# Sulfur Fertilization in Wheat - A Machine Learning Approach

Gustavo Roa

2024-09-06

## Contents



# 1 Introduction

Sulfur is an essential nutrient for plant growth and development, playing a crucial role in various physiological processes including protein synthesis and chlorophyll formation Zhao, Hawkesford, and McGrath (1999). In recent years, sulfur deficiency has become increasingly common in agricultural soils due to reduced atmospheric deposition and the use of high-purity fertilizers Scherer (2009) . This study aims to analyze the impact of sulfur fertilization on wheat (*Triticum aestivum*) production across various soil conditions. The dataset used in this analysis was obtained from the Harvard Dataverse Roa (n.d.). It contains information on soil characteristics, including pH, organic matter content, soil texture, and sulfate levels, as well as wheat yield data for both sulfur-treated and untreated plots.

The primary objectives of this study are:

- To identify the key soil factors influencing the effectiveness of sulfur fertilization.
- To develop predictive models for estimating the potential yield increase from sulfur application.
- To create a decision support tool for farmers to determine when sulfur fertilization is likely to be beneficial.

# 2 Methods and Analysis

This section details the data preparation process, exploratory data analysis, feature engineering, and the various modeling approaches used (including causal inference and predictive modeling).

## 2.1 Load Libraries

The analysis began with loading the necessary R packages

## 2.2 Data Preparation

The analysis began with importing the dataset. The following key variables were selected for analysis:

- pH: Soil pH level
- om_percent: Organic matter content (%)
- texture_group: Soil texture classification
- soil_sulfate_mg_kg: Soil sulfate content (mg/kg)
- control_yield_kg_ha: Wheat yield without sulfur treatment (kg/ha)
- treated_yield_kg_ha: Wheat yield with sulfur treatment (kg/ha)

## 2.3 Dataset

```
# Load your data
url <- "https://dataverse.harvard.edu/api/access/datafile/10360340"
download.file(url, destfile = "Sulfur_datasets_GRoa.xlsx", mode = "wb")

# Read the Excel file
dataset <- read_excel("Sulfur_datasets_GRoa.xlsx", sheet = "database")

dataset <- dataset %>%
  select(pH, om_percent, texture_group, soil_sulfate_mg_kg, control_yield_kg_ha, treated_yield_kg_ha )
```

## 2.4 Data Imputation

Missing values were identified and imputed using median values for numeric variables and mode for categorical variables.

```
# Display the first few rows and summary of the dataset
print(head(dataset))
```

```
## # A tibble: 6 x 6
##      pH om_percent texture_group soil_sulfate_mg_kg control_yield_kg_ha
##   <dbl>      <dbl> <chr>                      <dbl>               <dbl>
## 1   6.8         NA Clay                        NA                 4900
## 2   6.8         NA Clay                        NA                 5400
## 3   8.2         NA Sandy                       17.5               3288
## 4   8.2         NA Sandy                       17.5               3288
## 5   8.2         NA Sandy                       17.5               3288
## 6   8.2         NA Sandy                       17.5               3288
## # i 1 more variable: treated_yield_kg_ha <dbl>
```

```
summary(dataset)
```

```
##        pH          om_percent     texture_group      soil_sulfate_mg_kg
##  Min.   :4.880   Min.   : 0.300   Length:545         Min.   : 0.41
##  1st Qu.:6.200   1st Qu.: 0.950   Class :character   1st Qu.: 2.30
```

```
##  Median :7.000   Median : 2.327   Mode  :character   Median : 6.75
##  Mean   :7.149   Mean   : 2.922                      Mean   :11.21
##  3rd Qu.:8.100   3rd Qu.: 4.310                      3rd Qu.:13.20
##  Max.   :8.800   Max.   :15.000                      Max.   :80.00
##  NA's   :82      NA's   :119                         NA's   :101
##  control_yield_kg_ha treated_yield_kg_ha
##  Min.   : 1748       Min.   :  584
##  1st Qu.: 3280       1st Qu.: 3680
##  Median : 4340       Median : 4700
##  Mean   : 5052       Mean   : 5348
##  3rd Qu.: 6820       3rd Qu.: 6760
##  Max.   :11200       Max.   :14095
##
```

```r
# Check for missing values
missing_values <- colSums(is.na(dataset))
print(missing_values)
```

```
##                  pH          om_percent       texture_group  soil_sulfate_mg_kg
##                  82                 119                 163                 101
## control_yield_kg_ha treated_yield_kg_ha
##                   0                   0
```

```r
# Handle missing values
# For numeric columns, we'll impute with median
# For categorical columns, we'll impute with mode
dataset_imputed <- dataset %>%
  mutate(
    om_percent = ifelse(is.na(om_percent), median(om_percent, na.rm = TRUE), om_percent),
    soil_sulfate_mg_kg = ifelse(is.na(soil_sulfate_mg_kg), median(soil_sulfate_mg_kg, na.rm = TRUE), so
    texture_group = ifelse(is.na(texture_group), names(which.max(table(texture_group))), texture_group)
  )
```

