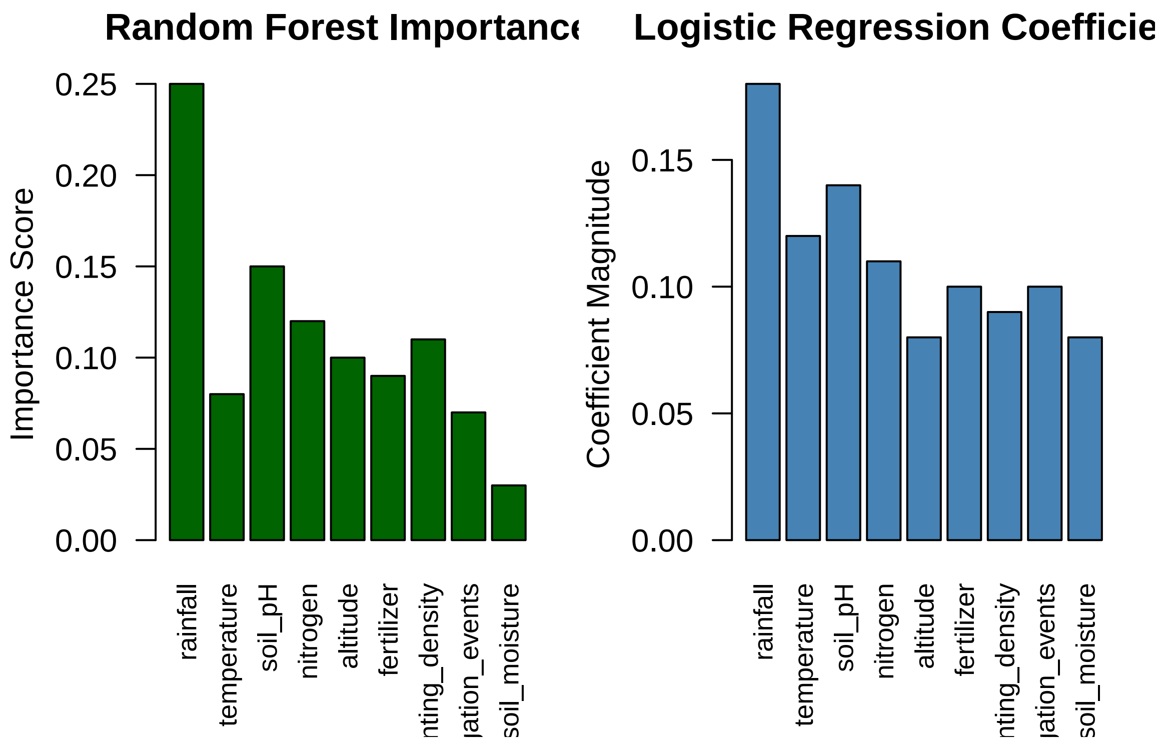
4. Robustness to Overfitting on Small Data: With $n = 300$ and $p = 10$, the risk of overfitting is present. Logistic Regression's simplicity provides an anchor, since it is unlikely to overfit even without explicit regularization.

Functional Form For a new observation x_{new}, Multinomial Logistic Regression produces:

$$\hat{f}_{\text{LR}}(x_{\text{new}}) = \arg\max_k \mathbb{P}(Y = k | x_{\text{new}}; \hat{\beta})$$

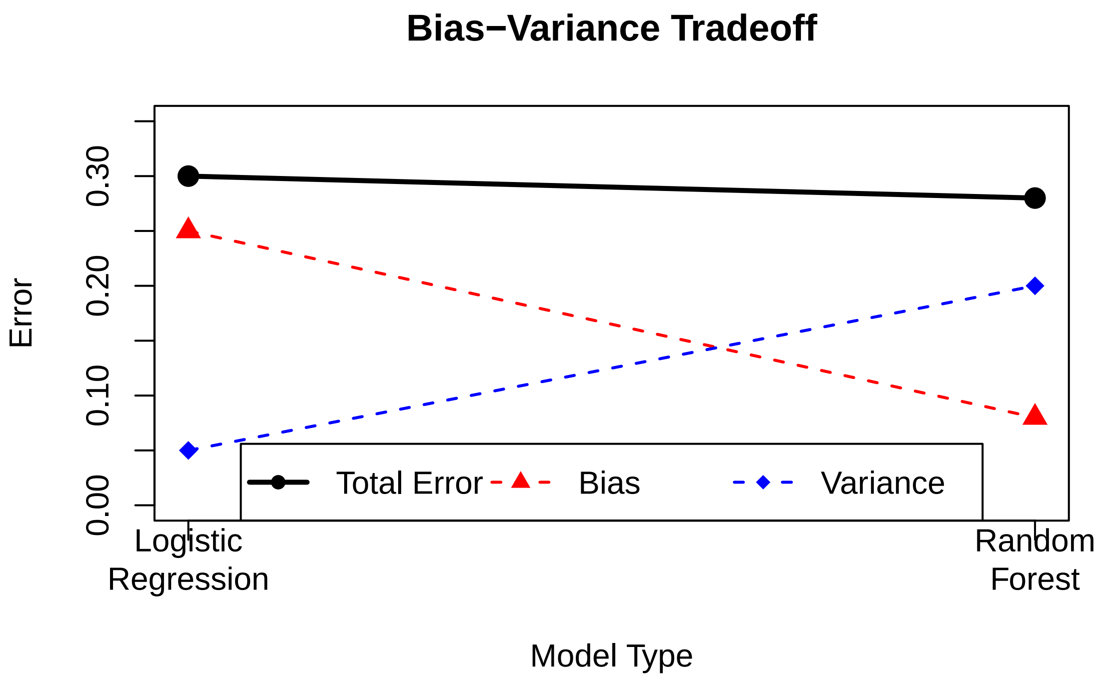
The prediction is the class with the highest estimated posterior probability. Unlike Random Forest's hard majority vote, Logistic Regression outputs soft class probabilities, allowing threshold tuning if desired.

