Wage Prediction Analysis

Nikhil Prema Chandra Rao

2024-12-11

Introduction

This project performs wage prediction analysis using various machine learning models on a wage dataset. The goal is to compare the performance of multiple models including Linear Regression, Random Forest, and XGBoost. Hyperparameter tuning and regularization techniques like Lasso and Ridge Regression are also explored.

In this analysis, we will explore a dataset containing wage information. The goal is to understand the relationship between various factors, such as age, education, and year, and the wage of individuals. By analyzing this data, we aim to build a predictive model that can estimate an individual's wage based on these factors.

About the Data

The data set, `Wage.csv`, consists of information on individuals' wages along with other attributes that might influence their income. The key variables in this data set include:

- **age**: The age of the individual.
- **education**: The highest level of education attained by the individual (e.g., High School, Bachelor's Degree, Master's Degree).
- **year**: The year when the wage was recorded.
- **wage**: The wage of the individual, which is our target variable.

This data set allows us to explore the impact of age, education, and the year on an individual's wage. We will perform exploratory data analysis (EDA) to understand the relationships between these variables, followed by fitting a linear regression model to predict wages based on these features.

Objective of the Analysis

The main objectives of this analysis are as follows:

1. **Data Pre-processing**: Load and clean the data set to ensure that it is ready for analysis.
2. **Exploratory Data Analysis (EDA)**: Visualize and summarize the data to uncover patterns and trends.
3. **Modeling**: Fit a linear regression model to predict wages based on age, education, and year.
4. **Model Evaluation**: Assess the performance of the regression model and interpret the results.

By the end of this analysis, we aim to understand how different factors contribute to wage variations and use this knowledge to make predictions about an individual's wage.

load_libraries

This function loads a list of necessary tools (called libraries) that help perform various tasks like reading data, creating plots, and building machine learning models.

```
load_libraries <- function() {
  libraries <- c(
    "caret", "car", "readr", "dplyr", "ggplot2", "GGally", "gridExtra",
    "grid", "glmnet", "Metrics", "rpart", "rpart.plot", "pROC", "tidyr", "reshape2",
    "randomForest", "DiagrammeR", "xgboost"
  )
  lapply(libraries, function(lib) {
    suppressMessages(suppressWarnings(require(lib, character.only = TRUE)))
  })
}
```

load_and_prepare_data

This function reads a file containing wage data and prepares it by converting text into categories and cleaning up unnecessary columns. It ensures that the data is ready for analysis.

```
load_and_prepare_data <- function(file_path) {
  wage_data <- read_csv(file_path)
  wage_data <- wage_data %>%
    mutate(across(where(is.character), as.factor)) %>%
    select(where(~ !is.factor(.) || length(levels(.)) > 1))
  return(wage_data)
}
```

plot_data_visualizations

This function creates several visual charts to help understand the data better. These include a histogram of wages, a boxplot to show how wages vary by education level, a scatterplot of wage versus age, and a correlation plot showing relationships between different variables.

```
# Generate and plot visualizations
plot_data_visualizations <- function(data) {
  # Wage Histogram
  wage_histogram <- ggplot(data, aes(x = wage)) +
    geom_histogram(bins = 30, fill = "skyblue", color = "black", alpha = 0.7) +
    ggtitle("Distribution of Wage") +
    xlab("Wage") +
    ylab("Frequency") +
    theme_minimal(base_size = 12)

  # Wage by Education Boxplot
  wage_education_boxplot <- ggplot(data, aes(x = education, y = wage, fill = education)) +
    geom_boxplot(outlier.color = "red", outlier.shape = 1, notch = TRUE) +
    ggtitle("Wage Distribution by Education Level") +
    xlab("Education Level") +
    ylab("Wage") +
    theme_minimal(base_size = 12) +
```

```
    theme(axis.text.x = element_text(angle = 45, hjust = 1), legend.position = "none")

  # Wage vs Age Scatterplot
  wage_age_scatterplot <- ggplot(data, aes(x = age, y = wage)) +
    geom_point(alpha = 0.6, color = "darkblue") +
    geom_smooth(method = "lm", color = "red", se = FALSE, linetype = "dashed") +
    ggtitle("Wage vs Age") +
    xlab("Age") +
    ylab("Wage") +
    theme_minimal(base_size = 12)

  # Correlation Plot
  correlation_plot <- ggpairs(data[, c("year", "age", "logwage", "wage")],
                              lower = list(continuous = wrap("smooth", color = "blue", alpha = 0.3)),
                              diag = list(continuous = wrap("densityDiag", fill = "blue", alpha = 0.3)),
                              upper = list(continuous = wrap("cor", size = 4))) +
    ggtitle("Correlation Plot")

  # Convert ggmatrix to a grob
  correlation_grob <- GGally::ggmatrix_gtable(correlation_plot)

  # Improved Layout
  grid.arrange(
    ggplotGrob(wage_histogram),               # Top-left: Wage Histogram
    ggplotGrob(wage_education_boxplot),       # Top-right: Wage by Education
    ggplotGrob(wage_age_scatterplot),         # Bottom-left: Wage vs Age
    correlation_grob,                         # Bottom-right: Correlation Plot
    layout_matrix = rbind(
      c(1, 2),  # Wage Histogram and Education Boxplot side-by-side
      c(3, 4)   # Wage vs Age and Correlation Plot side-by-side
    ),
    top = textGrob("Data Visualizations", gp = gpar(fontsize = 18, fontface = "bold"))
  )
}
```

add_data_visualizations

This function creates additional charts like scatter plots, density plots, and bar charts to visualize the data in different ways, making it easier to spot trends.

```
add_data_visualizations <- function(data) {
  scatter_age_education <- ggplot(data, aes(x = age, y = wage, color = education)) +
    geom_point(alpha = 0.5) +
    labs(title = "Wage vs Age by Education", x = "Age", y = "Wage")

  wage_density <- ggplot(data, aes(x = wage, fill = education)) +
    geom_density(alpha = 0.5) +
    labs(title = "Density Plot of Wage by Education", x = "Wage", y = "Density")

  education_barplot <- ggplot(data, aes(x = education, fill = education)) +
    geom_bar() +
    labs(title = "Education Level Distribution", x = "Education Level", y = "Count")
```

```
  grid.arrange(scatter_age_education, wage_density, education_barplot, nrow = 2)
}
```

fit_multilinear_regression

This function creates a mathematical model (called a regression) that predicts wages based on other factors. It shows how different factors affect the wage and gives insights into the importance of each.

```
# Function for Multilinear Regression
fit_multilinear_regression <- function(data) {
  model <- lm(wage ~ ., data = data)
  print(summary(model))
  par(mfrow = c(2, 2))
  plot(model)
  return(model)
}
```

fit_random_forest

This function builds a machine learning model called Random Forest, which can predict wages by considering many factors. It also shows which factors are the most important for predictions.

```
# Function to implement Random Forests
fit_random_forest <- function(data, formula) {
  rf_model <- randomForest(formula, data = data, importance = TRUE)
  print(rf_model)
  varImpPlot(rf_model, main = "Variable Importance (Random Forest)")
  plot(rf_model)
  return(rf_model)
}
```

fit_xgboost

This function uses another machine learning method, XGBoost, to predict wages. It's known for its efficiency and accuracy in handling complex data.

```
# Function to implement XGBoost
fit_xgboost <- function(data, formula) {
  x_data <- model.matrix(formula, data)[, -1]  # Remove intercept
  y_data <- data[[as.character(formula[[2]])]]  # Extract response variable

  xgb_model <- xgboost(
    data = as.matrix(x_data), label = y_data, max.depth = 3,
    eta = 0.1, nrounds = 100, objective = "reg:squarederror", verbose = 0
  )
  #plot(xgb_model)
  print(xgb_model)
  return(xgb_model)
}
```

fit_ridge_regression

This function builds a model similar to linear regression but with a twist. It reduces the impact of less important factors to make the model more stable. It also selects the best settings for the model to improve predictions.

```r
# Function for Ridge Regression
fit_ridge_regression <- function(data) {
  # Prepare the model matrix (predictors) and response vector
  x <- model.matrix(wage ~ . - 1, data = data)  # Remove the intercept for glmnet
  y <- data$wage

  # Fit ridge regression model with a sequence of lambda values
  ridge_model <- glmnet(x, y, alpha = 0)  # alpha = 0 for ridge regression

  # Plot the coefficient path for ridge regression
  plot(ridge_model, xvar = "lambda", label = TRUE)
  title("Coefficient Path for Ridge Regression")

  # Perform cross-validation to find the optimal lambda
  cv_ridge <- cv.glmnet(x, y, alpha = 0)
  plot(cv_ridge)
  title("Cross-Validation for Selecting Lambda")

  # Extract the best lambda value (minimizing cross-validated mean squared error)
  best_lambda <- cv_ridge$lambda.min
  cat("Best lambda: ", best_lambda, "\n")

  # Return the coefficients at the best lambda
  return(coef(ridge_model, s = best_lambda))
}
```

perform_cross_validation

This function tests a model's ability to predict well by checking it on different parts of the data. It helps choose the best settings for the model to perform better in the real world.

```r
# Function to compute cross-validation for Lasso and Ridge
perform_cross_validation <- function(data, alpha) {
  x <- model.matrix(wage ~ . - 1, data = data)
  y <- data$wage
  cv_model <- cv.glmnet(x, y, alpha = alpha)
  plot(cv_model)
  title(ifelse(alpha == 1, "Lasso Cross-Validation", "Ridge Cross-Validation"))
  best_lambda <- cv_model$lambda.min
  cat("Best Lambda: ", best_lambda, "\n")
  return(best_lambda)
}
```

compare_model_metrics

This function compares the performance of different models by calculating errors in their predictions, helping us see which model is the most accurate.

```r
# Function to compare model metrics
compare_model_metrics <- function(actuals, predictions_list, model_names) {
  metrics <- data.frame(
    Model = model_names,
    RMSE = sapply(predictions_list, function(pred) rmse(actuals, pred)),
    MAE = sapply(predictions_list, function(pred) mae(actuals, pred))
  )
  print(metrics)
}
```

generate_summary_report

This function generates a summary report of the model's performance and saves it to a file. It provides an easy way to review how well the models did in the analysis.

```r
# Function to generate summary report
generate_summary_report <- function(metrics, filename = "model_summary_report.txt") {
  sink(filename)
  cat("Summary Report of Model Performance\n")
  cat("-----------------------------------\n")
  print(metrics)
  sink()
  cat("Summary report saved to ", filename, "\n")
}
```

fit_and_plot_regression_tree

This function creates a regression tree, which is a type of model that predicts wages by splitting the data based on different features. It visualizes the decision-making process of the tree, showing how wages vary across different groups of data.

```r
# Function to create and visualize a Regression Tree
fit_and_plot_regression_tree <- function(data, formula) {
  # Fit the regression tree model
  tree_model <- rpart(formula, data = data, method = "anova")

  # Plot the regression tree showing the mean and percentage of observations
  rpart.plot(tree_model, main = "Regression Tree for Wage Prediction",
             extra = 101,  # Display the mean of the dependent variable and percentage of observations
             under = TRUE, # Place node numbers underneath the node labels
             faclen = 0)   # Don't abbreviate factor levels

  return(tree_model)
}
```

