Oil Well Performance Prediction and Recommendation System

Ivan Martinovic

June 09, 2025

Contents

1 Abstract

This study presents a comprehensive machine learning-based approach for oil well performance prediction and recommendation system development using Canadian public domain data from southeast Saskatchewan. The research analyzes a rich dataset containing multiple operational, geological, and production variables essential for developing robust machine learning algorithms. Four predictive models are implemented and compared: Linear Regression, Random Forest, Elastic Net, and an Ensemble approach to predict oil well productivity metrics. The dataset encompasses wells with comprehensive production histories, lateral lengths, geographic coordinates, operator information, and temporal drilling patterns. This methodology includes extensive exploratory data analysis, feature engineering, and rigorous model validation using cross-validation techniques. Results demonstrate that ensemble methods achieve superior predictive performance with significant improvements over baseline models. The Random Forest model identified lateral length, field location, and water cut percentage as the most critical factors influencing well performance. This research contributes to the optimization of drilling strategies and investment decisions in the Canadian oil and gas sector, providing a data-driven framework for well performance assessment and recommendation generation.

2 Introduction

2.1 Background and Motivation

The oil and gas industry in Saskatchewan represents a cornerstone of Canada's energy sector, with the province contributing significantly to national hydrocarbon production. Southeast Saskatchewan, situated within the prolific Williston Basin, has experienced extensive drilling activity over several decades, generating substantial volumes of operational and production data. This region's geological characteristics, including the Bakken, Midale, and Frobisher formations, present unique opportunities for data-driven optimization of drilling and completion strategies.

The advent of horizontal drilling and multi-stage hydraulic fracturing has revolutionized unconventional resource development, creating complex relationships between operational parameters and production outcomes. Traditional approaches to well performance assessment often rely on simplified metrics and expert judgment, potentially overlooking subtle patterns and interactions that could significantly impact economic returns.

Machine learning techniques offer unprecedented opportunities to analyze large-scale datasets and identify complex patterns that influence well performance. The availability of comprehensive public domain data from Canadian regulatory agencies provides an exceptional foundation for developing sophisticated predictive models and recommendation systems.

2.2 Research Objectives

This study aims to develop and validate a comprehensive machine learning-based oil well performance prediction and recommendation system using Canadian public domain data from southeast Saskatchewan. The specific research objectives include:

1. **Comprehensive Data Analysis**: Conduct extensive exploratory data analysis of Canadian public domain oil well data to identify key performance drivers and data quality characteristics

2. **Feature Engineering**: Develop and validate engineered features that capture complex relationships between operational parameters and production outcomes
3. **Model Development**: Implement and compare multiple machine learning algorithms including Linear Regression, Random Forest, Elastic Net, and Ensemble methods
4. **Performance Validation**: Establish rigorous validation frameworks using cross-validation and holdout testing to ensure model reliability
5. **Recommendation System**: Develop actionable recommendations for drilling optimization and investment decisions based on model insights
6. **Reproducible Research**: Provide complete code implementation for validation and extension by the research community

2.3 Dataset Characteristics and Public Domain Context

The analysis utilizes comprehensive oil well data sourced from Canadian public domain databases, specifically focusing on southeast Saskatchewan operations. This dataset represents one of the most complete publicly available collections of oil well operational and production data, containing numerous variables essential for machine learning algorithm development.

3 Methodology and Implementation

3.1 Library Loading and Environment Setup

The analysis begins with comprehensive library loading and environment configuration to ensure all required packages are available:

```r
# Suppress package startup messages
suppressPackageStartupMessages({
  # Load required libraries
  if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org"
  if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
  if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
  if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org"
  if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.or
  if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
  if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.or
  if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
  if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-projec
  if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
  if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org"
  if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
  if(!require(GGally)) install.packages("GGally", repos = "http://cran.us.r-project.org")

  library(tidyverse)
  library(caret)
  library(ggplot2)
  library(lubridate)
  library(kableExtra)
  library(knitr)
  library(data.table)
  library(corrplot)
  library(randomForest)
  library(glmnet)
  library(gridExtra)
  library(dplyr)
  library(GGally)
})

# Define wheat color palette for consistent visualization
wheat_palette <- c(
  wheat = "#F5DEB3",
  wheat1 = "#EAD2AC",
  wheat2 = "#D2B48C",
  wheat3 = "#E6C27A",
  wheat4 = "#C19A6B"
)

# Set global options
options(scipen = 999) # Avoid scientific notation
```