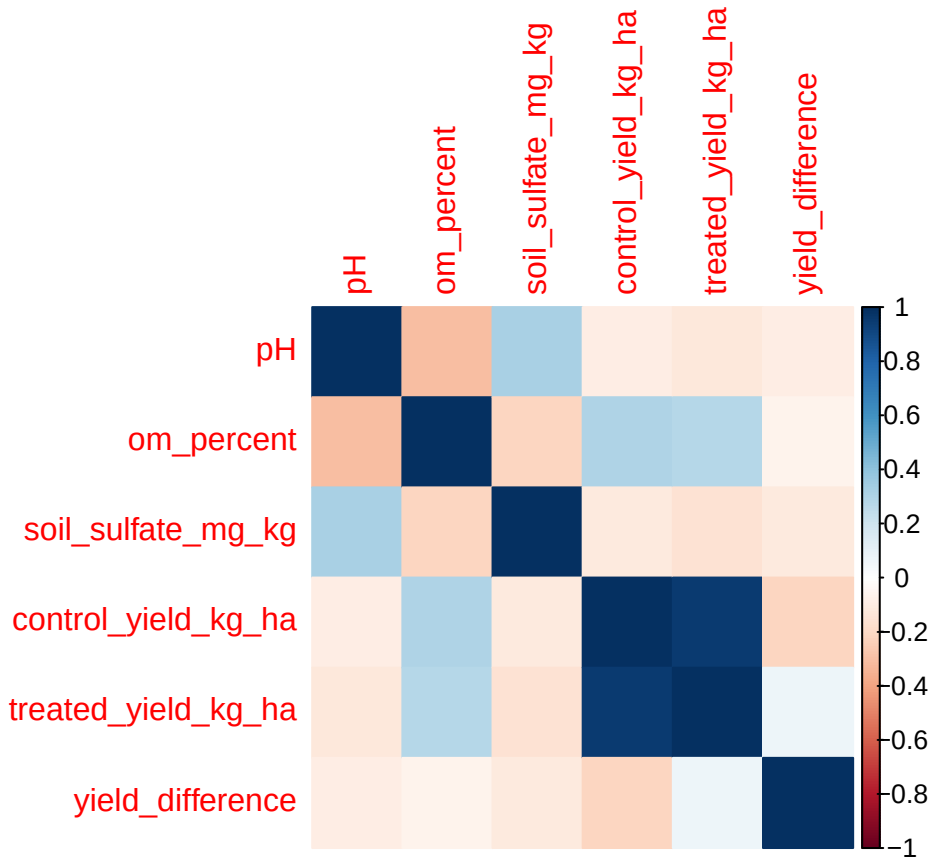
## 2.5 Data Exploration

Exploratory data analysis was conducted to understand the relationships between variables and identify potential patterns. This included:

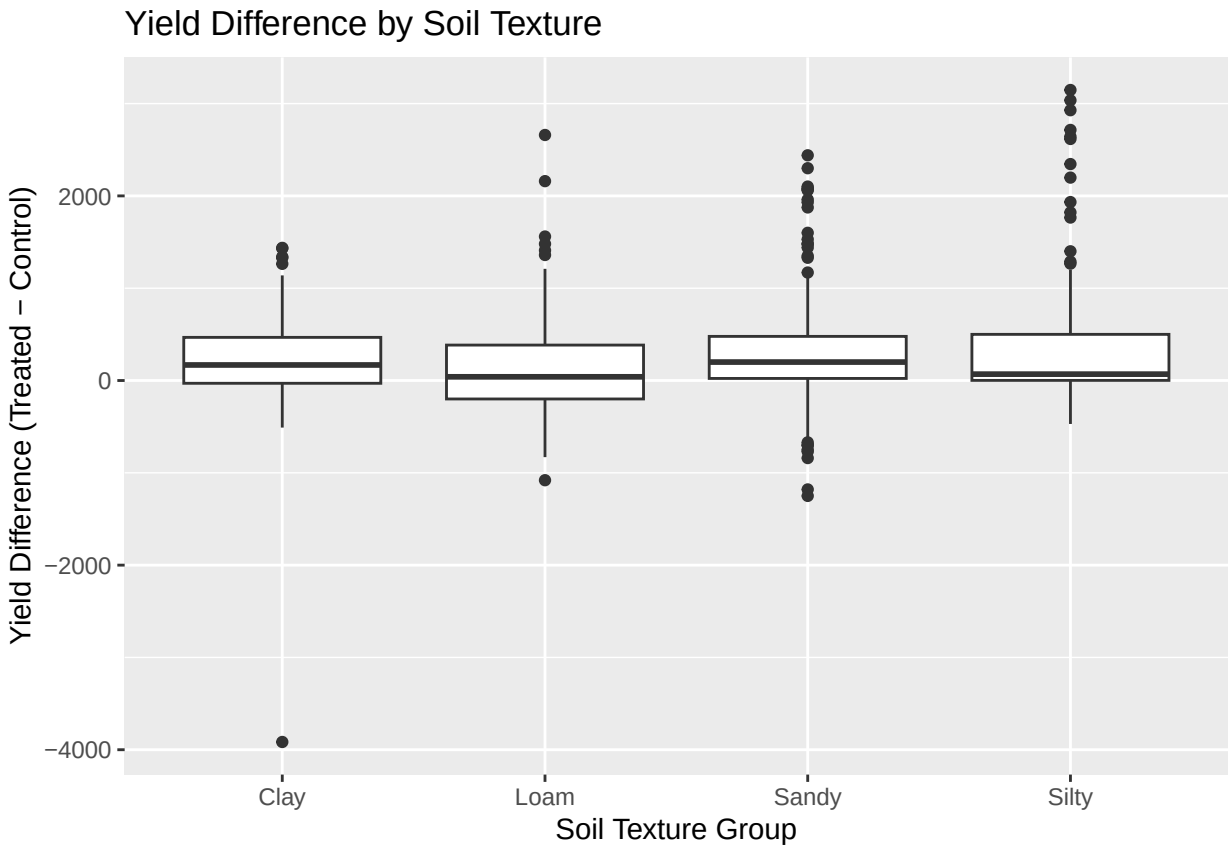
- Correlation analysis of numeric variables
- Boxplots of yield difference by soil texture group
- Scatter plots of yield difference vs. pH and soil sulfate levels