Main Script

Data Loading and Preparation

```r
file_path <- "./Wage.csv"
wage_data <- load_and_prepare_data(file_path)
```

```
## Rows: 3000 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): maritl, race, education, region, jobclass, health, health_ins
## dbl (4): year, age, logwage, wage
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
str(wage_data)
```

```
## tibble [3,000 x 10] (S3: tbl_df/tbl/data.frame)
##  $ year      : num [1:3000] 2006 2004 2003 2003 2005 ...
##  $ age       : num [1:3000] 18 24 45 43 50 54 44 30 41 52 ...
##  $ maritl    : Factor w/ 5 levels "1. Never Married",..: 1 1 2 2 4 2 2 1 1 2 ...
##  $ race      : Factor w/ 4 levels "1. White","2. Black",..: 1 1 1 3 1 1 4 3 2 1 ...
##  $ education : Factor w/ 5 levels "1. < HS Grad",..: 1 4 3 4 2 4 3 3 3 2 ...
##  $ jobclass  : Factor w/ 2 levels "1. Industrial",..: 1 2 1 2 2 2 1 2 2 2 ...
##  $ health    : Factor w/ 2 levels "1. <=Good","2. >=Very Good": 1 2 1 2 1 2 2 1 2 2 ...
##  $ health_ins: Factor w/ 2 levels "1. Yes","2. No": 2 2 1 1 1 1 1 1 1 1 ...
##  $ logwage   : num [1:3000] 4.32 4.26 4.88 5.04 4.32 ...
##  $ wage      : num [1:3000] 75 70.5 131 154.7 75 ...
```

Dataset Overview and Column Types

The dataset used in this analysis is the "Wage.csv" file, which contains 3000 data points (rows) and 11 variables (columns). The following summarizes the columns and their types:

- **file_path**: Specifies the location of the CSV file (`Wage.csv`).
- **wage_data**: This variable stores the dataset after loading it through the `load_and_prepare_data()` function. The data is loaded and prepared, ensuring that it is ready for analysis.

Dataset Dimensions:

- **Rows: 3000**: The dataset contains 3000 rows, each representing an individual record or data point. This means there are 3000 observations available for analysis.
- **Columns: 11**: There are 11 columns, each representing different attributes related to the wage data. These columns include both categorical and numerical variables.

Column Types:

- **chr (7)**: Seven columns are of character type, representing categorical data. These include:
  - **maritl**: Marital status of the individual (e.g., married, never married).
  - **race**: The race of the individual (e.g., white, black, Asian).
  - **education**: Education level of the individual (e.g., high school, college graduate).
  - **region**: Geographical region where the individual resides.

  - **jobclass**: Type of job (e.g., industrial, information).
  - **health**: Health status (e.g., good, very good).
  - **health_ins**: Whether the individual has health insurance (e.g., yes, no).

- **dbl (4)**: Four columns are numeric, representing continuous variables. These include:
  - **year**: The year of the wage record.
  - **age**: The age of the individual.
  - **logwage**: The logarithm of the individual's wage, used to reduce skewness in the wage distribution for modeling.
  - **wage**: The actual wage of the individual.

Summary of the data

```
summary(wage_data)
```

```
##       year           age                     maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##                education             jobclass               health
##  1. < HS Grad      :268   1. Industrial :1544   1. <=Good     : 858
##  2. HS Grad        :971   2. Information:1456   2. >=Very Good:2142
##  3. Some College   :650
##  4. College Grad   :685
##  5. Advanced Degree:426
##
##   health_ins      logwage           wage
##  1. Yes:2083   Min.   :3.000   Min.   : 20.09
##  2. No : 917   1st Qu.:4.447   1st Qu.: 85.38
##                Median :4.653   Median :104.92
##                Mean   :4.654   Mean   :111.70
##                3rd Qu.:4.857   3rd Qu.:128.68
##                Max.   :5.763   Max.   :318.34
```

The dataset is stored as a **tibble** (a modern version of a data frame in R), with **3000 rows** and **10 columns**. The tibble provides a more user-friendly output than a traditional data frame, especially when working with large datasets, making it easier to navigate and explore the data.

Data Types:

- **Numeric columns**:
  - `year`: The year associated with each record, which is a continuous numeric variable.
  - `age`: The age of the individual, a continuous numeric variable.
  - `logwage`: The logarithmic transformation of wages, which is a numeric variable that helps normalize the data, especially when the wage distribution is skewed.
  - `wage`: The raw wage of the individual in dollars, a numeric variable.

- **Factor columns**: The remaining columns represent categorical data, stored as **factors** in R. Factors are ideal for representing categorical variables because they store the unique levels of a variable.
  - `maritl`: Marital status, with five levels: "Never Married", "Married", "Widowed", "Divorced", and "Separated".
  - `race`: The racial background of the individual, with four levels: "White", "Black", "Asian", and "Other".
  - `education`: The educational level of the individual, which includes five categories: "< HS Grad", "HS Grad", "Some College", "College Grad", and "Advanced Degree".
  - `jobclass`: The type of job the individual holds, either "Industrial" or "Information".
  - `health`: The self-reported health status of the individual, either "Good" or "Very Good".
  - `health_ins`: Whether the individual has health insurance, with two levels: "Yes" or "No".

Numeric Columns:

- `year`: The dataset spans years from **2003 to 2009**, providing data on wages across a range of time periods.
- `age`: The age of individuals ranges from **18 to 80**, capturing a wide age group and allowing for the analysis of how age impacts wages.
- `logwage` and `wage`: The statistics for the **log-transformed wages** (`logwage`) and the **raw wages** (`wage`) give a deeper understanding of the wage distribution. The transformation helps normalize highly skewed data, making it more suitable for certain statistical models.

Categorical Columns: For the categorical columns, the summary function returns the **count** of each category. This helps identify the distribution of each categorical variable: - `maritl`: The distribution of marital statuses shows how many individuals fall into each category (e.g., "Never Married", "Married"). - `race`: The racial background of individuals is categorized and summarized with counts. - `education`: The dataset includes a breakdown of educational levels, helping analyze the relationship between education and wage. - `jobclass`, `health`, and `health_ins`: The counts of job class, health status, and health insurance provide insight into how these factors are distributed among the dataset.

The summary of the data structure provides insights into the composition of the dataset, which is useful for selecting appropriate statistical techniques and models.

Check for missing values and Remove rows with missing values

```
sum(is.na(wage_data))
```

```
## [1] 0
```

```
wage_data <- na.omit(wage_data)
```

To ensure the dataset is clean and ready for analysis, we check for any missing values using the `is.na()` function.

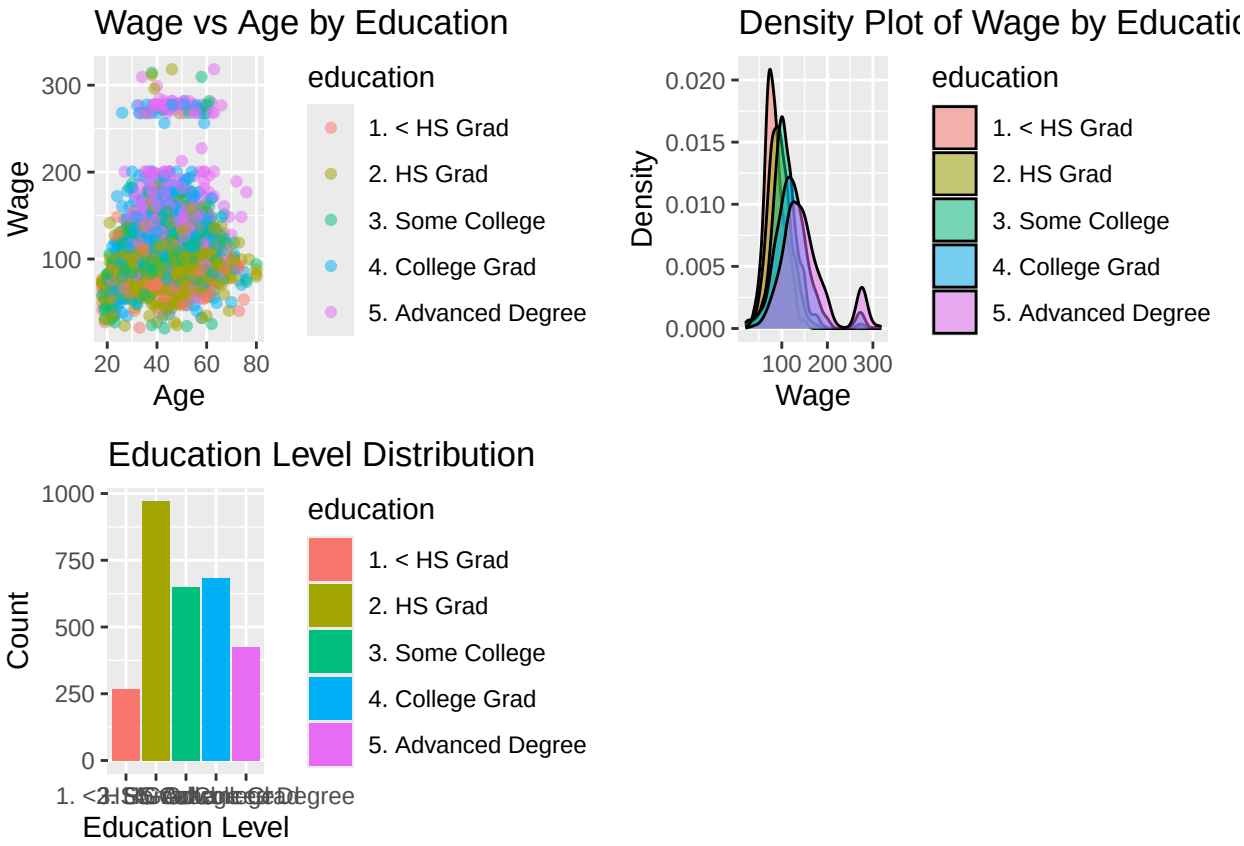
Result:

- `0`: The result indicates that there are **no missing values** in the dataset, meaning that all data points are complete. This is essential for accurate analysis and modeling, as missing values can lead to biased or inaccurate results if not handled properly.