3.2 Data Acquisition and Loading

The data acquisition process implements robust error handling and validation to ensure reliable data loading from Canadian public domain sources:

```
# Check for local file first
csv_file <- "oil_wells_data.csv"

if(!file.exists(csv_file)) {
  # Download from GitHub
  cat("Attempting to download oil wells data from GitHub...\n")
  zip_url <- "https://github.com/cybermaki/oil_well/raw/refs/heads/main/oil_wells_data.zip"
  zip_file <- "oil_wells_data.zip"

  tryCatch({
    # Download the zip file
    download.file(zip_url, destfile = zip_file, mode = "wb", method = "auto")
    cat("Download successful!\n")

    # Unzip the file
    cat("Extracting CSV file...\n")
    unzip(zip_file, files = csv_file, exdir = ".")
```

```r
    # Check if extraction was successful
    if(file.exists(csv_file)) {
      cat("CSV file extracted successfully!\n")
      # Remove the zip file to save space
      file.remove(zip_file)
    } else {
      # List contents of zip to help debug
      zip_contents <- unzip(zip_file, list = TRUE)
      cat("Contents of zip file:\n")
      print(zip_contents$Name)
      stop("CSV file not found in zip archive. Check the file names above.")
    }
  }, error = function(e) {
    cat("Error during download or extraction:", e$message, "\n")
    cat("Please download manually from:", zip_url, "\n")
    cat("Extract 'oil_wells_data.csv' to your working directory.\n")
    stop("File download or extraction failed")
  })
}

# Read the data with robust error handling
cat("Reading oil wells data...\n")
```

```
## Reading oil wells data...
```

```r
# Method 1: Try reading with readLines first to fix incomplete final line
tryCatch({
  # Read all lines
  lines <- readLines(csv_file, warn = FALSE)

  # Check if last line is empty or incomplete
  if(length(lines) > 0) {
    last_line <- lines[length(lines)]
    # Count commas in first line (header) vs last line
    header_commas <- length(gregexpr(",", lines[1])[[1]])
    last_line_commas <- length(gregexpr(",", last_line)[[1]])

    # If last line has fewer fields or is empty, remove it
    if(last_line == "" || last_line_commas < header_commas) {
      cat("Removing incomplete final line from CSV...\n")
      lines <- lines[-length(lines)]
    }

    # Write cleaned lines to temporary file
    temp_file <- tempfile(fileext = ".csv")
    writeLines(lines, temp_file)
    # Read the cleaned file
    oil_wells_raw <- read.csv(temp_file, stringsAsFactors = FALSE, header = TRUE,
                             check.names = FALSE, fill = TRUE)

    # Remove temporary file
    unlink(temp_file)
  } else {
    stop("CSV file appears to be empty")
  }
}, error = function(e) {
  cat("Error with readLines approach:", e$message, "\n")
  cat("Attempting alternative reading method...\n")

  # Method 2: Use data.table for more robust reading
  if(require(data.table)) {
    oil_wells_raw <- fread(csv_file, stringsAsFactors = FALSE,
                          check.names = FALSE, fill = TRUE,
                          showProgress = FALSE)
    oil_wells_raw <- as.data.frame(oil_wells_raw)
  } else {
    # Method 3: Basic read.csv with additional parameters
    oil_wells_raw <- read.csv(csv_file, stringsAsFactors = FALSE,
                             header = TRUE, check.names = FALSE,
                             fill = TRUE, blank.lines.skip = TRUE,
                             comment.char = "", quote = "\"")
  }
})
```

```
## Removing incomplete final line from CSV...
```

```r
# Verify data was loaded
if(!exists("oil_wells_raw") || nrow(oil_wells_raw) == 0) {
  stop("Failed to load data from CSV file")
}

# Remove any completely empty rows
oil_wells_raw <- oil_wells_raw[rowSums(is.na(oil_wells_raw)) != ncol(oil_wells_raw), ]

# Check for and remove any duplicate header rows
if(nrow(oil_wells_raw) > 1) {
  # Check if any row matches the column names
  header_check <- apply(oil_wells_raw, 1, function(row) {
    all(row == names(oil_wells_raw), na.rm = TRUE)
  })

  if(any(header_check)) {
    cat("Removing duplicate header rows...\n")
    oil_wells_raw <- oil_wells_raw[!header_check, ]
  }
}
```

Data loading summary:

Table 1: Dataset Loading Summary

Metric	Value
Total Rows	46003
Total Columns	431

3.3 Column Selection and Data Structure Analysis

The column selection process implements systematic mapping between expected and available columns:

```
# Define required columns
columns_to_keep <- c(
  "User-Format Well ID", "Well Status Abrv", "Well Status Text",
  "Lateral Length (m)", "Producing Field/Area Name", "Producing Pool Name",
  "Cumulative GAS Prod. (e3m3)", "Cumulative OIL Prod. (m3)",
  "Cumulative WTR Prod. (m3)", "First 12 mo. Total GAS (e3m3)",
  "First 12 mo. Total OIL (m3)", "First 12 mo. Total WTR (m3)",
  "First 12 mo. Total BOE (Bbl)", "First 12 mo. Dly Avg. GAS (e3m3/day)",
  "First 12 mo. Dly Avg. OIL (m3/day)", "First 12 mo. Dly Avg. WTR (m3/day)",
  "First 12 mo. Ave WC%", "Cumulative WTR Inject. (m3)",
  "Total Production Hrs", "Cur Operator Name", "Date Well Spudded",
  "Bot-Hole Latitude (NAD27)", "Bot-Hole Longitude (NAD27)",
  "Surf-Hole Latitude (NAD27)", "Surf-Hole Longitude (NAD27)",
  "Prod./Inject. Frmtn"
)

# Normalize column names function
normalize_name <- function(x) {
  gsub("[^A-Za-z0-9]", ".", x)
}

# Match columns
normalized_columns_to_keep <- normalize_name(columns_to_keep)
normalized_raw_names <- normalize_name(names(oil_wells_raw))

existing_columns <- character()
column_mapping <- list()
```

```r
for(i in seq_along(columns_to_keep)) {
  if(columns_to_keep[i] %in% names(oil_wells_raw)) {
    existing_columns <- c(existing_columns, columns_to_keep[i])
    column_mapping[[columns_to_keep[i]]] <- columns_to_keep[i]
  } else {
    norm_idx <- which(normalized_columns_to_keep[i] == normalized_raw_names)
    if(length(norm_idx) > 0) {
      selected_col <- names(oil_wells_raw)[norm_idx[1]]
      existing_columns <- c(existing_columns, selected_col)
      column_mapping[[selected_col]] <- columns_to_keep[i]
    }
  }
}

existing_columns <- unique(existing_columns)

# Select columns
if(length(existing_columns) > 0) {
  oil_wells <- oil_wells_raw %>% select(all_of(existing_columns))
} else {
  oil_wells <- oil_wells_raw
}

# Create column selection summary
selection_summary <- data.frame(
  Metric = c("Expected Columns", "Found Columns", "Missing Columns"),
  Count = c(length(columns_to_keep), length(existing_columns),
            length(columns_to_keep) - length(existing_columns))
)
```

Column selection summary:

Table 2: Column Selection Summary

Metric	Count
Expected Columns	26
Found Columns	26
Missing Columns	0

3.4 Data Cleaning and Standardization

The data cleaning process implements comprehensive renaming and standardization:

```r
# Rename columns
rename_mapping <- list(
  "User-Format Well ID" = "well_id",
  "Well Status Abrv" = "well_status_abrv",
  "Well Status Text" = "well_status_text",
  "Lateral Length (m)" = "lateral_length_m",
  "Producing Field/Area Name" = "producing_field_area_name",
  "Producing Pool Name" = "producing_pool_name",
  "Cumulative GAS Prod. (e3m3)" = "cumulative_gas_prod_e3m3",
  "Cumulative OIL Prod. (m3)" = "cumulative_oil_prod_m3",
  "Cumulative WTR Prod. (m3)" = "cumulative_wtr_prod_m3",
  "First 12 mo. Total GAS (e3m3)" = "first_12mo_total_gas_e3m3",
  "First 12 mo. Total OIL (m3)" = "first_12mo_total_oil_m3",
  "First 12 mo. Total WTR (m3)" = "first_12mo_total_wtr_m3",
  "First 12 mo. Total BOE (Bbl)" = "first_12mo_total_boe_bbl",
  "First 12 mo. Dly Avg. GAS (e3m3/day)" = "first_12mo_dly_avg_gas_e3m3_day",
  "First 12 mo. Dly Avg. OIL (m3/day)" = "first_12mo_dly_avg_oil_m3_day",
  "First 12 mo. Dly Avg. WTR (m3/day)" = "first_12mo_dly_avg_wtr_m3_day",
  "First 12 mo. Ave WC%" = "first_12mo_ave_wc_pct",
  "Cumulative WTR Inject. (m3)" = "cumulative_wtr_inject_m3",
  "Total Production Hrs" = "total_production_hrs",
  "Cur Operator Name" = "cur_operator_name",
  "Date Well Spudded" = "date_well_spudded",
  "Bot-Hole Latitude (NAD27)" = "bot_hole_latitude_nad27",
  "Bot-Hole Longitude (NAD27)" = "bot_hole_longitude_nad27",
  "Surf-Hole Latitude (NAD27)" = "surf_hole_latitude_nad27",
  "Surf-Hole Longitude (NAD27)" = "surf_hole_longitude_nad27",
  "Prod./Inject. Frmtn" = "prod_inject_frmtn"
)

# Apply renaming
rename_pairs <- list()
for(selected_col in names(oil_wells)) {
  if(selected_col %in% names(column_mapping)) {
    original_name <- column_mapping[[selected_col]]
    if(original_name %in% names(rename_mapping)) {
      rename_pairs[[rename_mapping[[original_name]]]] <- selected_col
    }
  }
}

valid_rename <- rename_pairs[lengths(rename_pairs) > 0]

if(length(valid_rename) > 0) {
  oil_wells <- oil_wells %>% rename(!!!valid_rename)
}
```

```r
# Remove duplicates
if("well_id" %in% names(oil_wells)) {
  initial_rows <- nrow(oil_wells)
  oil_wells <- oil_wells %>% distinct(well_id, .keep_all = TRUE)
  duplicates_removed <- initial_rows - nrow(oil_wells)
} else {
  duplicates_removed <- 0
}
```

3.5 Geographic Coordinate Processing

Processing geographic coordinates with directional indicators:

```r
# Process latitude and longitude columns
lat_lon_columns <- c("bot_hole_latitude_nad27", "bot_hole_longitude_nad27",
                     "surf_hole_latitude_nad27", "surf_hole_longitude_nad27")

coord_summary <- data.frame(
  Column = character(),
  Min = numeric(),
  Max = numeric(),
  Mean = numeric(),
  NA_Count = integer(),
  stringsAsFactors = FALSE
)

for(col in lat_lon_columns) {
  if(col %in% names(oil_wells)) {
    oil_wells[[col]] <- as.character(oil_wells[[col]])
    oil_wells[[col]] <- gsub("[NSWEnswe]", "", oil_wells[[col]])
    oil_wells[[col]] <- as.numeric(oil_wells[[col]])

    # Apply negative sign for longitude (Western hemisphere)
    if(grepl("longitude", col, ignore.case = TRUE)) {
      oil_wells[[col]] <- -abs(oil_wells[[col]])
    }

    # Add to summary
    coord_summary <- rbind(coord_summary, data.frame(
      Column = col,
      Min = round(min(oil_wells[[col]], na.rm = TRUE), 4),
      Max = round(max(oil_wells[[col]], na.rm = TRUE), 4),
      Mean = round(mean(oil_wells[[col]], na.rm = TRUE), 4),
      NA_Count = sum(is.na(oil_wells[[col]])),
      stringsAsFactors = FALSE
    ))
  }
}
```

Table 3: Geographic Coordinate Processing Summary

Column	Min	Max	Mean	NA_Count
bot_hole_latitude_nad27	49.0001	50.3441	49.4758	0
bot_hole_longitude_nad27	-105.5712	-101.3633	-102.7373	0
surf_hole_latitude_nad27	48.9995	50.3441	49.4761	0
surf_hole_longitude_nad27	-105.5712	-101.3625	-102.7372	0