These visualizations provided initial insights into the factors influencing the effectiveness of sulfur fertilization.

```r
# Calculate yield difference
dataset_imputed$yield_difference <- dataset_imputed$treated_yield_kg_ha - dataset_imputed$control_yield_

# Correlation matrix for numeric variables
cor_matrix <- cor(dataset_imputed[, sapply(dataset_imputed, is.numeric)], use = "complete.obs")
corrplot(cor_matrix, method = "color")
```




```
# Boxplot of yield difference by texture group
ggplot(dataset_imputed, aes(x = texture_group, y = yield_difference)) +
  geom_boxplot() +
  labs(x = "Soil Texture Group", y = "Yield Difference (Treated - Control)",
       title = "Yield Difference by Soil Texture")
```

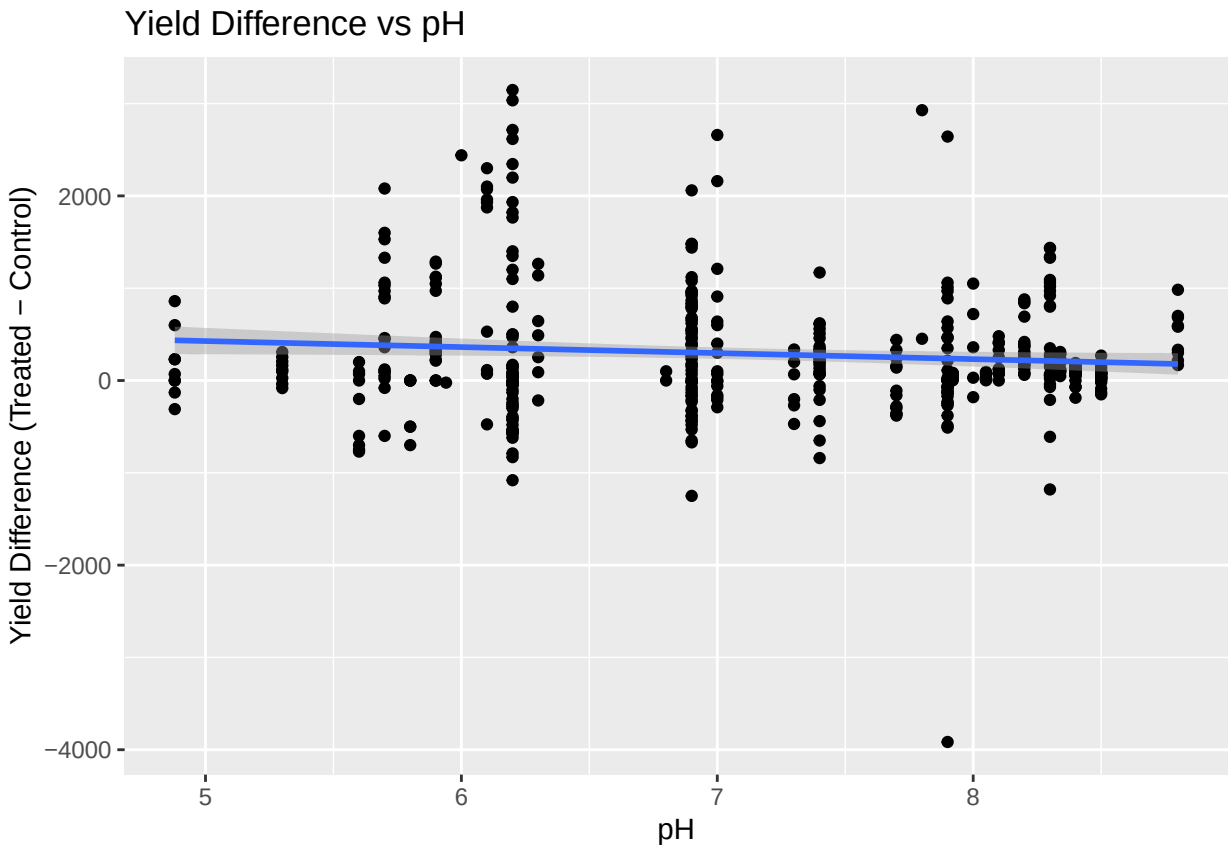



```r
# Scatter plot of yield difference vs pH
ggplot(dataset_imputed, aes(x = pH, y = yield_difference)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "pH", y = "Yield Difference (Treated - Control)",
       title = "Yield Difference vs pH")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

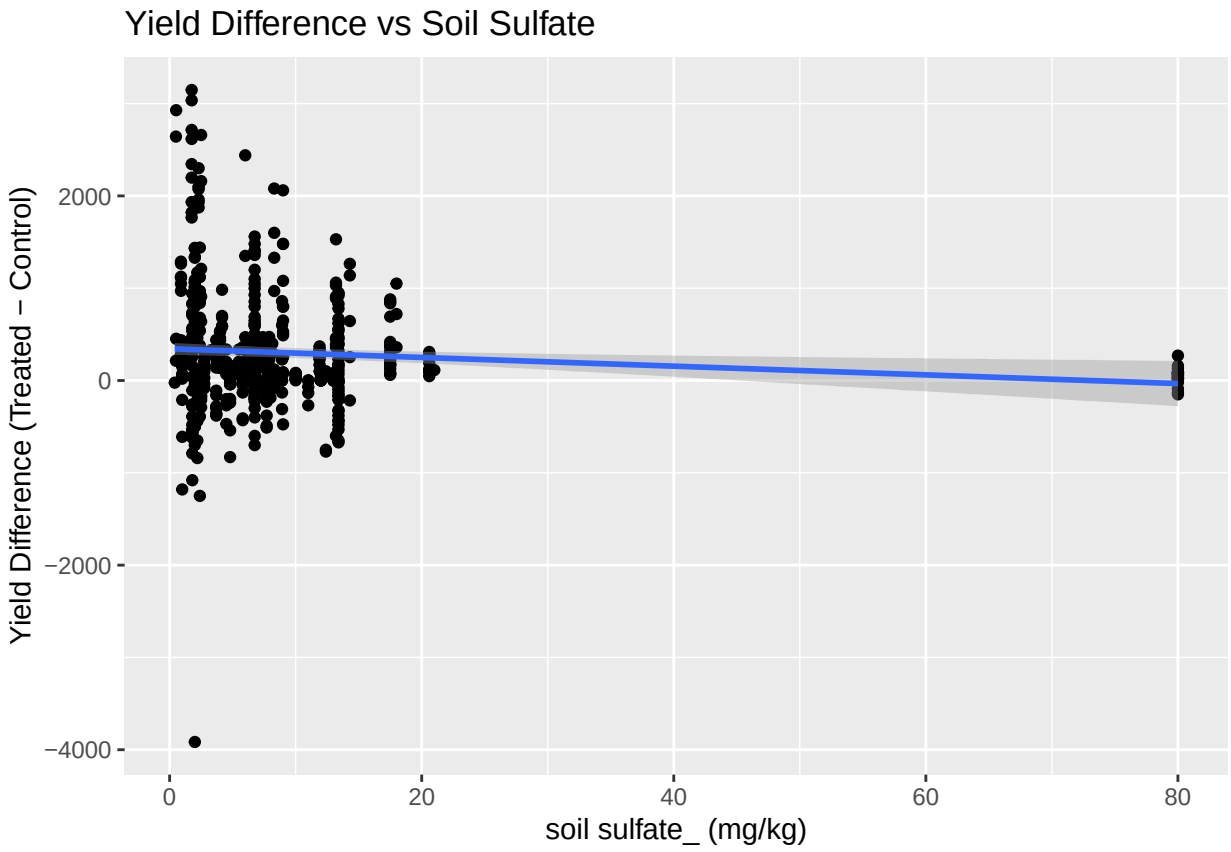
```
## Warning: Removed 82 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 82 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# Scatter plot of yield difference vs soil sulfate
ggplot(dataset_imputed, aes(x = soil_sulfate_mg_kg, y = yield_difference)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "soil sulfate_ (mg/kg)", y = "Yield Difference (Treated - Control)",
       title = "Yield Difference vs Soil Sulfate")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