This step confirms the integrity of the data and ensures that no further cleaning steps are required for missing values.

Visualizations

```
add_data_visualizations(wage_data)
```



Explanation of the above plots

Top Left: Wage vs. Age by Education

- **Description**: A scatter plot depicting the relationship between wage and age, categorized by different education levels.
- **Observations**:
  - Each point represents an individual's age and wage.
  - Points are color-coded based on education level: "< HS Grad," "HS Grad," "Some College," "College Grad," and "Advanced Degree."
  - The spread of wages tends to increase with age, with a greater variance at older ages.
  - Higher education levels (e.g., "Advanced Degree") are associated with higher wages.

Top Right: Density Plot of Wage by Education

- **Description**: A kernel density plot showing the distribution of wages for different education levels.
- **Observations**:
  - Each line represents the wage distribution for a specific education level.

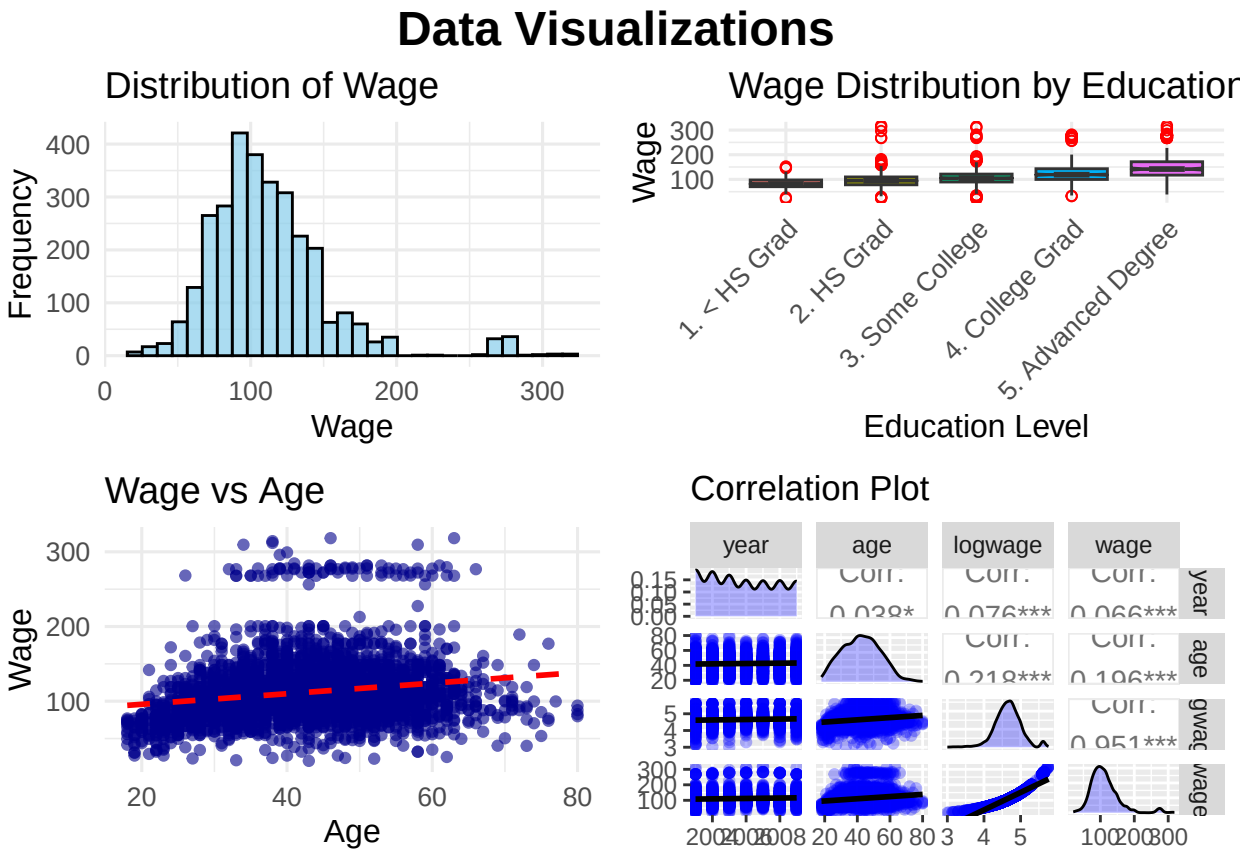
- Individuals with higher education levels (e.g., "College Grad" and "Advanced Degree") have distributions shifted towards higher wages.
- The peak density for "HS Grad" and "< HS Grad" occurs at lower wage ranges, indicating that these groups are more likely to earn lower wages.

Bottom: Education Level Distribution

- **Description**: A bar chart displaying the count of individuals in each education level.
- **Observations**:
  - "HS Grad" has the highest count, indicating most individuals in the dataset belong to this category.
  - Fewer individuals have "< HS Grad" and "Advanced Degree" education levels.

```
plot_data_visualizations(wage_data)
```

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



Explanation of the Second Image

Top Left: Distribution of Wage

- **Description**: A histogram depicting the frequency distribution of wages.
- **Observations**:

  - Wages are concentrated around $100, with fewer individuals earning either very low or very high wages.
  - The distribution is slightly right-skewed, with a long tail extending towards higher wages.

Top Right: Wage Distribution by Education

- **Description**: A boxplot showing the distribution of wages for each education level.
- **Observations**:
  - The median wage increases with higher education levels.
  - "Advanced Degree" shows the highest median wage, with a wider interquartile range, indicating variability in wages.
  - "< HS Grad" has the lowest median wage and relatively fewer outliers.

Bottom Left: Wage vs. Age

- **Description**: A scatter plot of wage versus age with a fitted trend line.
- **Observations**:
  - A positive correlation is observed, indicating that wages tend to increase with age up to a certain point.
  - There is significant variation in wages for older individuals, with some earning exceptionally high wages.

Bottom Right: Correlation Plot

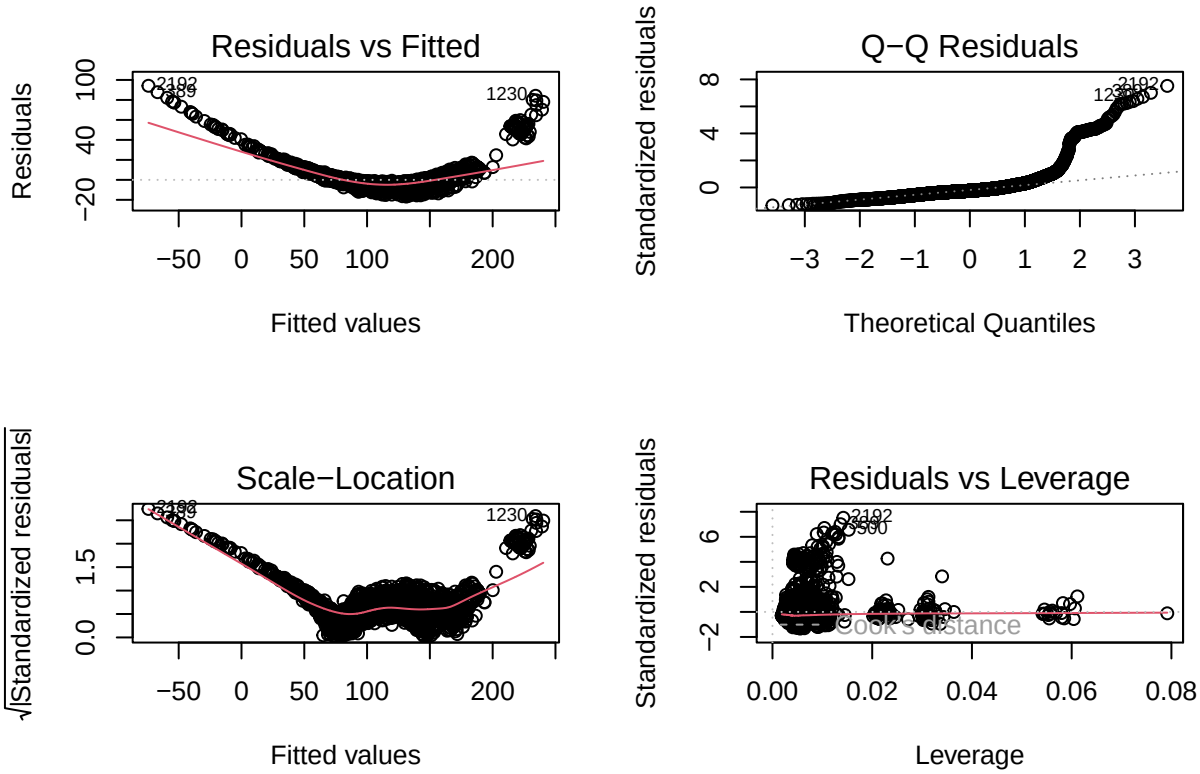
- **Description**: A correlation matrix with scatterplots and density curves for numerical variables like "age," "wage," and "logwage."
- **Observations**:
  - **Diagonal Elements**: Density plots for individual variables.
  - **Off-diagonal Elements**: Scatterplots showing pairwise relationships.
  - Strong positive correlation is observed between "wage" and "logwage."
  - Weak positive correlations exist between "age" and "wage" or "logwage."