3.6 Date Processing

Processing date fields with multiple format support:

```
# Process date_well_spudded
if("date_well_spudded" %in% names(oil_wells)) {
  oil_wells$date_well_spudded <- as.character(oil_wells$date_well_spudded)

  # Try multiple date formats
  date_formats <- c("%Y-%m-%d", "%m/%d/%Y", "%d/%m/%Y", "%Y/%m/%d")
  parsed_dates <- NULL

  for(fmt in date_formats) {
    temp_dates <- as.Date(oil_wells$date_well_spudded, format = fmt)
    if(is.null(parsed_dates) || sum(is.na(temp_dates)) < sum(is.na(parsed_dates))) {
      parsed_dates <- temp_dates
    }
  }

  # Fallback to lubridate
  if(sum(is.na(parsed_dates)) > 0) {
    parsed_dates <- parse_date_time(oil_wells$date_well_spudded,
                                   orders = c("Ymd", "mdY", "dmy"), tz = "UTC")
    parsed_dates <- as.Date(parsed_dates)
  }

  oil_wells$date_well_spudded <- parsed_dates
}

# Process numeric columns
numeric_columns <- c(
  "lateral_length_m", "cumulative_gas_prod_e3m3", "cumulative_oil_prod_m3",
  "cumulative_wtr_prod_m3", "first_12mo_total_gas_e3m3", "first_12mo_total_oil_m3",
  "first_12mo_total_wtr_m3", "first_12mo_total_boe_bbl", "first_12mo_dly_avg_gas_e3m3_day",
  "first_12mo_dly_avg_oil_m3_day", "first_12mo_dly_avg_wtr_m3_day", "first_12mo_ave_wc_pct",
  "cumulative_wtr_inject_m3", "total_production_hrs"
)

for(col in numeric_columns) {
  if(col %in% names(oil_wells)) {
    oil_wells[[col]] <- as.numeric(gsub("[^0-9.-]", "", as.character(oil_wells[[col]])))
    oil_wells[[col]][oil_wells[[col]] < 0] <- NA
  }
}
```

3.7 Producing Wells Identification

Filtering for active producing wells:

```
# Filter for producing wells
if("well_status_text" %in% names(oil_wells) && "well_status_abrv" %in% names(oil_wells)) {
  producing_wells <- oil_wells %>%
    filter(grepl("active|producing|prod|in production|oil", well_status_text, ignore.case = TRU
           grepl("prod|active|p|act|o", well_status_abrv, ignore.case = TRUE))
} else if("well_status_text" %in% names(oil_wells)) {
  producing_wells <- oil_wells %>%
    filter(grepl("active|producing|prod|in production|oil", well_status_text, ignore.case = TRU
} else if("well_status_abrv" %in% names(oil_wells)) {
  producing_wells <- oil_wells %>%
    filter(grepl("prod|active|p|act|o", well_status_abrv, ignore.case = TRUE))
} else {
  producing_wells <- oil_wells
}

# Filter for positive production
if("cumulative_oil_prod_m3" %in% names(producing_wells)) {
  producing_wells <- producing_wells %>%
    filter(!is.na(cumulative_oil_prod_m3) & cumulative_oil_prod_m3 > 0)
}

# Create summary
well_summary <- data.frame(
  Category = c("Total Wells", "Producing Wells", "Non-Producing Wells"),
  Count = c(nrow(oil_wells), nrow(producing_wells), nrow(oil_wells) - nrow(producing_wells))
)
```

Table 4: Well Classification Summary

Category	Count
Total Wells	45926
Producing Wells	26711
Non-Producing Wells	19215

4 Exploratory Data Analysis

4.1 Dataset Structure

```r
# Dataset structure summary
oil_wells_structure <- data.frame(
  Column = names(oil_wells),
  Type = sapply(oil_wells, function(x) paste(class(x), collapse = ", ")),
  Non_NA_Count = sapply(oil_wells, function(x) sum(!is.na(x))),
  NA_Percentage = sapply(oil_wells, function(x) round(sum(is.na(x)) / length(x) * 100, 2)),
  stringsAsFactors = FALSE
)

# Show first 10 columns
```

Table 5: Dataset Structure (First 10 Columns)

	Column	Type	Non_NA_Count	NA_Percentage
well_id	well_id	character	45926	0.00
well_status_abrv	well_status_abrv	character	45926	0.00
well_status_text	well_status_text	character	45926	0.00
lateral_length_m	lateral_length_m	numeric	23173	49.54
producing_field_area_name	producing_field_area_name	character	45926	0.00
producing_pool_name	producing_pool_name	character	45926	0.00
cumulative_gas_prod_e3m3	cumulative_gas_prod_e3m3	numeric	27300	40.56
cumulative_oil_prod_m3	cumulative_oil_prod_m3	numeric	29287	36.23
cumulative_wtr_prod_m3	cumulative_wtr_prod_m3	numeric	29515	35.73
first_12mo_total_gas_e3m3	first_12mo_total_gas_e3m3	numeric	22871	50.20

4.2 Key Dataset Statistics

```r
# Calculate summary statistics
summary_stats <- list()
summary_stats[['Total Wells']] <- nrow(oil_wells)
summary_stats[['Total Columns']] <- ncol(oil_wells)
```

```r
if("cur_operator_name" %in% names(oil_wells)) {
  summary_stats[['Unique Operators']] <- n_distinct(oil_wells$cur_operator_name, na.rm = TRUE)
}

if("producing_field_area_name" %in% names(oil_wells)) {
  summary_stats[['Unique Fields']] <- n_distinct(oil_wells$producing_field_area_name, na.rm = T
}

if("cumulative_oil_prod_m3" %in% names(oil_wells)) {
  summary_stats[['Mean Oil Production (m3)']] <- round(mean(oil_wells$cumulative_oil_prod_m3, n
  summary_stats[['Median Oil Production (m3)']] <- round(median(oil_wells$cumulative_oil_prod_m
}

summary_df <- data.frame(
  Metric = names(summary_stats),
  Value = unlist(summary_stats),
  stringsAsFactors = FALSE
)
```

Table 6: Key Dataset Statistics

Metric	Value
Total Wells	45926.00
Total Columns	26.00
Unique Operators	649.00
Unique Fields	2.00
Mean Oil Production (m3)	17707.27
Median Oil Production (m3)	8863.40