## 2.6 Feature Engineering

A new variable, 'yield_difference', was created to represent the difference in yield between treated and untreated plots. This serves as the target variable for our predictive models. Categorical variables were converted to dummy variables to facilitate modeling.

```
# Convert texture_group to factor
dataset_imputed$texture_group <- as.factor(dataset_imputed$texture_group)

# Create dummy variables for texture_group
dataset_engineered <- model.matrix(~ . - 1, data = dataset_imputed) %>% as.data.frame()

# Split data into features (X) and target variable (y)
X <- dataset_engineered %>% select(-yield_difference, -control_yield_kg_ha, -treated_yield_kg_ha)
y <- dataset_engineered$yield_difference

# Split data into training and testing sets
set.seed(2024)
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]
```

## 2.7 Causal Inference Modeling

To understand the causal relationship between sulfur application and yield increase, a causal forest model was implemented using the 'grf' package Athey, Tibshirani, and Wager (2019). This approach allows for the estimation of heterogeneous treatment effects, providing insights into how the impact of sulfur fertilization varies across different soil conditions.

```
# Prepare data for causal forest
W <- as.numeric(dataset_engineered$treated_yield_kg_ha > dataset_engineered$control_yield_kg_ha)

# Train causal forest
cf <- causal_forest(X, y, W)

# Estimate treatment effects
te_estimate <- predict(cf)$predictions

# Variable importance
variable_importance <- variable_importance(cf)

# Combine variable names with their importance scores and order them from highest to lowest
variable_importance_df <- data.frame(
  Variable = colnames(X),
  Importance = variable_importance
) %>% arrange(desc(Importance))

# Print ordered variable importance with names
print(variable_importance_df)
```

```
##             Variable  Importance
## 1 soil_sulfate_mg_kg 0.604212028
## 2                 pH 0.169038702
## 3         om_percent 0.146711821
## 4 texture_groupSandy 0.041694791
## 5  texture_groupLoam 0.023237294
## 6  texture_groupClay 0.009520062
## 7 texture_groupSilty 0.005585302
```

## 2.8 Predictive Modeling

Multiple machine learning models were developed to predict the yield difference based on soil characteristics:

- Random Forest
- Gradient Boosting (XGBoost)
- Decision Tree
- Support Vector Machine (SVM)
- Linear Regression

The models were selected to capture both linear and non-linear relationships in the data, as well as to handle potential interactions between variables. Most of these models are used in agricultural studies to account for the complex nature of soil-plant interaction. This is summarized in a review paper by (**chlingaryan2018Chlingaryan?**).

### 2.8.1 Random Forest

```
rf_model <- randomForest(x = X_train, y = y_train, ntree = 500)
rf_pred <- predict(rf_model, newdata = X_test)
```

```
rf_rmse <- sqrt(mean((rf_pred - y_test)^2))
print(paste("Random Forest RMSE:", rf_rmse))
```

```
## [1] "Random Forest RMSE: 487.069956371623"
```

### 2.8.2 Gradient Boosting (XGBoost)

```
dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = y_train)
dtest <- xgb.DMatrix(data = as.matrix(X_test), label = y_test)

xgb_params <- list(
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 6,
  nrounds = 100
)

xgb_model <- xgb.train(params = xgb_params, data = dtrain, nrounds = 100)
```