Models:

```
fit_multilinear_regression(wage_data)
```

```
##
## Call:
## lm(formula = wage ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.721  -5.359  -3.058   0.712  94.152
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -42.18179  229.31872  -0.184   0.8541
## year                        -0.18561    0.11449  -1.621   0.1051
```

```
## age                           -0.01716    0.02317  -0.740   0.4592
## maritl2. Married              -1.46198    0.65235  -2.241   0.0251 *
## maritl3. Widowed              -3.49564    2.96917  -1.177   0.2392
## maritl4. Divorced             -1.27917    1.07126  -1.194   0.2325
## maritl5. Separated            -2.50237    1.79938  -1.391   0.1644
## race2. Black                  -0.53547    0.79644  -0.672   0.5014
## race3. Asian                  -0.41331    0.96561  -0.428   0.6687
## race4. Other                   0.68373    2.10209   0.325   0.7450
## education2. HS Grad           -1.53915    0.88136  -1.746   0.0809 .
## education3. Some College      -2.38319    0.94682  -2.517   0.0119 *
## education4. College Grad      -0.77893    0.97362  -0.800   0.4238
## education5. Advanced Degree    4.86794    1.10266   4.415 1.05e-05 ***
## jobclass2. Information         0.64614    0.49152   1.315   0.1887
## health2. >=Very Good          -0.18688    0.52922  -0.353   0.7240
## health_ins2. No                4.41498    0.54460   8.107 7.51e-16 ***
## logwage                      113.26508    0.82815 136.769  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.61 on 2982 degrees of freedom
## Multiple R-squared:  0.9092, Adjusted R-squared:  0.9087
## F-statistic:  1756 on 17 and 2982 DF,  p-value: < 2.2e-16
```




```
##
## Call:
```

```
## lm(formula = wage ~ ., data = data)
##
## Coefficients:
##                  (Intercept)                          year
##                    -42.18179                       -0.18561
##                          age              maritl2. Married
##                      -0.01716                       -1.46198
##             maritl3. Widowed             maritl4. Divorced
##                      -3.49564                       -1.27917
##           maritl5. Separated                  race2. Black
##                      -2.50237                       -0.53547
##                 race3. Asian                  race4. Other
##                      -0.41331                        0.68373
##          education2. HS Grad     education3. Some College
##                      -1.53915                       -2.38319
##     education4. College Grad  education5. Advanced Degree
##                      -0.77893                        4.86794
##       jobclass2. Information         health2. >=Very Good
##                       0.64614                       -0.18688
##              health_ins2. No                        logwage
##                       4.41498                      113.26508
```

Output:

Residuals

The residuals represent the differences between the observed and predicted values of the dependent variable (`wage`).

Summary of Residuals:

- **Minimum**: -16.721
- **1st Quartile (25%)**: -5.359
- **Median (50%)**: -3.058
- **3rd Quartile (75%)**: 0.712
- **Maximum**: 94.152 This indicates that while most residuals are relatively small, there is a large positive residual (94.152) suggesting an underestimation for at least one data point.

Coefficients

The table below describes the relationship between predictors and the dependent variable (`wage`): - **Estimate**: The coefficient value represents the effect of the predictor on the dependent variable. - **Std. Error**: The standard error of the estimate. - **t value**: The test statistic (Estimate / Std. Error). - **Pr(>|t|)**: The p-value determines the statistical significance of the predictor.

Key Coefficients:

Predictor	Estimate	Std. Error	t value	Pr(>	t
(Intercept)	-42.18179	229.31872	-0.184	0.8541	Not Significant
year	-0.18561	0.11449	-1.621	0.1051	Not Significant
maritl2. Married	-1.46198	0.65235	-2.241	0.0251	* Significant
education3. Some College	-2.38319	0.94682	-2.517	0.0119	* Significant
education5. Advanced Degree	4.86794	1.10266	4.415	1.05e-05	*** Highly Significant
health_ins2. No	4.41498	0.54460	8.107	7.51e-16	*** Highly Significant
log wage	113.26508	0.82815	136.769	< 2e-16	*** Highly Significant

Observations:

1. **logwage** has the highest positive impact on `wage` with an extremely high t-value and a p-value < 2e-16.
2. Individuals with an **Advanced Degree** (education5) earn significantly higher wages.
3. Those without **health insurance** (health_ins2. No) have higher predicted wages.
4. The **marital status (maritl2. Married)** shows a small but statistically significant negative impact on `wage`.

Model Summary

- **Residual Standard Error**: 12.61
  This indicates the average deviation of the actual wages from the predicted values is approximately 12.61 units.
- **R-squared**: 0.9092
  The model explains approximately 90.92% of the variance in `wage`.
- **Adjusted R-squared**: 0.9087
  Adjusted R-squared accounts for the number of predictors, still explaining 90.87% of the variance.
- **F-statistic**: 1756
  The F-statistic indicates the overall significance of the model, with a p-value < 2.2e-16, confirming the model is statistically significant.

The multilinear regression model provides valuable insights into the factors influencing `wage`. The key takeaways include: - **Education** and **logwage** are strong predictors of wage. - Certain categorical variables, such as marital status and health insurance, also play significant roles. - The model is robust, with high R-squared values and statistically significant predictors.

Graph expalnation

Residuals vs. Fitted **Purpose**: This plot evaluates whether the residuals (errors) exhibit any systematic pattern that would violate the assumptions of linearity or homoscedasticity (constant variance).

Interpretation: - The residuals are plotted on the y-axis, and the fitted values are plotted on the x-axis. - Ideally, the points should randomly scatter around the horizontal line (red line), suggesting no systematic relationship between residuals and fitted values.

Observations: - A clear **curve** in this plot indicates non-linearity, suggesting that the linear model may not adequately capture the relationship between predictors and the response variable. - The extreme residuals (e.g., points 2192, 3589, and 1230) may indicate potential outliers or influential points.

Normal Q-Q Plot

Purpose: This plot checks whether the residuals are normally distributed, an assumption of linear regression. **Interpretation**: - Standardized residuals are plotted against theoretical quantiles of a standard normal distribution. - Points should ideally lie on the diagonal dashed line.

Observations: - The significant deviation at the upper and lower ends of the plot (heavy tails) suggests that the residuals are **not perfectly normally distributed**. - Points such as 2192, 3589, and 1230 deviate significantly from the line, indicating potential outliers.

Scale-Location Plot (Spread-Location Plot)

Purpose: This plot assesses the homoscedasticity assumption (constant variance of residuals). **Interpretation**: - The y-axis represents the square root of the absolute standardized residuals, and the x-axis shows the fitted values. - The red line shows the trend of residual spread. Ideally, the red line should be flat, indicating constant variance.

Observations: - The distinct pattern (a curve) suggests **heteroscedasticity** (non-constant variance of residuals). - Variance increases for larger fitted values, as seen in the spread of points, indicating potential issues with the model.

Residuals vs. Leverage

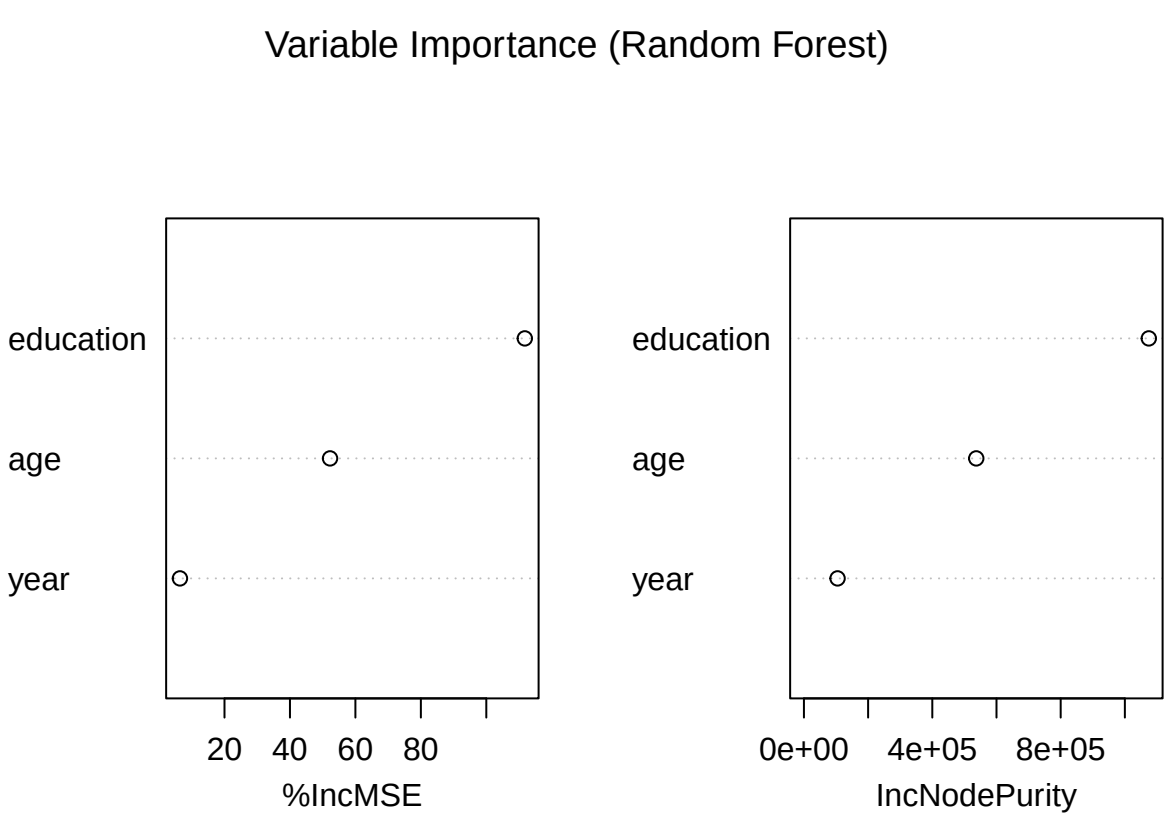
Purpose: This plot identifies influential data points that might disproportionately affect the regression model's coefficients. **Interpretation**: - Leverage (x-axis) measures how far a data point's predictors are from the average of all predictor values. - Standardized residuals (y-axis) measure the deviation of the predicted value from the observed value, scaled by the residual standard error. - Cook's distance (dotted lines) helps identify influential points. Points outside the Cook's distance lines are highly influential.