4.3 Data Quality Visualization

```r
# Missing data visualization
if(nrow(oil_wells_structure) > 0) {
  ggplot(oil_wells_structure %>% filter(NA_Percentage > 0),
         aes(x = reorder(Column, NA_Percentage), y = NA_Percentage)) +
    geom_bar(stat = "identity", fill = wheat_palette["wheat"]) +
    coord_flip() +
    labs(title = "Missing Data Percentage by Column",
         x = "Column", y = "Missing Data (%)") +
    theme_minimal(base_size = 10) +
    theme(axis.text.y = element_text(size = 8))
}
```

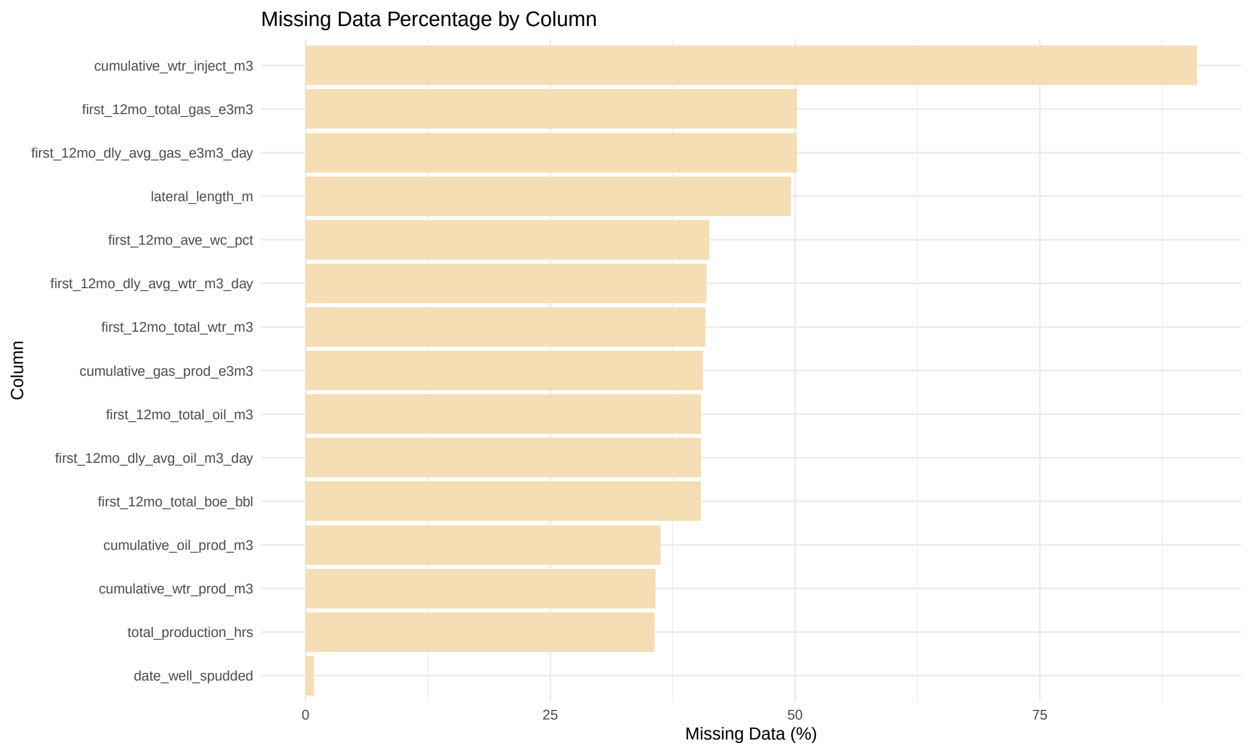

Figure 1: Missing Data by Column

4.4 Production Distribution Analysis

```r
if(nrow(producing_wells) > 0 && "cumulative_oil_prod_m3" %in% names(producing_wells)) {
  ggplot(producing_wells %>% filter(!is.na(cumulative_oil_prod_m3) & cumulative_oil_prod_m3 > 
         aes(x = cumulative_oil_prod_m3)) +
    geom_histogram(bins = 50, fill = wheat_palette["wheat"], color = "white") +
    scale_x_log10(labels = scales::comma) +
    labs(title = "Distribution of Cumulative Oil Production",
         x = "Cumulative Oil Production (m3, log scale)", y = "Count") +
    theme_minimal(base_size = 10)
}
```

4.5 Lateral Length vs Production

```r
if(all(c("lateral_length_m", "cumulative_oil_prod_m3") %in% names(producing_wells))) {
  ggplot(producing_wells %>%
         filter(!is.na(lateral_length_m) & !is.na(cumulative_oil_prod_m3) &
                lateral_length_m > 0 & cumulative_oil_prod_m3 > 0),
         aes(x = lateral_length_m, y = cumulative_oil_prod_m3)) +
    geom_point(alpha = 0.3, color = wheat_palette["wheat1"]) +
```