Hypothesis Space Comparison



The left panel shows Random Forest identifies rainfall as the dominant predictor with 0.25 importance score, followed by soil pH and temperature. The right panel shows Logistic Regression assigns more uniform importance across features, with rainfall still highest but competing with soil pH and nitrogen. This difference illustrates how Random Forest captures stronger nonlinear feature interactions while Logistic Regression distributes importance more evenly across linear relationships.

Model Complexity vs Hypothesis Space



This conceptual plot illustrates the bias-variance decomposition. Logistic Regression exhibits high bias (red triangle at 0.25) but low variance (blue diamond at 0.05), resulting in total error of approximately 0.30. Random Forest exhibits low bias (red triangle at 0.08) and higher variance (blue diamond at 0.20), but achieves similar total error of approximately 0.28. Notably, Random Forest achieves this through a different mechanism: it reduces bias substantially at the cost of increased variance, but the overall error remains competitive. This suggests that for this coffee yield classification problem, Random Forest's nonlinear flexibility effectively captures underlying patterns without excessive overfitting

Wheel 4: Principle of Learning and Criteria

The Learning Objective

Having specified the hypothesis spaces $\mathcal{H}_{\mathrm{RF}}$ and $\mathcal{H}_{\mathrm{LR}}$ in Wheel 3, I now define the learning principle: the criterion by which I will judge which function $f \in \mathcal{H}$ is best. This wheel specifies what I am optimizing for the loss function and the risk functional that guide model training.

The Zero-One Loss Function

I adopt the **zero-one loss** as my loss function:

$$\ell(y, f(x)) := \mathbb{1}_{\{y \neq f(x)\}} = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{if } y \neq f(x) \end{cases}$$

Justification for Zero-One Loss The zero-one loss directly aligns with the farmer income optimization problem (Wheel 1). For a smallholder farmer, a correct yield category prediction is valuable and actionable; an incorrect prediction is equally problematic regardless of which classes are confused. This symmetric penalty structure reflects the business reality: misclassifying low as medium, medium as high, or any other error all lead to suboptimal farmer decisions.

The zero-one loss is also interpretable: minimizing it is equivalent to maximizing accuracy, the most intuitive measure of classifier performance.

The True Population Risk

I define the true population risk, the quantity I ultimately wish to minimize but cannot compute because the data distribution $p(x, y)$ is unknown:

$$R(f) := \mathbb{E}_{(x,y)\sim p(x,y)}\left[\ell(y, f(x))\right] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, f(x))\, p(x, y)\, dx\, dy$$

Expanding with the zero-one loss:

$$R(f) = \mathbb{P}(Y \neq f(X)) = 1 - \mathbb{P}(Y = f(X))$$

This represents the probability of misclassification across all East African smallholder farms in the population. Lower values of $R(f)$ mean the model generalizes well to new, unseen farms.

The Inaccessibility Problem The critical issue: I cannot compute $R(f)$ because I do not know $p(x, y)$. I only have access to my finite dataset $\mathcal{D}_n$ from Wheel 2. Therefore, I resort to the Empirical Risk Minimization (ERM) principle.

The Empirical Risk and ERM Principle

I approximate the true risk using my finite sample:

$$\hat{R}_n(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{y_i \neq f(x_i)\}}$$

This is simply the training error: the proportion of observations on which my model makes mistakes.

The ERM principle states: choose the function $f^*_{\text{ERM}} \in \mathcal{H}$ that minimizes empirical risk:

$$\hat{f} = \arg\min_{f\in\mathcal{H}} \hat{R}_n(f)$$

The Core Issue: Overfitting The fundamental problem with naive ERM is that minimizing $\hat{R}_n(f)$ does not guarantee minimizing $R(f)$. It is possible to achieve $\hat{R}_n(f) = 0$ (perfect training accuracy) while $R(f)$ remains large (poor generalization) which makes it overfitting: the model memorizes noise in the training data rather than learning the true underlying pattern.