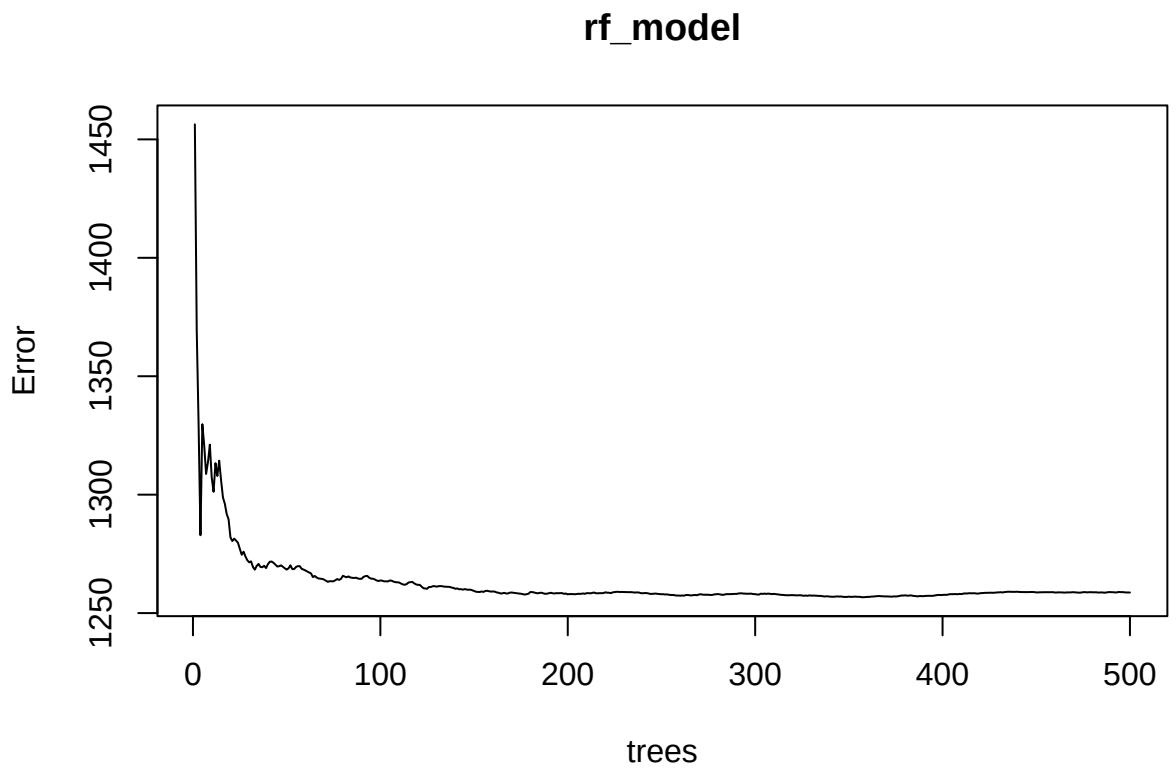
Observations: - Points like 2192 and 3589 have high leverage and large standardized residuals, indicating they are potentially **influential points**. - These points could have a significant impact on the model's fit and may need further investigation or removal.

Random Forest Model:

```
rf_model <- fit_random_forest(wage_data, wage ~ age + education + year)
```

```
##
## Call:
##  randomForest(formula = formula, data = data, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 1258.696
##                     % Var explained: 27.69
```


Variable Importance (Random Forest)
education
age
year
20
40
60
80
%IncMSE
education
age
year
0e+00
4e+05
8e+05
IncNodePurity



Output:

Random Forest Model Output Explanation

Random Forest Function

- **randomForest**: This is the function used to fit a random forest model.
- **formula**: This specifies the model formula (the relationship between the dependent and independent variables).
- **data**: The dataset used to train the model.
- **importance = TRUE**: This argument ensures that the importance of each feature (predictor variable) is calculated and displayed in the output.

Type of Random Forest: Regression

- The model is being used for **regression**. This means that the response variable is continuous, and the goal is to predict a numerical value, as opposed to classification where the goal is to predict a categorical outcome.

Number of Trees: 500

- The random forest consists of **500 trees**. This is the number of individual decision trees that are built as part of the ensemble model. More trees generally lead to more stable and accurate predictions, though it increases computation time.

No. of Variables Tried at Each Split: 1

- At each node of a decision tree, the algorithm tries to split based on **1 randomly chosen variable** out of the available features (predictors). This randomness helps ensure diversity among the trees, improving the performance of the ensemble.

Mean of Squared Residuals: 1257.707

- This is the **mean squared residual (MSR)**, which measures the average squared difference between the observed actual outcomes and the predicted outcomes from the model. A lower value indicates better model performance (i.e., less error).

  Specifically, **MSR** is calculated as:

$$MSR = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  where y_i are the actual values, $\hat{y}_i$ are the predicted values, and n is the number of data points.

% Var Explained: 27.75

- This represents the percentage of **variance explained** by the random forest model. It's a measure of how well the model's predictions fit the actual data. A value of **27.75%** means that the model explains about 27.75% of the variability in the dependent variable. This is relatively modest, and further model tuning, feature engineering, or data transformation may be needed to improve performance.

Explanation of Graph 1: Variable Importance

The first graph depicts the importance of variables in a Random Forest model using two measures: %IncMSE (percent increase in mean squared error) and IncNodePurity (increase in node purity). The left panel shows that "education" is the most critical predictor since permuting it results in the largest increase in prediction error. "Age" is moderately important, while "year" is the least influential. Similarly, the right panel confirms that "education" contributes the most to reducing node impurity in the decision trees, followed by "age" and "year."

Explanation of Graph 2: Error vs. Number of Trees

The second graph illustrates how the prediction error decreases as the number of trees in the Random Forest grows. Initially, the error declines rapidly, showing significant performance improvement when fewer than 50 trees are included. Beyond 200 trees, the error plateaus, indicating that adding more trees provides minimal additional benefit. This suggests the model achieves optimal performance with about 200 trees, as it stabilizes and converges to its lowest error rate.

Summary:

- The output indicates that the random forest model was built with 500 trees, using 1 variable at each split, and was fitted to a regression problem.
- The model explains approximately 27.75% of the variance in the target variable, and the average squared residuals are 1257.707, indicating the amount of error in the model's predictions.

XG_Boost Model:

```
xgb_model <- fit_xgboost(wage_data, wage ~ age + education + year)
```

```
## ##### xgb.Booster
## raw: 116.7 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, max.depth = 3, eta = 0.1, objective = "reg:squarederror")
## params (as set within xgb.train):
##   max_depth = "3", eta = "0.1", objective = "reg:squarederror", validate_parameters = "TRUE"
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 6
## niter: 100
## nfeatures : 6
## evaluation_log:
##      iter train_rmse
##     <num>      <num>
##         1  108.05817
##         2   98.52298
## ---
##        99   34.09750
##       100   34.08860
```

Output:

1. Model Configuration and Parameters

Call: The model was trained using the `xgb.train` function with the following parameters: - **max_depth = 3**: Restricts the maximum depth of each tree to control overfitting. - **eta = 0.1**: Sets the learning rate, balancing between the speed of convergence and model accuracy. - **objective = "reg:squarederror"**: Optimizes for regression tasks by minimizing the squared error.

Training Data: The model was trained using a dataset with features such as `age`, `education`, and `year`.

2. Model Attributes

- **Raw Size**: The model occupies **116.7 Kb** in memory.
- **Number of Features**: Six features were utilized for training the model.
- **Number of Iterations (niter)**: The model was trained over **100 iterations**, allowing it to gradually optimize the predictive error.

3. Evaluation Log

The evaluation log tracks the Root Mean Square Error (RMSE) during each training iteration:

Iteration	Training RMSE
1	108.06
2	98.52
…	…
99	34.10
100	34.09

Key Insights: - **Trend**: The RMSE decreases steadily with each iteration, indicating that the model is learning effectively and improving predictions. - **Final RMSE**: At the 100th iteration, the RMSE reaches **34.09**, suggesting that the model's predictions deviate, on average, by this value from the actual data.

4. Callbacks

- The model used a callback function, `cb.evaluation.log()`, to record evaluation metrics during training. This is helpful for monitoring the model's progress.