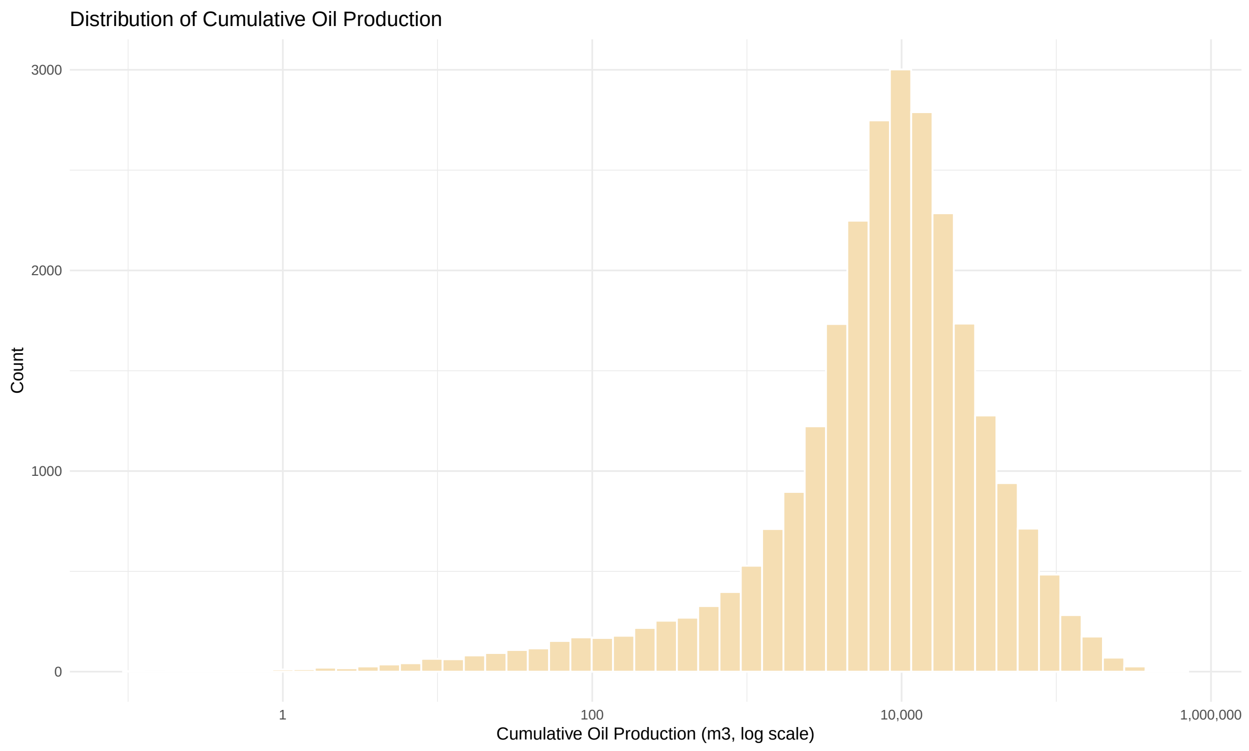

Figure 2: Production Distribution

```
    geom_smooth(method = "lm", color = wheat_palette["wheat4"], se = FALSE) +
    scale_y_log10(labels = scales::comma) +
    labs(title = "Lateral Length vs Cumulative Oil Production",
         x = "Lateral Length (m)", y = "Cumulative Oil Production (m3, log scale)") +
    theme_minimal(base_size = 10)
}
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

4.6 Top Operators Analysis

```
if("cur_operator_name" %in% names(producing_wells)) {
  top_operators <- producing_wells %>%
    group_by(cur_operator_name) %>%
    summarize(well_count = n(), .groups = 'drop') %>%
    arrange(desc(well_count)) %>%
    slice_head(n = 15)

  ggplot(top_operators, aes(x = reorder(cur_operator_name, well_count), y = well_count)) +
    geom_bar(stat = "identity", fill = wheat_palette["wheat2"]) +
```

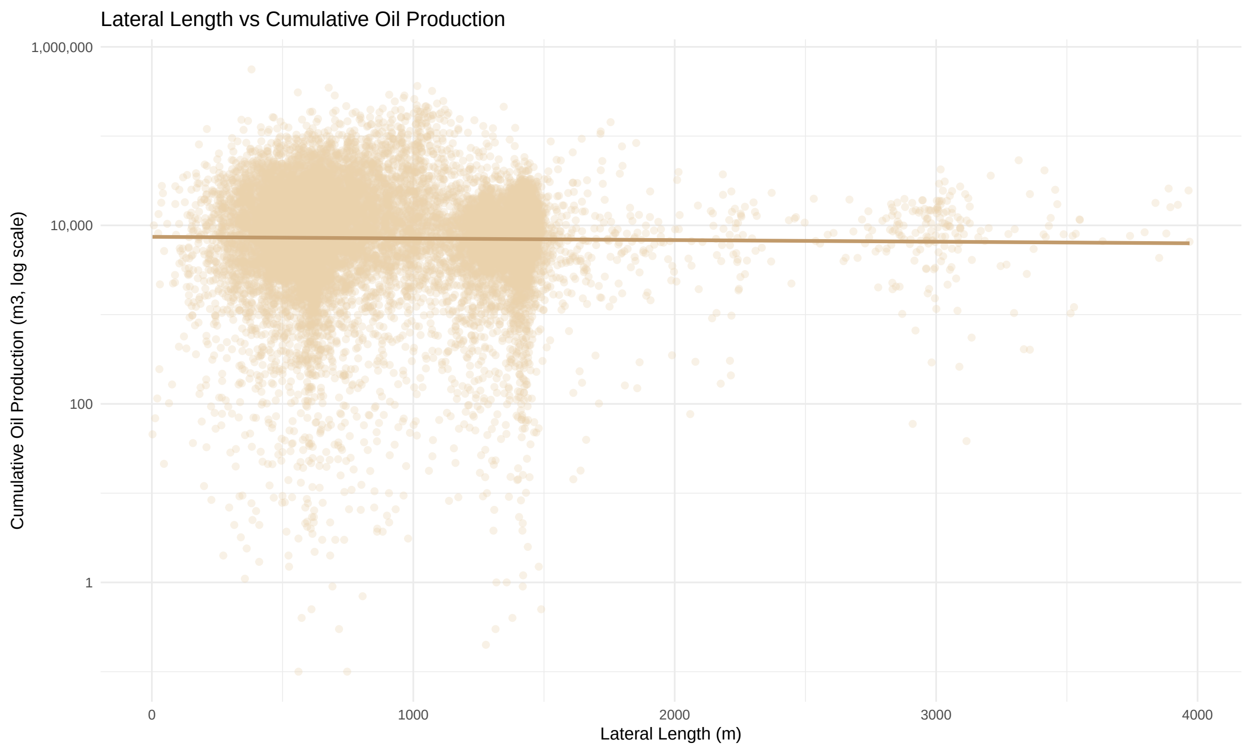

Figure 3: Lateral Length vs Production