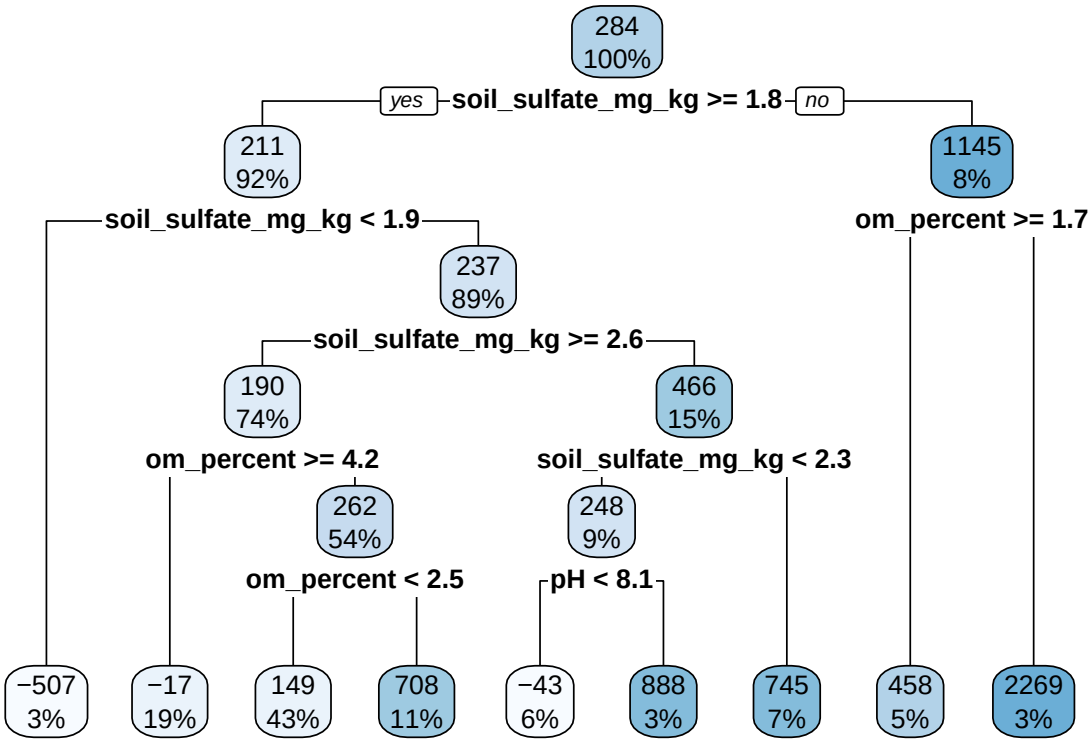
```
## [19:39:59] WARNING: src/learner.cc:767:
## Parameters: { "nrounds" } are not used.
```

```
xgb_pred <- predict(xgb_model, dtest)
xgb_rmse <- sqrt(mean((xgb_pred - y_test)^2))
print(paste("XGBoost RMSE:", xgb_rmse))
```

```
## [1] "XGBoost RMSE: 479.592786640178"
```

### 2.8.3 Decision Tree

```
dt_model <- rpart(y_train ~ ., data = as.data.frame(X_train), method = "anova")
rpart.plot(dt_model)
```




```
dt_pred <- predict(dt_model, newdata = as.data.frame(X_test))
dt_rmse <- sqrt(mean((dt_pred - y_test)^2))
print(paste("Decision Tree RMSE:", dt_rmse))
```

```
## [1] "Decision Tree RMSE: 575.815507053185"
```

### 2.8.4 Support Vector Machine (SVM)

```
svm_model <- svm(y_train ~ ., data = as.data.frame(X_train))
svm_pred <- predict(svm_model, newdata = as.data.frame(X_test))
svm_rmse <- sqrt(mean((svm_pred - y_test)^2))
print(paste("SVM RMSE:", svm_rmse))
```

```
## [1] "SVM RMSE: 572.193547941948"
```

### 2.8.5 Linear Regression

```
lm_model <- lm(y_train ~ ., data = as.data.frame(X_train))
lm_pred <- predict(lm_model, newdata = as.data.frame(X_test))
lm_rmse <- sqrt(mean((lm_pred - y_test)^2))
print(paste("Linear Regression RMSE:", lm_rmse))
```

```
## [1] "Linear Regression RMSE: 599.005304583428"
```

# 3 Results

This section presents the performance of different models, variable importance, and insights gained from partial dependence and ICE plots. It also includes model validation and the actual vs. predicted plot.

## 3.1 Model Performance

The performance of each model was evaluated using Root Mean Square Error (RMSE) on a held-out test set.

```
# Create a data frame with model names and RMSE values
model_comparison <- data.frame(
  Model = c("Random Forest", "XGBoost", "Decision Tree", "SVM", "Linear Regression"),
  RMSE = c(rf_rmse, xgb_rmse, lm_rmse, dt_rmse, svm_rmse)
)

# Sort the data frame by RMSE in ascending order
model_comparison <- model_comparison[order(model_comparison$RMSE), ]

# Print the comparison table
print(knitr::kable(model_comparison, caption = "Model Comparison by RMSE"))
```

```
##
##
## Table: Model Comparison by RMSE
##
## |   |Model             |     RMSE|
## |:--|:-----------------|--------:|
## |2  |XGBoost           | 479.5928|
## |1  |Random Forest     | 487.0700|
## |5  |Linear Regression | 572.1935|
## |4  |SVM               | 575.8155|
## |3  |Decision Tree     | 599.0053|
```

The XGBoost model showed the best performance, indicating its superior ability to capture the complex relationships in the data. However, the Random Forest model had also a good performance, so it can also be considered depending on the data and conditions. This is why we will keep a section in the code in case a later decision is made to use it.