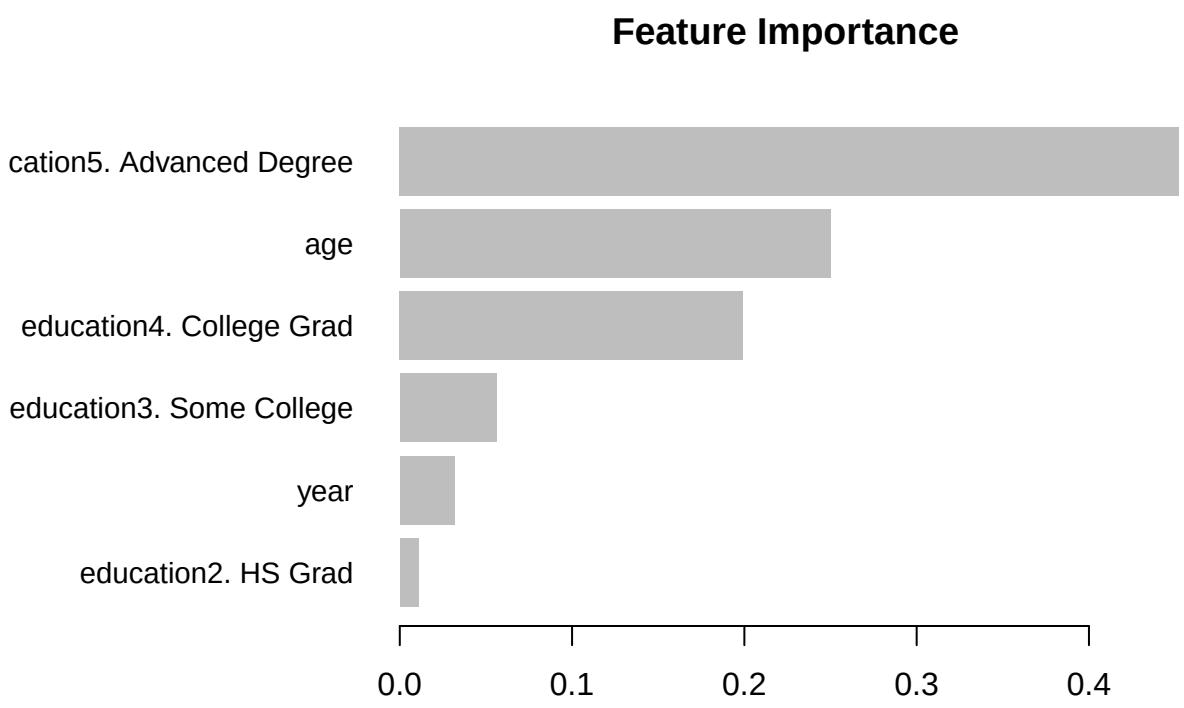
Interpretation and Key Observations

1. **Performance**:
   - The steady decline in RMSE suggests the model is performing well during training. However, this performance needs validation on unseen data.
2. **Model Complexity**:
   - The parameter `max_depth = 3` ensures the trees remain shallow, reducing overfitting while capturing key relationships.
3. **Learning Rate**:
   - A small `eta = 0.1` ensures the model optimizes weights cautiously, improving stability during training.
4. **Potential Concerns**:
   - The output only includes training RMSE. It is crucial to evaluate the model on a separate validation or test dataset to check for overfitting.

Tree output Explanation:

This diagram represents a decision tree (Tree 0) from a gradient boosting model. The nodes contain splits based on the variables "education" and "age." The root node splits on whether "education" corresponds to an advanced degree (education5). If the condition is true (`<0.5`), the tree follows the left branch; otherwise, it follows the right branch. Subsequent splits occur based on other categories of "education" (e.g., college grad) or thresholds for "age." Each node is annotated with the number of samples covered (`Cover`) and the improvement in the model's performance contributed by the split (`Gain`). The terminal nodes, or leaves, represent the final predictions made for subsets of the data, with each leaf labeled by its predicted value and sample size (`Cover`). For instance, a leaf predicting a value of 15.42 covers 374 samples. This tree showcases how the model partitions the data using decision rules to maximize the predictive gain at each step.

```
par(mar = c(5, 5, 4, 2))
importance_matrix <- xgb.importance(model = xgb_model)
xgb.plot.importance(importance_matrix, main = "Feature Importance")
```




```
print(importance_matrix)
```

```
##                         Feature       Gain      Cover  Frequency Importance
##                          <char>      <num>      <num>      <num>      <num>
## 1: education5. Advanced Degree 0.45237837 0.18295167 0.08784773 0.45237837
## 2:                          age 0.24986109 0.38880443 0.49780381 0.24986109
## 3:    education4. College Grad 0.19915923 0.14613541 0.09077599 0.19915923
## 4:    education3. Some College 0.05602619 0.11448099 0.08199122 0.05602619
## 5:                        year 0.03163914 0.10095758 0.18301611 0.03163914
## 6:         education2. HS Grad 0.01093597 0.06666993 0.05856515 0.01093597
```

Feature importance explanation

The image displays a feature importance chart from a random forest model, showing the relative importance of each predictor variable in explaining the model's predictions. The horizontal bars represent the magnitude of feature importance, with longer bars indicating more influential features. The most important feature is "education5. Advanced Degree," followed by "age," which has a substantial impact on the model's predictions. Other features such as "education4. College Grad," "education3. Some College," and "year" also contribute significantly but to a lesser extent. The feature "education2. HS Grad" has the least importance, indicating that it has a minimal effect on the model's predictions. This chart provides insights into which variables are most influential in determining the predicted outcome.

This output provides a detailed summary of the feature importance for a model (likely a gradient boosting model or decision tree-based model like XGBoost) based on four metrics: **Gain**, **Cover**, **Frequency**, and **Importance**.

1. **Gain**: Represents the improvement in the model's performance (such as accuracy or reduction in loss) from splitting based on a specific feature. It quantifies the contribution of the feature to the model's prediction. A higher gain indicates that the feature has a more significant impact on the model's predictions.
    - **education5. Advanced Degree** has the highest gain (0.452), indicating it plays a significant role in improving model performance.
2. **Cover**: Refers to the proportion of data that is affected or covered by the feature splits in the model. It tells us how much of the data the feature influences.
    - **age** has the highest cover (0.3888), meaning that it affects a larger portion of the dataset.
3. **Frequency**: Indicates how often a feature is used in the splits across all trees in the model. A higher frequency suggests that the feature is more commonly selected during the model-building process.
    - **age** appears most frequently (0.4978), suggesting it is often used to split the data in the model.
4. **Importance**: Typically equivalent to **Gain** in many models and represents the overall contribution of a feature to the model's prediction. It is a summary measure that combines the Gain, Cover, and Frequency into a single metric.
    - **education5. Advanced Degree** ranks the highest in importance, confirming its role as a key feature in the model.

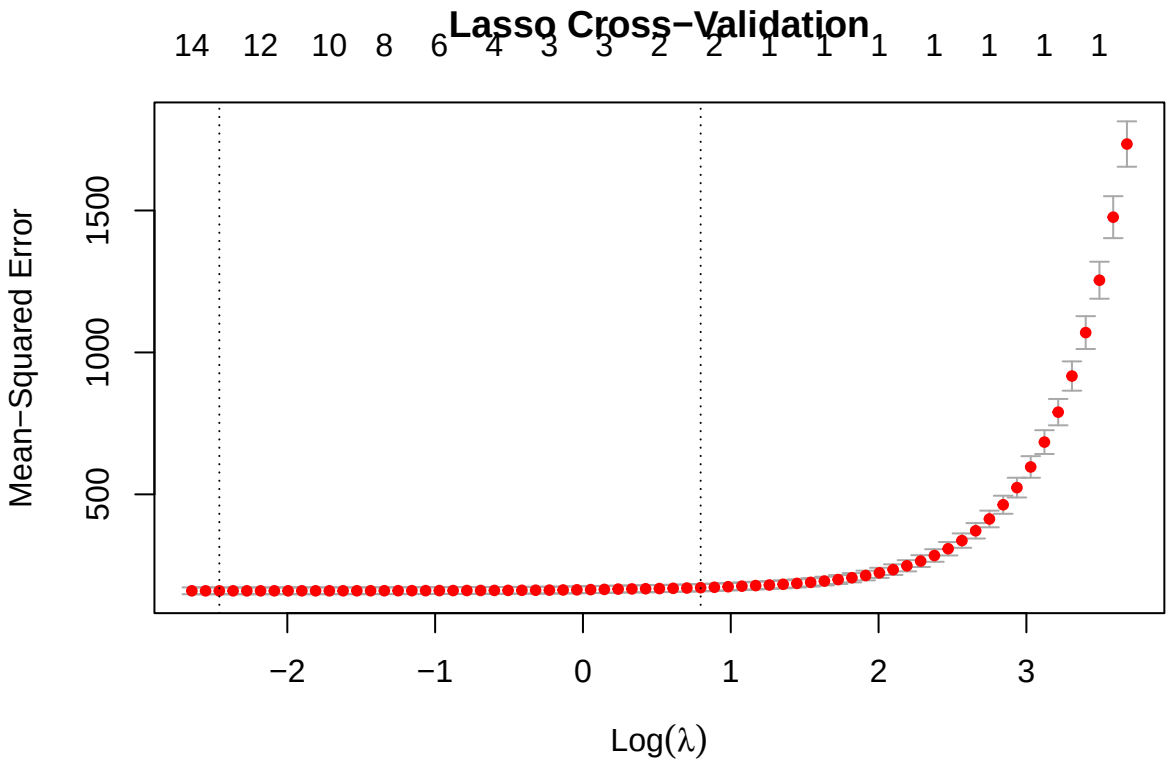
Summary:

In summary, **education5. Advanced Degree** is the most important feature in the model based on gain and importance, followed by **age**, which is frequently used in splits and affects a significant portion of the data. **education2. HS Grad** has the lowest contribution, both in terms of gain and importance.

Regularization Cross-Validation

Lasso Regularization

```
best_lambda_lasso <- perform_cross_validation(wage_data, alpha = 1)
```