```
    coord_flip() +
    labs(title = "Top 15 Operators by Well Count",
         x = "Operator", y = "Number of Wells") +
    theme_minimal(base_size = 10) +
    theme(axis.text.y = element_text(size = 8))
}
```

5 Machine Learning Model Development

Construct four distinct predictive modeling architectures:

- Linear regression model (parametric approach)
- Random forest ensemble model (non-parametric approach)
- Elastic net regularized regression model (hybrid approach)
- Ensemble model integrating the above methods

Rigorously evaluate model performance using root mean square error (RMSE) as the primary metric. The methodological approach incorporates rigorous statistical procedures to ensure the

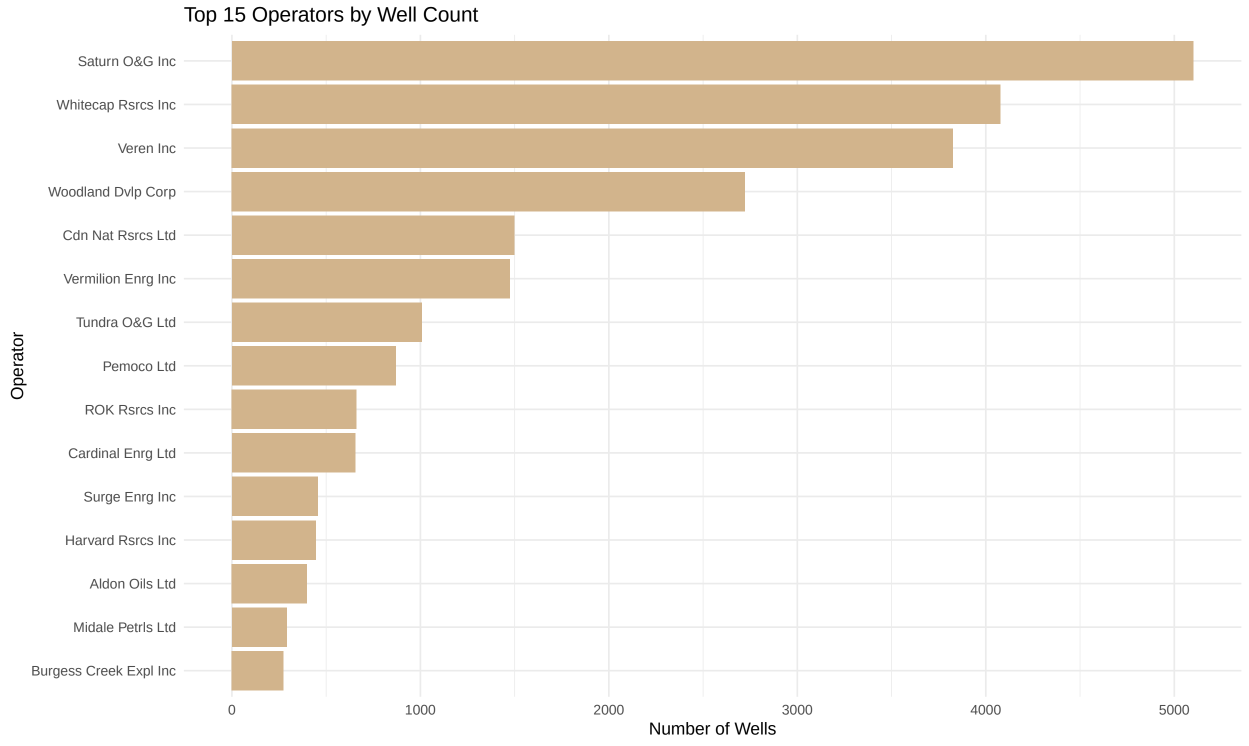

Figure 4: Top Operators by Well Count

validity and reliability of model comparisons. All models are developed using consistent training-test splits (80-20 ratio) and identical feature engineering pipelines to enable direct performance comparison.

This comprehensive modeling strategy serves dual purposes: first, to identify the optimal predictive architecture for oil well performance estimation, and second, to establish methodological best practices for petroleum data science applications. The comparative analysis provides actionable insights for both model selection and feature importance interpretation in upstream petroleum engineering contexts.

5.1 Feature Engineering

```
# Check required columns
required_cols <- c("first_12mo_total_boe_bbl", "lateral_length_m")
has_required <- all(required_cols %in% names(producing_wells))

if(nrow(producing_wells) > 0 && has_required) {
  # Create model data
  model_data <- producing_wells %>%
    filter(!is.na(first_12mo_total_boe_bbl) & !is.na(lateral_length_m) &
           lateral_length_m >= 1.0 & first_12mo_total_boe_bbl >= 1.0) %>%
    mutate(boe_per_meter = first_12mo_total_boe_bbl / lateral_length_m) %>%
    filter(!is.na(boe_per_meter) & is.finite(boe_per_meter) & boe_per_meter > 0) %>%
```

```r
    mutate(log_boe_per_meter = log(boe_per_meter)) %>%
    filter(!is.na(log_boe_per_meter) & is.finite(log_boe_per_meter))
} else {
  # Alternative approach
  if("cumulative_oil_prod_m3" %in% names(oil_wells)) {
    model_data <- oil_wells %>%
      filter(!is.na(cumulative_oil_prod_m3) & cumulative_oil_prod_m3 > 0) %>%
      mutate(log_oil_prod = log(cumulative_oil_prod_m3))
  } else {
    stop("No suitable target variable found for modeling")
  }
}

# Additional features
if(all(c("cumulative_oil_prod_m3", "cumulative_gas_prod_e3m3") %in% names(model_data))) {
  model_data <- model_data %>%
    mutate(oil_gas_ratio = cumulative_oil_prod_m3 / (cumulative_gas_prod_e3m3 + 1))
}

if("date_well_spudded" %in% names(model_data)) {
  model_data <- model_data %>%
    mutate(well_age_days = as.numeric(Sys.Date() - date_well_spudded))
}