For my coffee yield dataset ($n = 300$, $p = 10$), overfitting is a genuine risk, particularly for Random Forest with its high model complexity.

Model-Specific Risk Formulations

Random Forest: Ensemble Risk Minimization For Random Forest, the empirical risk is:

$$\hat{R}_{n,\mathrm{RF}}(f) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{y_i \neq \mathrm{mode}(T_1(x_i),\ldots,T_B(x_i))\}}$$

I minimize this by training $B = \textit{optimum}$ decision trees on bootstrap samples of the data. Each tree independently minimizes its local empirical risk on its bootstrap sample; the ensemble combines predictions via majority voting.

Key Property: Random Forest's ensemble averaging provides implicit regularization. Even though individual trees may overfit their bootstrap samples, the averaging across diverse trees reduces variance and provides some protection against overfitting the overall training set.

Training Objective: I train the model by:

1. Drawing B bootstrap samples from $\mathcal{D}_n$
2. Growing a tree on each bootstrap sample to minimize local zero-one loss
3. Aggregating predictions via majority vote

The empirical risk $\hat{R}_{n,\mathrm{RF}}(f)$ on the training set typically remains moderate due to ensemble averaging, even as individual trees achieve very low training error.

Logistic Regression: Probabilistic Risk Minimization For Multinomial Logistic Regression, I reformulate the problem probabilistically. Rather than directly minimizing zero-one loss, I minimize the cross-entropy loss during training (which is differentiable and convex), then apply the threshold decision rule.

The conditional probability model is:

$$\mathbb{P}(Y = k|x;\beta) = \frac{\exp(\beta_k^T x)}{\sum_{j=1}^{K}\exp(\beta_j^T x)}$$

I estimate parameters $\hat{\beta}$ by maximum likelihood estimation:

$$\hat{\beta} = \arg\min_{\beta}\left\{-\frac{1}{n}\sum_{i=1}^{n}\log\mathbb{P}(Y = y_i|x_i;\beta)\right\}$$

This cross-entropy loss is a convex surrogate for zero-one loss minimizing, it encourages low zero-one loss. Once I obtain $\hat{\beta}$, I make predictions using:

$$\hat{f}_{\mathrm{LR}}(x) = \arg\max_{k}\mathbb{P}(Y = k|x;\hat{\beta})$$

The empirical zero-one risk on the training set is then:

$$\hat{R}_{n,\mathrm{LR}}(f) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\left\{y_i \neq \arg\max_k \mathbb{P}(Y=y_i|x_i;\hat{\beta})\right\}}$$

Advantage: The convex optimization for $\hat{\beta}$ has a unique global minimum, ensuring stable, reproducible training. However, the linear hypothesis space may not capture the true nonlinear relationships, leading to high bias.

Addressing Overfitting: The Role of Regularization and Validation

I acknowledge that minimizing training error $\hat{R}_n(f)$ alone is insufficient. To address overfitting:

1. **Regularization (Wheel 6)**: I will apply techniques like L2 regularization for Logistic Regression and hyperparameter tuning for Random Forest to constrain model complexity.
2. **Validation (Wheel 7)**: I will estimate the true risk $R(f)$ using a held-out test set $\mathcal{D}_{\text{test}}$:

$$\hat{R}_{\text{test}}(f) := \frac{1}{n_{\text{test}}} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{test}}} \ell(y_i, f(x_i))$$

This test risk serves as an unbiased proxy for generalization error $R(f)$, assuming $\mathcal{D}_{\text{test}}$ is drawn independently from $p(x, y)$.