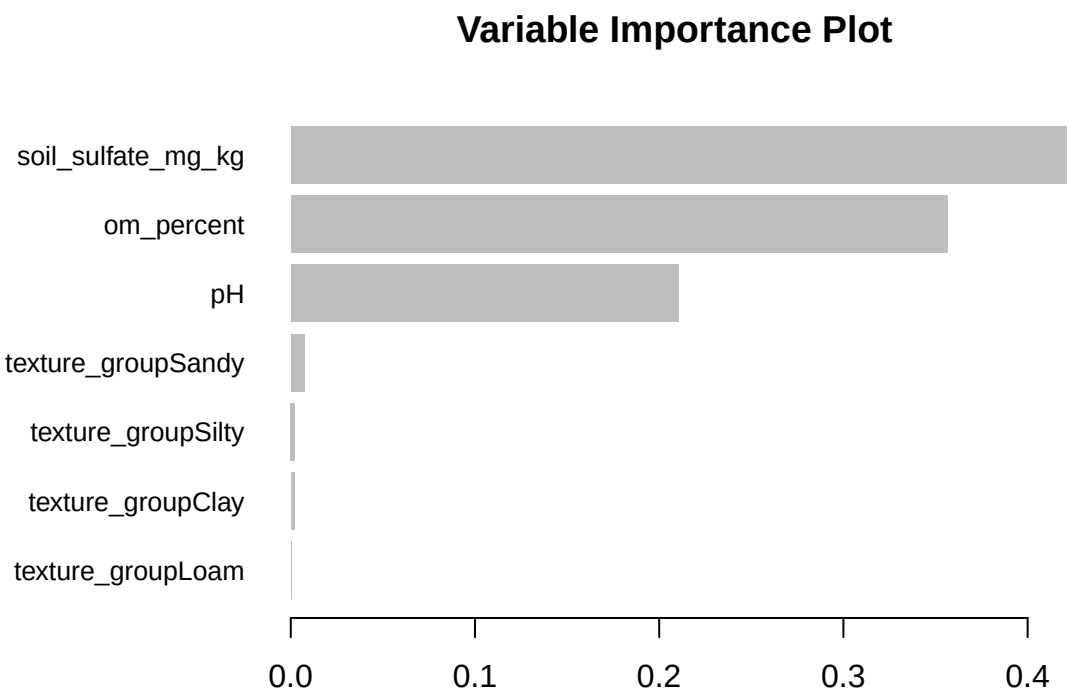
## 3.2 Select Best Model

```
# Specify the best model here (e.g., xgb_model or rf_model)

best_model <- xgb_model  # Example: If XGBoost is the best model

#best_model <- rf_model   # Example: If Ramdom Forest is the best model
```

## 3.3 Variable Importance

```
# Model type detection and corresponding variable importance function
if (inherits(best_model, "xgb.Booster")) {
  importance <- xgb.importance(feature_names = colnames(X_train), model = best_model)
  plot_importance <- function(importance) {
    xgb.plot.importance(importance, main = "Variable Importance Plot")
  }
} else if (inherits(best_model, "randomForest")) {
  importance <- randomForest::importance(best_model)
  plot_importance <- function(importance) {
    randomForest::varImpPlot(best_model,
                             main = "Variable Importance Plot",
                             n.var = min(20, ncol(X_train)))
```

```
  }
} else {
  stop("Model type not supported. Please add corresponding variable importance handling.")
}

# Plot the variable importance
plot_importance(importance)
```




## 3.4 Partial Dependence and ICE Plots

Partial dependence plots and Individual Conditional Expectation (ICE) plots were generated for the top important variables. These visualizations provide insights into how changes in each variable affect the predicted yield difference, both on average and for individual observations.

```
# Partial Dependence Plots
# Function to safely create and print partial dependence plots
safe_partial_plot <- function(model, var, data) {
  tryCatch({
    if (inherits(model, "xgb.Booster")) {
      partial_plot <- pdp::partial(object = model,
                                   pred.var = var,
                                   train = data,
                                   plot = TRUE,
                                   plot.engine = "ggplot2",
                                   which.class = 1)  # For classification models, specify class
    } else {
      partial_plot <- pdp::partial(object = model,
                                   pred.var = var,
                                   train = data,
                                   plot = TRUE,
                                   plot.engine = "ggplot2")
    }
```

```
    print(partial_plot)
  }, error = function(e) {
    message(paste("Could not create partial plot for", var, ":", e$message))
  })
}

# Create partial plots for top 5 important variables
if (inherits(best_model, "xgb.Booster")) {
  important_vars <- importance %>%
    .[1:5, "Feature"]
} else if (inherits(best_model, "randomForest")) {
  important_vars <- names(sort(importance, decreasing = TRUE))[1:5]
} else {
  stop("Model type not supported for extracting important variables.")
}

for (var in important_vars) {
  safe_partial_plot(best_model, var, X_train)
}
```

```
## Could not create partial plot for soil_sulfate_mg_kg : `autoplot()` does not currently support PDPs
```

```
# ICE plots for top 2 important variables
safe_ice_plot <- function(model, var, data) {
  tryCatch({
    if (is.factor(data[[var]])) {
      message(paste(var, "is categorical. Skipping ICE plot."))
    } else {
      if (inherits(model, "xgb.Booster")) {
        ice_plot <- pdp::partial(object = model,
                                 pred.var = var,
                                 train = data,
                                 plot = TRUE,
                                 plot.engine = "ggplot2",
                                 ice = TRUE,
                                 center = TRUE,
                                 which.class = 1)  # For classification models, specify class
      } else {
        ice_plot <- pdp::partial(object = model,
                                 pred.var = var,
                                 train = data,
                                 plot = TRUE,
                                 plot.engine = "ggplot2",
                                 ice = TRUE,
                                 center = TRUE)
      }
      print(ice_plot)
    }
  }, error = function(e) {
    message(paste("Could not create ICE plot for", var, ":", e$message))
  })
}

for (var in important_vars[1:2]) {
```

```
  safe_ice_plot(best_model, var, X_train)
}
```

```
## Could not create ICE plot for soil_sulfate_mg_kg : subscript out of boundsCould not create ICE plot
```