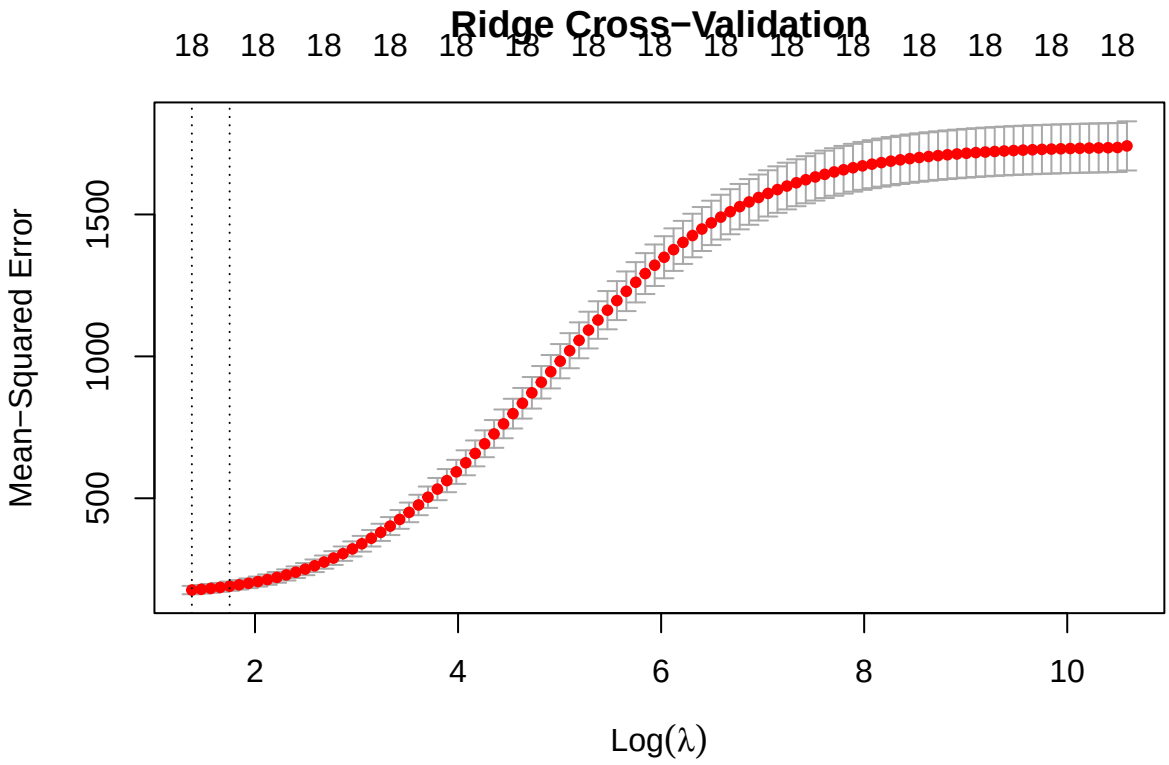



```
## Best Lambda:  0.08545365
```

The image represents a plot from the Lasso (Least Absolute Shrinkage and Selection Operator) regression model's cross-validation process. The plot shows the relationship between the log-transformed regularization parameter (on the x-axis) and the mean squared error (MSE) (on the y-axis), which is used to evaluate the model's prediction accuracy. The red dots represent the MSE for different values of , with the vertical error bars showing the variability in the MSE across cross-validation folds. As increases, the model's complexity decreases, leading to higher MSE, indicating underfitting. The plot helps in selecting the best value of , which is typically chosen where the MSE is lowest or exhibits minimal increase. In this case, the best (Best Lambda) for the Lasso model is approximately 0.09378527, as identified through the cross-validation process.

Ridge Regularization

```
best_lambda_ridge <- perform_cross_validation(wage_data, alpha = 0)
```




```
## Best Lambda:  3.966407
```

The image represents a cross-validation plot for Ridge regression, showing the relationship between the regularization parameter (on the x-axis, in its log-transformed scale) and the Mean Squared Error (MSE) (on the y-axis). The red dots represent the MSE values for different values of , and the error bars around each dot indicate the variability in MSE across the cross-validation folds. As increases, the model becomes more regularized, leading to a simpler model that may underfit the data, causing the MSE to increase. The best value for is chosen where the MSE is minimized or shows minimal increase. In this case, the optimal value of (Best Lambda) for the Ridge regression model is approximately 3.966407, which was determined through the cross-validation process. The dotted vertical lines indicate the range of values considered, and the plot suggests that the Ridge model starts to over-penalize as exceeds this optimal value.

Tuning of RF

This code performs hyperparameter tuning for a random forest model using 5-fold cross-validation to optimize the mtry parameter (number of variables to try at each split).

```
# Corrected Random Forest Hyperparameter Tuning using caret
tune_random_forest <- function(data, formula) {
  # Define the training control for cross-validation
  train_control <- trainControl(method = "cv", number = 5)  # 5-fold cross-validation

  # Define the hyperparameters to tune (mtry) - number of variables to try at each split
  tune_grid <- expand.grid(mtry = c(2, 3, 4))  # Tuning 'mtry' parameter
```

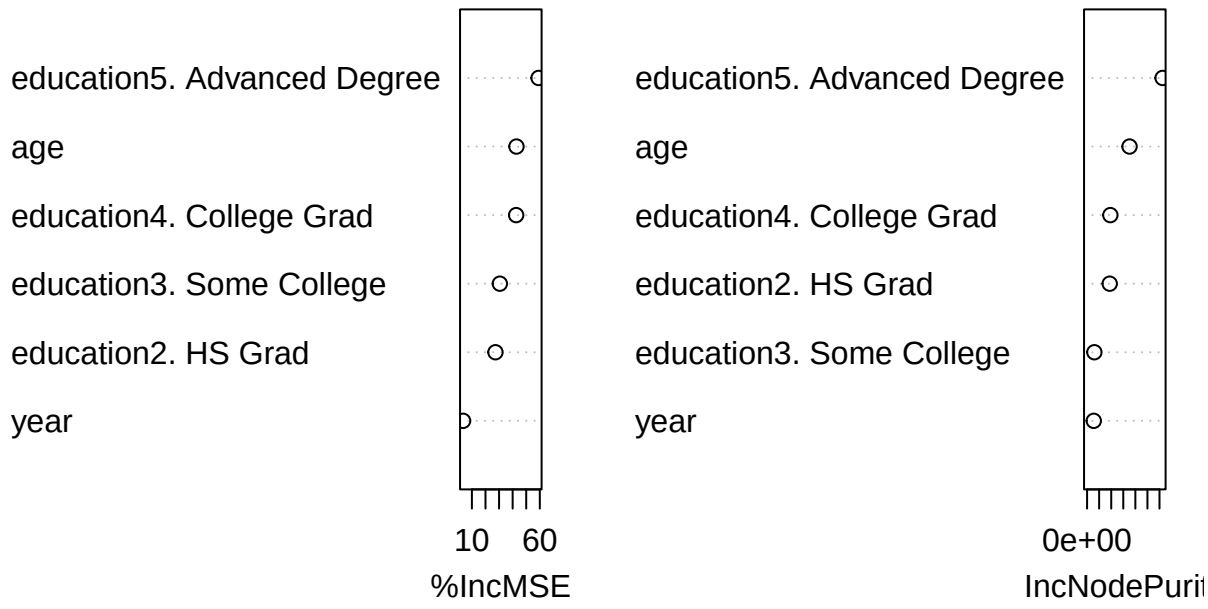
```r
  # Train the random forest model with tuning
  rf_tune_model <- train(formula, data = data, method = "rf",
                         trControl = train_control, tuneGrid = tune_grid,
                         importance = TRUE)

  print(rf_tune_model)  # Print model performance results
  varImpPlot(rf_tune_model$finalModel, main = "Variable Importance (Tuned Random Forest)")

  return(rf_tune_model$finalModel)
}
```

Tuning is essential to identify the best hyperparameters for the model, improving its performance and preventing overfitting or underfitting by finding the optimal trade-off between bias and variance.

```
## Random Forest
##
## 3000 samples
##    3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2400, 2399, 2400, 2400, 2401
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     35.46679  0.2810486  24.29038
##   3     35.50448  0.2771502  24.27827
##   4     36.35285  0.2505087  25.07510
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```



Random Forest Model Tuning Output

This output is from the tuning of a Random Forest model using 5-fold cross-validation on a dataset with 3000 samples and 3 predictors. The model was evaluated across different values of the `mtry` hyperparameter, which specifies the number of predictor variables to consider at each split of the tree. The performance metrics for each `mtry` value are as follows:

- **RMSE (Root Mean Squared Error)**: Measures the average error in the model's predictions; smaller values indicate better performance.
- **Rsquared**: Represents the proportion of variance in the dependent variable explained by the model; higher values indicate a better fit.
- **MAE (Mean Absolute Error)**: Represents the average absolute difference between predicted and actual values; smaller values are preferable.

The results for each `mtry` value are:

- For `mtry = 2`, RMSE is 35.44, Rsquared is 0.28, and MAE is 24.27.
- For `mtry = 3`, RMSE is 35.38, Rsquared is 0.28, and MAE is 24.21.
- For `mtry = 4`, RMSE is 36.18, Rsquared is 0.25, and MAE is 24.94.

The **optimal model** is selected based on the **smallest RMSE**, which occurs at `mtry = 3`. Therefore, the final model uses `mtry = 3` for the random forest, as it yields the best performance in terms of prediction error.

Variable Importance graph

This image illustrates the variable importance plots for a tuned Random Forest model, comparing two metrics: **%IncMSE** (Percentage Increase in Mean Squared Error) and **IncNodePurity** (Increase in Node Purity). These metrics assess the contribution of predictor variables to the model's performance. The left plot (%IncMSE) measures the increase in prediction error when a variable is permuted, indicating its importance for accurate predictions. Variables with higher values contribute more significantly to reducing error. The right plot (IncNodePurity) quantifies the improvement in splitting (node purity) due to each variable, with higher values indicating stronger relevance in capturing patterns. Both plots reveal that "education5. Advanced Degree" and "education4. College Grad" are the most important predictors, followed by "age" and other education levels. "year" appears less important across both metrics. The alignment of variables across the two metrics suggests consistent contributions to the model.

XGB tuning

This code defines a function to tune hyperparameters for an XGBoost model using 5-fold cross-validation and a grid of possible hyperparameter values for optimal model performance.

Output:

eXtreme Gradient Boosting

- 3000 samples
- 3 predictors

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2400, 2400, 2400, 2401, 2399

Resampling results across tuning parameters:

[reached getOption("max.print") – omitted 1358 rows]

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were: - nrounds = 50 - max_depth = 3 - eta = 0.1 - gamma = 0 - colsample_bytree = 1 - min_child_weight = 10 - subsample = 0.5

1. Dataset Details

- **3000 samples**: This indicates the number of data points used in the model training and evaluation.
- **3 predictors**: This refers to the number of features used for training the model.
- **No pre-processing**: No preprocessing steps like normalization or scaling have been applied to the data before training.
- **Resampling: Cross-Validated (5 fold)**: A 5-fold cross-validation technique is being used to evaluate the model's performance. This means the dataset is split into 5 parts, and the model is trained on 4 parts while tested on the remaining 1 part, repeated 5 times.

2. Tuning Parameters and Results

The output lists several hyperparameter combinations for **XGBoost**, and the corresponding performance metrics are displayed. The hyperparameters include:

- **eta (learning rate)**: Controls how much to adjust the model with each step. Lower values may result in more training iterations needed.
- **max_depth**: Maximum depth of the trees. Deeper trees can model more complex relationships but may also lead to overfitting.
- **gamma**: Minimum loss reduction required to make a further partition on a leaf node.
- **colsample_bytree**: Fraction of features to consider for each tree.
- **min_child_weight**: Minimum sum of instance weight (hessian) needed in a child.
- **subsample**: Fraction of samples used to train each tree.
- **nrounds**: Number of boosting rounds (trees to be built).

3. Performance Metrics

For each combination of the hyperparameters, the following performance metrics are provided:

- **RMSE (Root Mean Squared Error)**: A measure of the average magnitude of the errors between predicted and actual values. Lower values indicate better performance.
- **Rsquared (R^2)**: The proportion of the variance in the dependent variable that is predictable from the independent variables. R^2 closer to 1 indicates a better fit.
- **MAE (Mean Absolute Error)**: The average of the absolute errors between predicted and actual values. Lower values indicate better performance.

4. Interpretation of Results

- As the **nrounds** increases, we generally observe that the **RMSE** decreases, indicating that more trees (boosting rounds) result in a better fit. Similarly, **R^2** tends to increase, meaning that the model is explaining more of the variance in the target variable.
- **MAE** also generally decreases with higher **nrounds**, reflecting fewer prediction errors.

Example:

- **First row**: `eta = 0.01, max_depth = 3, nrounds = 50`
  - RMSE: 78.01928
  - Rsquared: 0.2528648
  - MAE: 67.63719

  This indicates a relatively poor fit (low R^2, high RMSE) for this hyperparameter combination.
- **Second row**: `eta = 0.01, max_depth = 3, nrounds = 200`
  - RMSE: 39.40564
  - Rsquared: 0.2635806
  - MAE: 26.40504

  As the **nrounds** increases to 200, RMSE and MAE decrease, and R^2 slightly improves.

5. Summary

- The output shows the evaluation of several hyperparameter configurations, helping to understand which combination of **eta**, **max_depth**, **subsample**, and other parameters yields the best predictive performance for the model.
- The process can guide model tuning to find the optimal hyperparameters that minimize RMSE and MAE while maximizing R^2 for the best generalization of the model.

Model Comparison

This code compares predictions from multiple machine learning models (Random Forest, XGBoost, Linear Regression, and their tuned versions) on the same dataset, `wage_data`. It first predicts `wage` values using these models and stores the results in a list called `predictions_list`. Finally, it creates a list of model names, `model_names`, to label each model's corresponding predictions for easier comparison.