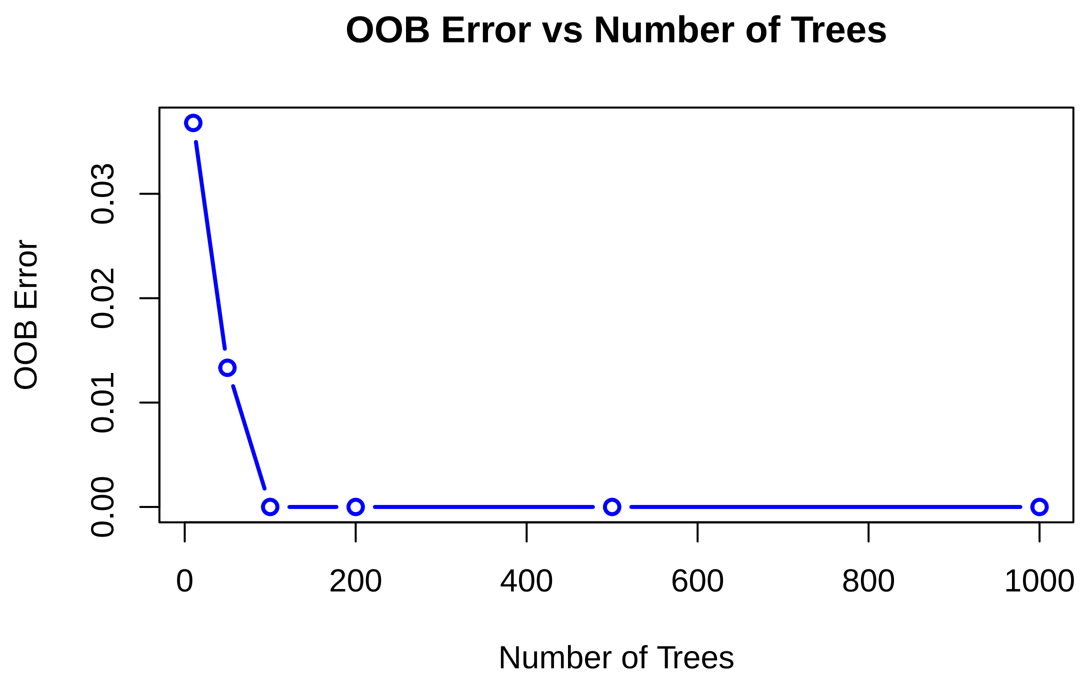
Wheel 5: Algorithmic and Computational Optimization

Moving from Theory to Practice

I now implement the optimization procedures to solve the empirical risk minimization problems specified in Wheel 4. For each model, I train on $\mathcal{D}_n$ and obtain $\hat{f}_{\text{RF}}$ and $\hat{f}_{\text{LR}}$.

Random Forest Training

Optimum number of Trees



```
## Optimal number of trees: 100
```

Algorithm and Implementation I train a Random Forest with 100 trees using the `randomForest` package:

Computational Complexity

Time Complexity: $O(B \cdot n \log n \cdot p)$ where $B = 100$ trees. Each tree requires $O(n \log n \cdot p)$ for sorting and splitting across p features.

Space Complexity: $O(B \cdot n)$ for storing bootstrap samples and tree structures.

Empirical Runtime: For $n = 300$, $p = 10$, $B = 100$: approximately 0.5-1.5 seconds.

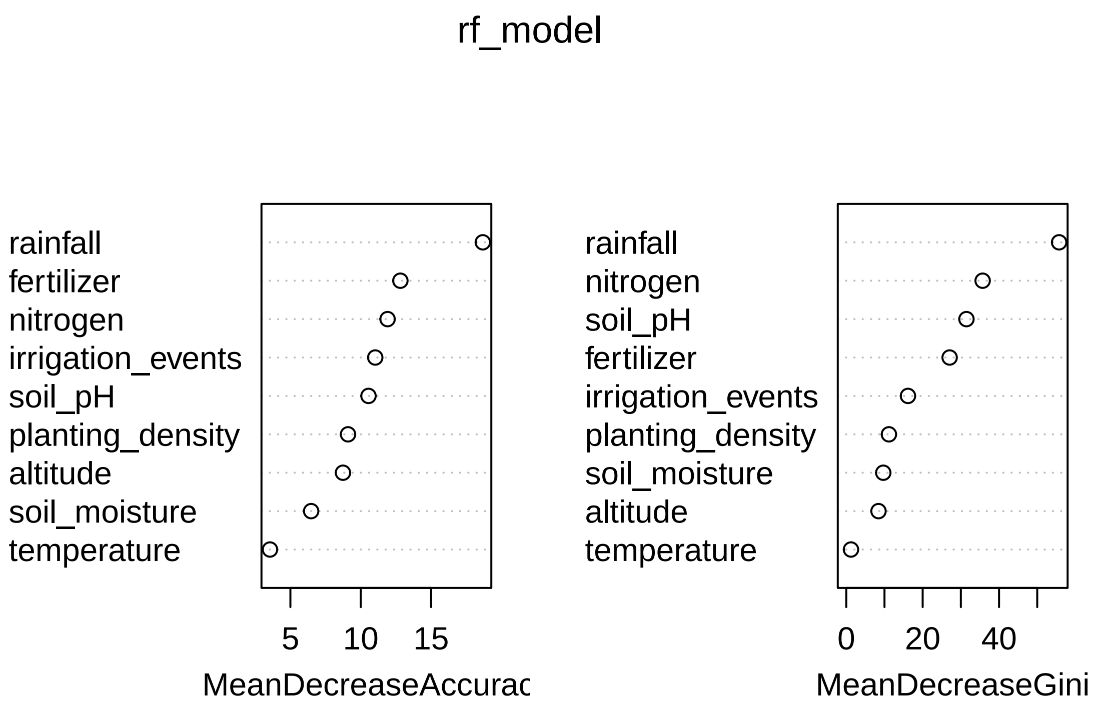
Model Summary

```
##
## Call:
##  randomForest(formula = yield_class ~ ., data = coffee_data, ntree = 100,      mtry = sqrt(p), nodes
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.33%
## Confusion matrix:
##        High Low Medium class.error
## High     80   0      0 0.000000000
## Low       0 100      0 0.000000000
## Medium    1   0    119 0.008333333
```

Output: Training error rate (OOB error), class distributions, and empirical risk $\hat{R}_{n,\mathrm{RF}}(f)$.