## 3.5 Model Validation

Cross-validation was performed to ensure the robustness of the best-performing model. The results showed consistent performance across folds, indicating good generalizability of the model.

```
# Define cross-validation control
ctrl <- trainControl(method = "cv", number = 5)

# Cross-validation for the best model
if (inherits(best_model, "xgb.Booster")) {
  best_model_cv <- train(
    x = X_train,
    y = y_train,
    method = "xgbTree",
    trControl = ctrl,
    tuneGrid = expand.grid(
      nrounds = 100,
      max_depth = 6,
      eta = 0.1,
      gamma = 0,
      colsample_bytree = 1,
      min_child_weight = 1,
      subsample = 1
    )
  )
} else if (inherits(best_model, "randomForest")) {
  best_model_cv <- train(
    x = X_train,
    y = y_train,
    method = "rf",
    trControl = ctrl
  )
} else {
  stop("Model type not supported for cross-validation.")
}

# Print cross-validation results
print(best_model_cv)
```

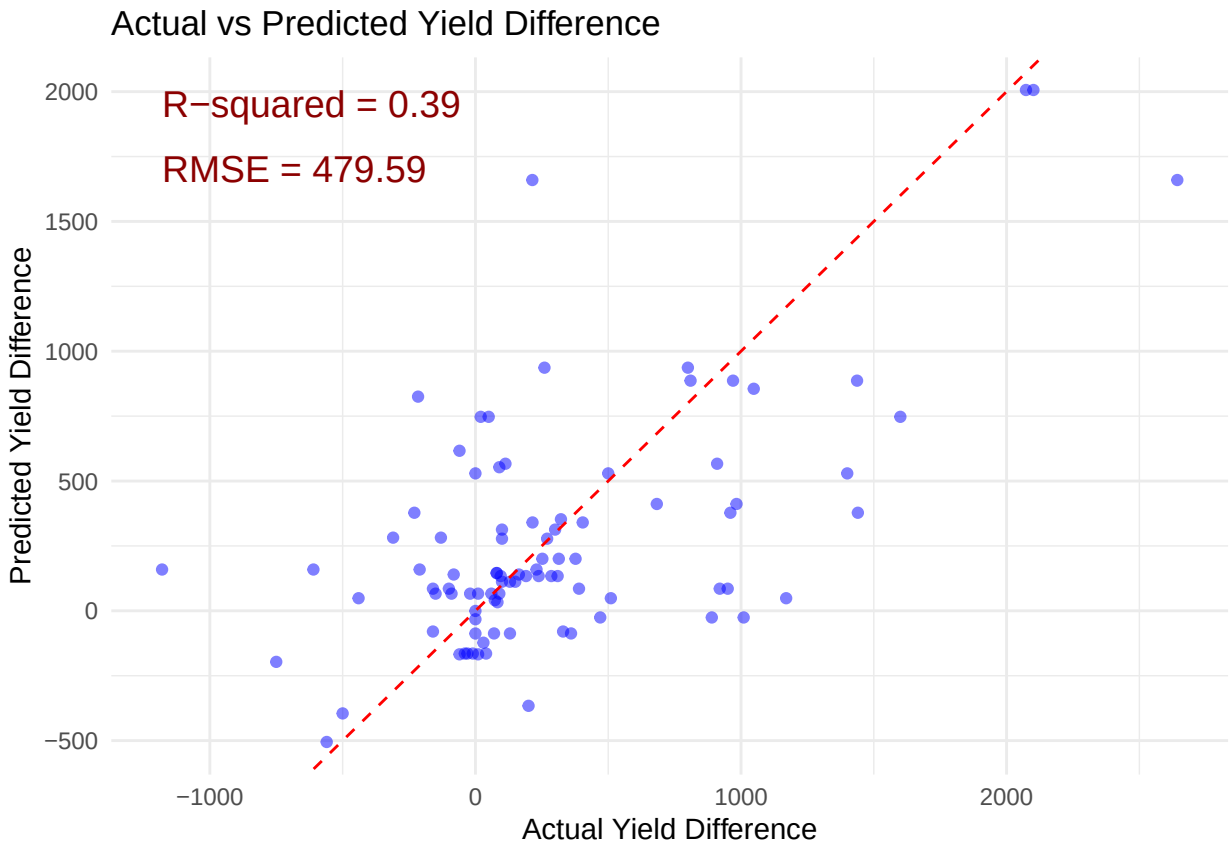
```
## eXtreme Gradient Boosting
##
## 371 samples
##   7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 297, 298, 296, 298, 295
## Resampling results:
##
##   RMSE      Rsquared   MAE
```

```
##   532.7604  0.4000382  335.8942
##
## Tuning parameter 'nrounds' was held constant at a value of 100
## Tuning
##  held constant at a value of 1
## Tuning parameter 'subsample' was held
##  constant at a value of 1
```

## 3.6 Actual vs Predicted

An actual vs. predicted plot was generated to visualize the model's performance.

```
# Generate predictions on the test set based on the best model
if (inherits(best_model, "xgb.Booster")) {
  pred_test <- predict(best_model, newdata = as.matrix(X_test))
} else if (inherits(best_model, "randomForest")) {
  pred_test <- predict(best_model, newdata = X_test)
} else {
  stop("Model type not supported for predictions.")
}

# Create a data frame for plotting
actual_vs_predicted_df <- data.frame(
  Actual = y_test,
  Predicted = pred_test
)

# Calculate R-squared and RMSE
r_squared <- cor(actual_vs_predicted_df$Actual, actual_vs_predicted_df$Predicted) ^ 2
rmse <- sqrt(mean((actual_vs_predicted_df$Predicted - actual_vs_predicted_df$Actual) ^ 2))

# Plot Actual vs Predicted with R-squared and RMSE
ggplot(actual_vs_predicted_df, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Actual vs Predicted Yield Difference",
       x = "Actual Yield Difference",
       y = "Predicted Yield Difference") +
  theme_minimal() +
  annotate("text", x = min(actual_vs_predicted_df$Actual),
           y = max(actual_vs_predicted_df$Predicted),
           label = paste("R-squared =", round(r_squared, 2)),
           hjust = 0, vjust = 1, size = 5, color = "darkred") +
  annotate("text", x = min(actual_vs_predicted_df$Actual),
           y = max(actual_vs_predicted_df$Predicted) - 0.1 * (max(actual_vs_predicted_df$Predicted) - mi
           label = paste("RMSE =", round(rmse, 2)),
           hjust = 0, vjust = 1, size = 5, color = "darkred")
```