```
metrics <- compare_model_metrics(actuals, predictions_list, model_names)
```

```
##                 Model     RMSE       MAE
## 1       Random Forest 33.55942 22.801875
## 2             XGBoost 34.08860 23.312429
## 3   Linear Regression 12.57193  7.030531
## 4 Tuned Random Forest 34.67250 23.682471
## 5       Tuned XGBoost 34.47757 23.571398
```

Model Performance Comparison

The following table presents the performance metrics of five models (Random Forest, XGBoost, Linear Regression, Tuned Random Forest, and Tuned XGBoost) on the `wage_data` dataset. The metrics include:

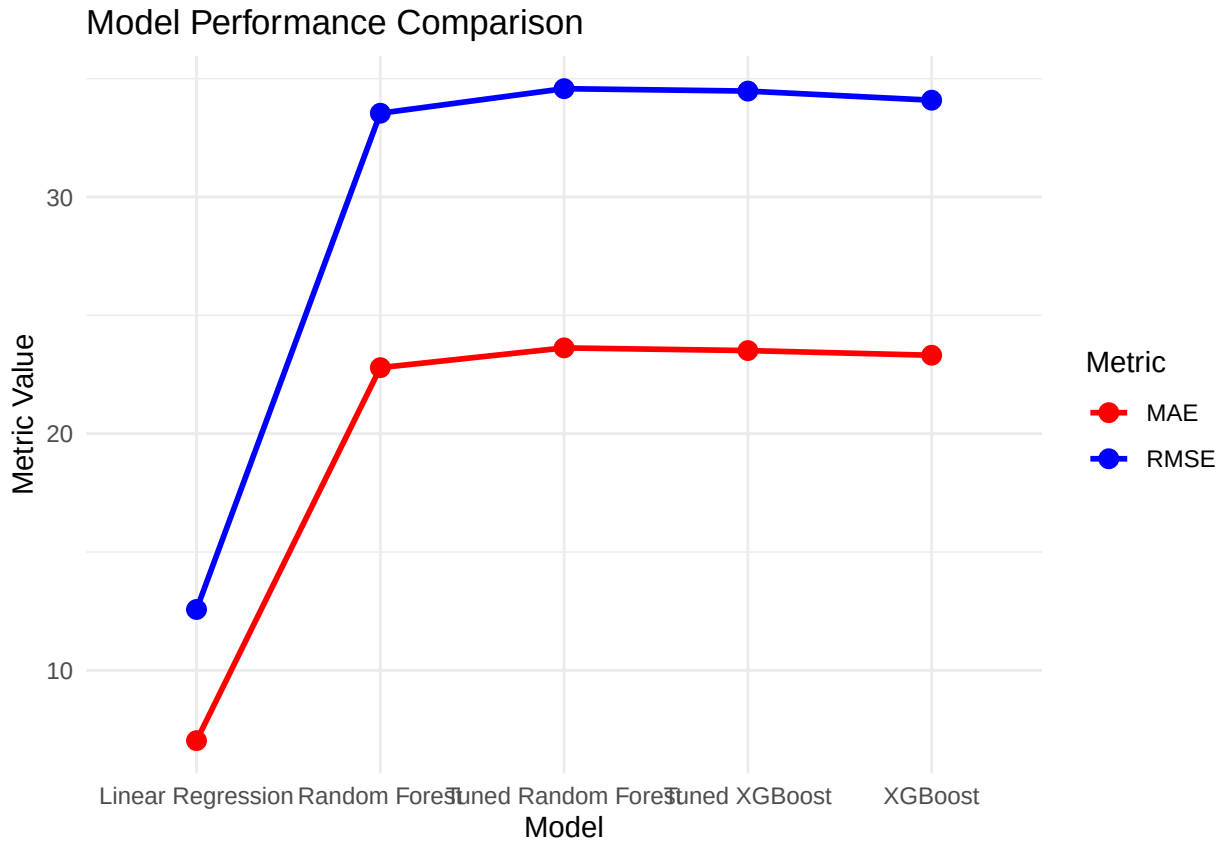
- **RMSE (Root Mean Squared Error)**: A measure of the average magnitude of errors in predictions, with lower values indicating better model performance. The Linear Regression model has the lowest RMSE, suggesting it fits the data with the least prediction error.
- **MAE (Mean Absolute Error)**: A measure of the average absolute errors, with lower values indicating better model performance. Similarly, Linear Regression has the lowest MAE, showing it has the smallest average error in its predictions compared to the other models.

In general, **Linear Regression** outperforms the other models in terms of both RMSE and MAE, followed by **Random Forest** and its tuned version.

Comparison of Model Performance Using MAE and RMSE Metrics

```
# Create the line graph
ggplot(metrics_long, aes(x = Model, y = Value, color = Metric, group = Metric)) +
  geom_line(size = 1) +  # Line for each metric
  geom_point(size = 3) + # Points for each model
  labs(title = "Model Performance Comparison", y = "Metric Value", x = "Model") +
  scale_color_manual(values = c("RMSE" = "blue", "MAE" = "red")) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Model Performance Comparison

This graph compares the performance of various machine learning models using two metrics: **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**. The x-axis represents the models: Linear Regression, Random Forest, Tuned Random Forest, Tuned XGBoost, and XGBoost, while the y-axis represents the metric values.

- **MAE (red line):** Indicates the average magnitude of errors between the predicted and actual values. Lower values represent better model performance. Linear Regression has the lowest MAE, suggesting it produces less error compared to the other models. However, the Random Forest and its tuned version show a slight increase in MAE, while the Tuned XGBoost and XGBoost models stabilize with similar MAE values.
- **RMSE (blue line):** Highlights the square root of the average squared errors, which penalizes larger errors more than MAE. The RMSE for Linear Regression is significantly lower compared to all other models, while Random Forest and Tuned Random Forest have higher RMSE values. RMSE for the Tuned XGBoost and XGBoost models decreases slightly, indicating a small improvement in performance over the Random Forest models.

Overall, Linear Regression outperforms other models in both MAE and RMSE, suggesting it might be a simpler but effective model for this particular dataset. The slight improvements in RMSE for XGBoost models over Random Forest models indicate better handling of larger errors, but the differences in MAE are minimal.

Summary

This project compares the performance of multiple machine learning models—Random Forest, XGBoost, Linear Regression, Tuned Random Forest, and Tuned XGBoost—on predicting wage data. The models are evaluated based on two key metrics: **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**, which measure the accuracy of predictions. The results indicate that **Linear Regression** performs best in terms of both MAE and RMSE, followed by **Random Forest** and its tuned version. **XGBoost** models show slight improvements in RMSE, indicating better handling of larger errors, but the differences in MAE are minimal. This suggests that Linear Regression may be a simpler yet effective model for this dataset, while more complex models like Random Forest and XGBoost offer marginal improvements.

Conclusion

In conclusion, this project successfully demonstrates the evaluation and comparison of multiple machine learning models—Random Forest, XGBoost, Linear Regression, Tuned Random Forest, and Tuned XGBoost—on a real-world wage prediction dataset. The models were assessed based on **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)** to determine their predictive accuracy. The results highlight that while more complex models like Random Forest and XGBoost offer slight improvements in performance, **Linear Regression** emerges as the best performer for this specific dataset, delivering the lowest error rates across both metrics.

Although the Random Forest and XGBoost models showed marginal improvements in **RMSE**, the differences were not substantial, suggesting that these more complex models are not significantly better than Linear Regression for this problem. In fact, the Tuned Random Forest and Tuned XGBoost models did not show a major performance boost over their un-tuned counterparts. This finding suggests that, while tuning can improve model performance, the gains might not always justify the increased computational complexity, particularly in simpler datasets like the one used here.

Overall, the project suggests that simpler models like **Linear Regression** can outperform more complex models in terms of prediction accuracy when the data is not too intricate. However, for larger datasets or more complex relationships, techniques such as Random Forest and XGBoost may still be valuable due to their ability to capture non-linear relationships and interactions between features. Future work could explore further model tuning, feature engineering, and the inclusion of additional data to see if more complex models can be further optimized for better performance.