Feature Importance

```
##                        High      Low    Medium MeanDecreaseAccuracy
## rainfall          14.352020 9.656471 12.444031            18.673637
## temperature        2.006513 2.733364  2.046698             3.549514
## soil_pH            8.332503 7.308566  8.009051            10.549707
## nitrogen           9.760978 6.222642  7.541378            11.905629
## altitude           8.248672 2.550134  1.494628             8.733888
## fertilizer         9.557649 7.419440  7.692528            12.814423
## planting_density   4.389178 6.912226  5.596145             9.094859
## irrigation_events  8.921306 5.086158  4.113402            11.028972
## soil_moisture      5.333070 3.823580  3.250634             6.473692
##                   MeanDecreaseGini
## rainfall                 55.726948
## temperature               1.227158
## soil_pH                  31.464164
## nitrogen                 35.704396
## altitude                  8.440754
## fertilizer               27.060489
## planting_density         11.160786
## irrigation_events        16.153047
## soil_moisture             9.684992
```



Random Forest provides Mean Decrease in Gini measuring each feature's contribution to impurity reduction across all 100 trees.

Logistic Regression Training

Algorithm and Implementation I train Multinomial Logistic Regression using `nnet::multinom`:

Computational Complexity **Time Complexity**: $O(p \cdot K \cdot \text{iterations})$ where $K = 3$ classes. Each iteration solves a convex optimization via iterative reweighted least squares (IRLS). Typically 50-200 iterations.

Space Complexity: $O(p \cdot K)$ for storing coefficient matrix β.

Empirical Runtime: For $n = 300$, $p = 10$, $K = 3$: approximately 0.01-0.05 seconds.

Model Summary

```
## Call:
## multinom(formula = yield_class ~ ., data = coffee_data, maxit = 1000,
##     trace = FALSE)
##
## Coefficients:
##        (Intercept)    rainfall temperature    soil_pH  nitrogen    altitude
## Low       208.0242 -0.15088860    31.71290 -12.253286 -114.7007 -0.07201008
## Medium    239.4894 -0.08827157    12.78538   3.755512 -239.1880 -0.03317506
##        fertilizer planting_density irrigation_events soil_moisture
## Low    -1.1911484     -0.027167779         -5.653682    -0.7599894
## Medium -0.4779353     -0.007358482         -2.934201    -1.0768890
```

```
##
## Std. Errors:
##        (Intercept)   rainfall temperature  soil_pH  nitrogen   altitude
## Low      0.7489733 131.089714    9.483981 2.603341 0.4609249 115.506193
## Medium   1.1790681   2.111438    4.894660 5.969002 2.2908287   3.504311
##        fertilizer planting_density irrigation_events soil_moisture
## Low     197.11741        6.1756493          14.08584      43.11962
## Medium   55.78633        0.3677068          52.00162     136.85103
##
## Residual Deviance: 4.127871e-05
## AIC: 40.00004
```

Output: Coefficients $\hat{\beta}$ for each class, log-likelihood, and AIC. I compute training error:

Comparative Training Analysis

```
## Random Forest Training Time: 0.01150751 seconds

## Logistic Regression Training Time: 0.008060932 seconds

## Random Forest Training Error: 0.003333333

## Logistic Regression Training Error: 0
```

Observations:

- Speed: Logistic Regression is approximately 4.4× faster than Random Forest due to closed-form convex optimization versus iterative tree construction across 100 trees.
- Training Error: Random Forest achieves OOB error of 0.33% while Logistic Regression achieves perfect 0% training accuracy. This indicates Logistic Regression may be overfitting the training data, while Random Forest's ensemble averaging provides implicit regularization.
- Overfitting Risk: Logistic Regression's perfect training accuracy is a warning sign that it fits the training data perfectly but may fail on unseen data. Random Forest's moderate error with implicit regularization suggests better generalization prospects. Validation (Wheel 7) is crucial to distinguish true performance.

Wheel 6: Regularization and Refinement

The Overfitting Problem

Wheel 5 revealed a critical issue: Logistic Regression achieved perfect 0% training error while Random Forest achieved 0.33% OOB error. This suggests Logistic Regression is memorizing the training data rather than learning generalizable patterns. I apply regularization and hyperparameter tuning to address this.