### 3.6.1 Decision Support Tool

This decision support tool allows farmers to input key soil characteristics—pH, organic matter percentage, soil sulfate concentration, and soil texture—and receive a recommendation on whether to apply sulfur fertilizer. The tool uses the best predictive model to forecast the expected yield difference if sulfur is applied. If a positive yield increase is predicted, the tool advises applying the fertilizer and estimates the potential yield gain

```
# Function to make predictions based on user inputs
predict_sulfur_fertilizer <- function(pH, om_percent, soil_sulfate_mg_kg, texture_group) {

  # Ensure the input values are within the range of the training data
  pH <- min(max(pH, min(dataset_imputed$pH, na.rm = TRUE)), max(dataset_imputed$pH, na.rm = TRUE))
  om_percent <- min(max(om_percent, min(dataset_imputed$om_percent, na.rm = TRUE)), max(dataset_imputed$
  soil_sulfate_mg_kg <- min(max(soil_sulfate_mg_kg, min(dataset_imputed$soil_sulfate_mg_kg, na.rm = TRU

  # Convert texture_group to the appropriate factor levels
  texture_group <- factor(texture_group, levels = levels(dataset_imputed$texture_group))

  # Create a data frame with the inputs
  input_data <- data.frame(
    pH = pH,
    om_percent = om_percent,
    soil_sulfate_mg_kg = soil_sulfate_mg_kg,
    texture_group = texture_group
  )
```

```
  # Create dummy variables for texture_group and ensure consistent structure with training data
  input_data_engineered <- model.matrix(~ pH + om_percent + soil_sulfate_mg_kg + texture_group, data = i
  input_data_engineered <- as.data.frame(input_data_engineered)

  # Ensure that all necessary columns are present
  missing_cols <- setdiff(names(X_train), names(input_data_engineered))
  input_data_engineered[missing_cols] <- 0

  # Reorder columns to match the training data
  input_data_engineered <- input_data_engineered[, names(X_train)]

  # Make prediction using the best model
  if (inherits(best_model, "xgb.Booster")) {
    predicted_yield_diff <- predict(best_model, newdata = as.matrix(input_data_engineered))
  } else if (inherits(best_model, "randomForest")) {
    predicted_yield_diff <- predict(best_model, newdata = input_data_engineered)
  } else {
    stop("Model type not supported for predictions.")
  }

  # Decision rule based on predicted yield difference
  if (predicted_yield_diff > 0) {
    decision <- "Apply Sulfur Fertilizer"
    yield_increase <- predicted_yield_diff
    result <- paste(decision, "- Expected yield increase:", round(yield_increase, 2), "kg/ha")
  } else {
    decision <- "Don't Apply Sulfur Fertilizer"
    result <- decision
  }

  return(result)
}
```

**3.6.1.1 Example** In this example, the farmer inputs the following soil characteristics: pH of 6.5, organic matter percentage of 5%, soil sulfate concentration of 15 mg/kg, and a "Loam" soil texture. Based on these conditions, the decision support tool recommends "Don't Apply Sulfur Fertilizer."

```
# Example usage:
# Replace these values with the actual soil characteristics the farmer inputs
example_pH <- 6.5
example_om_percent <- 5
example_soil_sulfate_mg_kg <- 15
example_texture_group <- "Loam"

# Predict and print the decision
decision <- predict_sulfur_fertilizer(example_pH, example_om_percent, example_soil_sulfate_mg_kg, exampl
print(decision)
```

```
## [1] "Don't Apply Sulfur Fertilizer"
```

# 4 Conclusion

This study demonstrates the potential of machine learning techniques in predicting the effectiveness of sulfur fertilization based on soil characteristics in wheat. The XGBoost model showed the best performance,

capturing complex interactions between soil properties and their impact on wheat yield response to sulfur application.

However, the model's precision is limited, suggesting the need for a larger dataset and additional parameters like weather data.

The decision support tool developed here could optimize sulfur fertilizer use, improving yields by offering data-driven recommendations. However, several limitations exist:

- The dataset may not represent all agricultural regions.
- Temporal variations in soil conditions and climate factors were not considered.
- Economic aspects of sulfur fertilization were not analyzed.

Future research should focus on:

- Incorporating additional variables such as climate data and different crops.
- Developing region-specific models to account for local conditions.
- Integrating economic factors for cost-effective fertilization recommendations.
- Validating model predictions through field trials in diverse agricultural settings.

**Disclosure**

This document was developed with the assistance of AI tools, including ChatGPT, which was used to help improve writing clarity, structure, and to assist with some coding tasks. Additionally, grammar-checking tools were employed to refine the text. The final content reflects the author's expertise and the use of AI to enhance productivity and accuracy.

# References

Athey, Susan, Julie Tibshirani, and Stefan Wager. 2019. "Generalized Random Forests." *The Annals of Statistics* 47 (2): 1148–78. https://doi.org/10.1214/18-AOS1709.

Roa, Gustavo. n.d. "Dataset for Meta-Analysis on Sulfur Fertilization Effect on Wheat Protein and Yield." https://doi.org/10.7910/DVN/4RLPP1.

Scherer, Heinrich Wilhelm. 2009. "Sulfur in Soils." *Journal of Plant Nutrition and Soil Science* 172 (3): 326–35. https://doi.org/10.1002/jpln.200900037.

Zhao, FJ, MJ Hawkesford, and SP McGrath. 1999. "Sulphur Assimilation and Effects on Yield and Quality of Wheat." *Journal of Cereal Science* 30 (1): 1–17. https://doi.org/10.1006/jcrs.1998.0241.