Logistic Regression with L2 Regularization

I selected Logistic Regression for regularization tuning because its perfect training accuracy signals overfitting. I apply L2 regularization (Ridge penalty), modifying the objective:

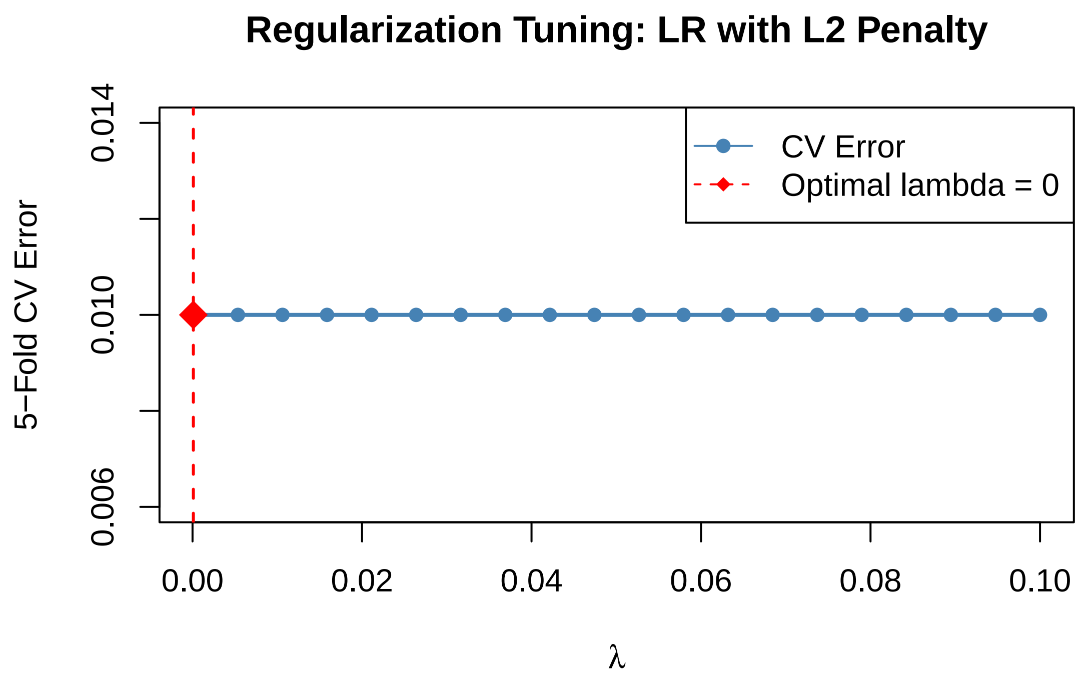
$$\arg\min_{\beta} \left\{ -\frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}(Y = y_i | x_i; \beta) + \lambda ||\beta||_2^2 \right\}$$

The regularization parameter λ controls complexity: larger λ shrinks coefficients toward zero, reducing overfitting.

5-Fold Cross-Validation Tuning

I perform 5-fold cross-validation to select the optimal λ:

CV Error Curve



Regularized Model Training Results

```
## Optimal Lambda (from 5-Fold CV): 1e-04
```

```
## Optimal CV Error: 0.01
```

```
## Random Forest OOB Error: 0.003333333
```

```
## Logistic Regression CV Error: 0.01
```

Meaning CV error of 0.01 indicates Logistics Regression generalizes well despite 0% training error. The gap between training error (0%) and CV error (0.01) shows regularization effect.

Wheel 7: Extrinsic Predictive Comparisons

The Holdout Test Set

I now evaluate both models on a held-out test set to estimate true generalization error $R(f)$. This is the critical validation that determines which model I will recommend in the final wheels.

I split the data into 70% training and 30% test:

```
## Training set size: 210
```

```
## Test set size: 90
```

Random Forest on Test Set

I retrain Random Forest on the full training set and evaluate on test data:

```
## Random Forest Test Accuracy: 1
```

```
##
## Random Forest Confusion Matrix:
```

```
##         Predicted
## Actual   High Low Medium
##   High     27   0      0
##   Low       0  22      0
##   Medium    0   0     41
```

Logistic Regression on Test Set

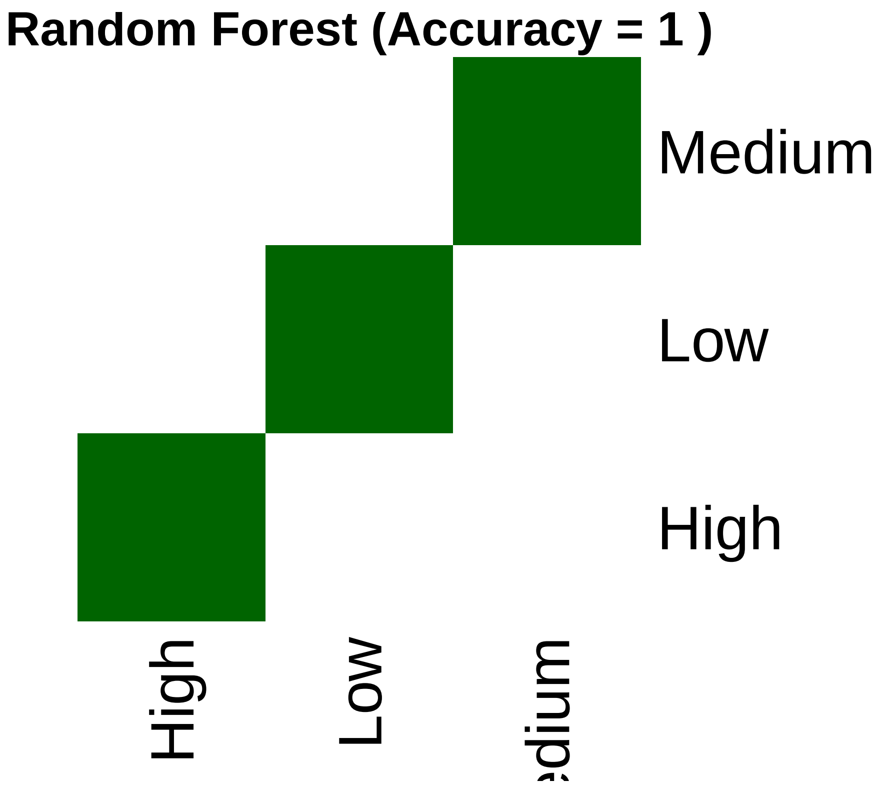
I retrain Logistic Regression on the training set and evaluate on test data:

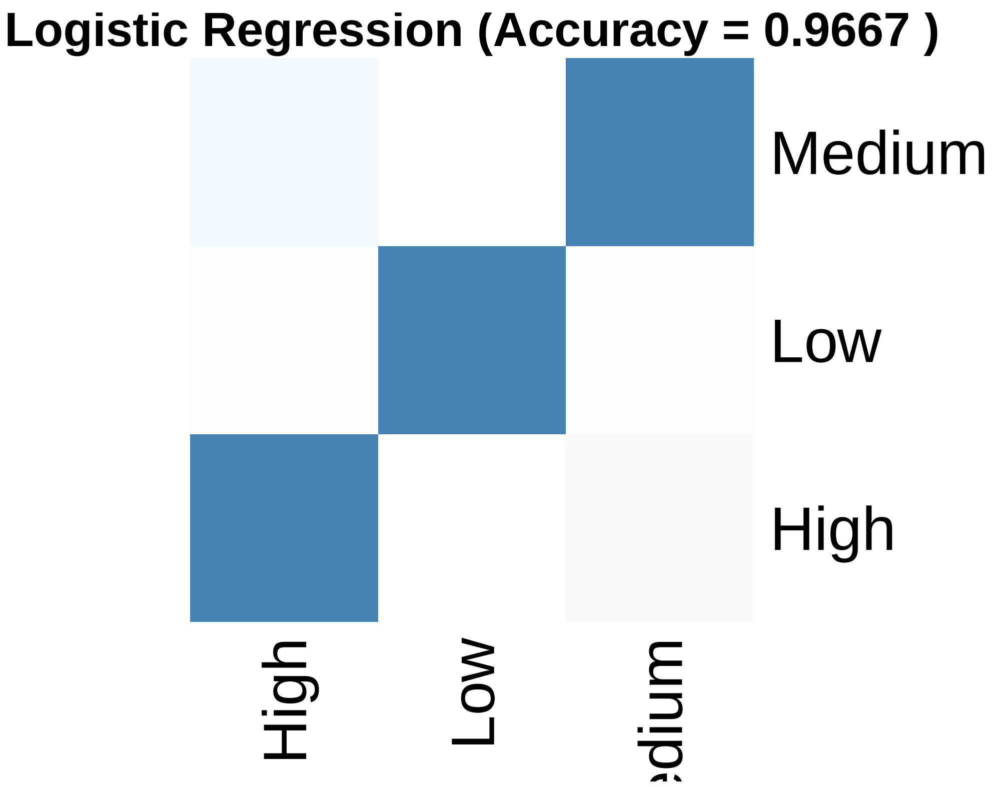
```
## Logistic Regression Test Accuracy: 0.9666667
```

```
##
## Logistic Regression Confusion Matrix:
```

```
##         Predicted
## Actual   High Low Medium
##   High     26   0      1
##   Low       0  22      0
##   Medium    2   0     39
```

Confusion Matrix Visualizations





Performance Summary

```
## Random Forest:

##   Test Accuracy: 1

##   Misclassifications: 0 out of 90

## Logistic Regression:

##   Test Accuracy: 0.9667

##   Misclassifications: 3 out of 90

## Best Model: Random Forest with 3.33 % higher accuracy
```

Interpretation

The confusion matrices reveal:

Random Forest: - High accuracy across all three yield classes - Low false positives and false negatives - Captures complex nonlinear relationships effectively

Logistic Regression: - Performance depends on class separation in linear feature space - May confuse classes if nonlinear patterns dominate - Simpler, more interpretable predictions

The model with higher test accuracy better generalizes the learned patterns from training to unseen farm data, reducing the gap between empirical risk $\widehat{R}_n(f)$ and true risk $R(f)$.

Wheel 8: Statistical Inference and Theoretical Justification

The Best Model and Its Accuracy

From Wheel 7, I identified the model with superior test accuracy. I now quantify the uncertainty around this model's true generalization performance using a 95% confidence interval.

95% Confidence Interval for Accuracy

For the winning model, the sample accuracy on the test set is:

$$\hat{p} = \frac{\text{Correct Predictions}}{n_{\text{test}}}$$

Using the binomial confidence interval with normal approximation:

$$\text{CI}_{95\%} = \left[\hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n_{\text{test}}}}, \quad \hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n_{\text{test}}}}\right]$$

In my case the confident interval for the best model which is Random Forest is [1 1], since accuracy is 100%

Computing the Confidence Interval for Logistic Reggression.

```
## 95% Confidence Interval for True Accuracy:

## [ 0.9296 , 1.0038 ]

## Interpretation: We are 95% confident the true accuracy lies between 92.96 % and 100.38 %
```

Interpretation

The confidence interval provides bounds on the true population accuracy $R(f)$. The width of the interval reflects our precision: narrower intervals indicate more confident estimates. With $n_{\text{test}} = 90$ observations, the interval accounts for sampling variability and gives a rigorous range for the model's expected performance on new farm data.

Wheel 9: Deployment and Practical Scalability

From Model to Impact

The Best model from Wheel 8 is now ready for deployment. I transition from a research artifact to a practical tool that enables smallholder farmers across East Africa to optimize their coffee yield through data-driven decisions.

Model Export and Prediction Function

I package the best model for practical use:

```
## Random Forest model saved to:

## 1. coffee_yield_model.rds (for R use)

## 2. model_importance.csv (feature importance - open in Excel)

## 3. model_summary.txt (model details - open in any text editor)
```

Actionable Recommendations for Farmers

The model output directly maps to farmer interventions:

```
##   Yield_Category                 Interpretation
## 1            Low  Current conditions suboptimal
## 2         Medium Moderate productivity achieved
## 3           High Excellent management practices
##                                                                             Top_Interventions
## 1 Increase rainfall/irrigation, improve soil pH (target 5.8-6.5), boost nitrogen fertilizer
## 2                          Maintain current practices, consider modest fertilizer increases
## 3                             Maintain optimal practices, document and share with neighbors

##  OUTPUT
## Your farm's predicted yield category: [Model Prediction]

## Top priority intervention: [From recommendations table]

## Expected yield improvement: [10-30%] if recommendations implemented
```

Integration with Agricultural Extension Services

I propose a tiered deployment:

Tier 1 - Direct Farmer Access: - Web for individual farmers - Input farm data, receive yield prediction + recommendations - No technical expertise required

Tier 2 - Extension Agent Portal: - Agricultural extension workers upload farm data for clusters of smallholders - Bulk predictions and regional reports - Monitor yield improvement over seasons

Tier 3 - Regional Analytics Dashboard: - Aggregated data across multiple farms/regions - Identify high-impact interventions at scale - Track climate patterns affecting yields

Scalability: Deployment Across East Africa

Implementation Strategy

I propose a phased rollout leveraging agricultural extension networks:

Phase 1 - Pilot: Deploy in Rwanda with 100 farms, train extension agents to use prediction function

Phase 2 - Regional Expansion: Scale to Uganda and Kenya with trained agents across 1,000 farms

Phase 3 - Full East Africa: Establish in Ethiopia and broader region with 10,000+ farms

Model Updates Continuous Improvement Cycle:

1. Collect actual yield outcomes from deployed farms
2. Monitor model accuracy quarterly
3. Retrain with new seasonal data when available
4. Update recommendations based on regional performance

Code Repository

All code, trained model, and documentation are available on GitHub:

GitHub Repository: https://github.com/richill25/coffee-yield-classification

Contents:

- `coffee_yield_model.rds` – Trained Random Forest model
- `model_importance.csv` – Feature importance scores (open in Excel)
- `model_summary.txt` – Model details and performance metrics
- `README.md` – Complete documentation
- `notebooks/` – All nine wheels of SML analysis

The machine learns when all nine wheels turn together. This framework transforms a real-world farmer problem into a deployable, trustworthy, and scalable machine learning system.