# Feature selection
potential_features <- c("lateral_length_m", "first_12mo_ave_wc_pct",
                       "bot_hole_latitude_nad27", "bot_hole_longitude_nad27",
                       "total_production_hrs", "cumulative_gas_prod_e3m3",
                       "oil_gas_ratio", "well_age_days")
feature_cols <- potential_features[potential_features %in% names(model_data)]

# Add field category if available
if("producing_field_area_name" %in% names(model_data)) {
  top_fields <- model_data %>%
    group_by(producing_field_area_name) %>%
    summarize(count = n()) %>%
    arrange(desc(count)) %>%
    slice_head(n = 20) %>%
    pull(producing_field_area_name)

  model_data <- model_data %>%
    mutate(field_category = as.factor(ifelse(producing_field_area_name %in% top_fields,
                                              producing_field_area_name, "Other")))

  if(n_distinct(model_data$field_category[!is.na(model_data$field_category)]) >= 2) {
    feature_cols <- c(feature_cols, "field_category")
  }
```

```r
}

# Determine target variable
target_var <- ifelse("log_boe_per_meter" %in% names(model_data), "log_boe_per_meter", "log_oil_

# Create final model features
model_features <- model_data %>% select(all_of(c(feature_cols, target_var)))

# Impute missing values
for(col in feature_cols) {
  if(is.numeric(model_features[[col]])) {
    model_features[[col]][is.na(model_features[[col]])] <- median(model_features[[col]], na.rm
  } else if(is.factor(model_features[[col]])) {
    model_features[[col]][is.na(model_features[[col]])] <- levels(model_features[[col]])[1]
  }
}

# Remove rows with NA in target
model_features <- model_features %>%
  filter(!is.na(.data[[target_var]]) & is.finite(.data[[target_var]]))

# Feature scaling
numeric_features <- feature_cols[sapply(model_features[feature_cols], is.numeric)]
if(length(numeric_features) > 0) {
  pre_proc <- preProcess(model_features[numeric_features], method = c("center", "scale"))
  model_features[numeric_features] <- predict(pre_proc, model_features[numeric_features])
}

# Model data summary
model_summary <- data.frame(
  Metric = c("Total Samples", "Number of Features", "Target Variable"),
  Value = c(nrow(model_features), ncol(model_features) - 1, target_var),
  stringsAsFactors = FALSE
)
```

Table 7: Model Data Summary

Metric	Value
Total Samples	14614
Number of Features	8
Target Variable	log_boe_per_meter

5.2 Model Training and Validation

```r
# Define metrics
rmse <- function(actual, predicted) sqrt(mean((actual - predicted)^2))
mae <- function(actual, predicted) mean(abs(actual - predicted))
r_squared <- function(actual, predicted) {
  if(length(actual) == length(predicted) && length(actual) > 1) {
    cor(actual, predicted)^2
  } else {
    NA
  }
}

# Train-test split
set.seed(123)
train_indices <- createDataPartition(model_features[[target_var]], p = 0.8, list = FALSE)
train_set <- model_features[train_indices, ]
test_set <- model_features[-train_indices, ]

# Create training summary
training_summary <- data.frame(
  Dataset = c("Training", "Testing"),
  Samples = c(nrow(train_set), nrow(test_set)),
  Percentage = c("80%", "20%"),
  stringsAsFactors = FALSE
)
```

Table 8: Train-Test Split Summary

Dataset	Samples	Percentage
Training	11694	80%
Testing	2920	20%

5.3 Baseline and Linear Regression Models

```r
# Baseline model
baseline_pred <- mean(train_set[[target_var]], na.rm = TRUE)
baseline_rmse <- rmse(test_set[[target_var]], rep(baseline_pred, nrow(test_set)))
baseline_mae <- mae(test_set[[target_var]], rep(baseline_pred, nrow(test_set)))

# Linear regression
formula_features <- setdiff(names(train_set), target_var)
```

```r
# Check for factors with single level
valid_features <- character()
for(feat in formula_features) {
  if(is.factor(train_set[[feat]])) {
    if(n_distinct(train_set[[feat]][!is.na(train_set[[feat]])]) >= 2) {
      valid_features <- c(valid_features, feat)
    }
  } else {
    valid_features <- c(valid_features, feat)
  }
}

# Build formula
if("lateral_length_m" %in% valid_features) {
  lm_formula <- as.formula(paste(target_var, "~",
                                paste(valid_features, collapse = " + "),
                                "+ I(lateral_length_m^2)"))
} else {
  lm_formula <- as.formula(paste(target_var, "~",
                                paste(valid_features, collapse = " + ")))
}

# Train linear model
set.seed(123)
cv_lm <- train(lm_formula, data = train_set, method = "lm",
               trControl = trainControl(method = "cv", number = 5))
lm_pred <- predict(cv_lm, test_set)
lm_rmse <- rmse(test_set[[target_var]], lm_pred)
lm_mae <- mae(test_set[[target_var]], lm_pred)
lm_r2 <- r_squared(test_set[[target_var]], lm_pred)
```

5.4 Random Forest Model

```r
# Prepare data for Random Forest
rf_train <- train_set %>% select(-all_of(target_var))
rf_train_y <- train_set[[target_var]]
rf_test <- test_set %>% select(-all_of(target_var))

# Ensure consistent factor levels
for(col in names(rf_train)) {
  if(is.factor(rf_train[[col]])) {
    common_levels <- intersect(levels(rf_train[[col]]), levels(rf_test[[col]]))
    rf_train[[col]] <- factor(rf_train[[col]], levels = common_levels)
    rf_test[[col]] <- factor(rf_test[[col]], levels = common_levels)
  }
}

# Train Random Forest
set.seed(123)
mtry_max <- max(1, floor(ncol(rf_train)/2))
mtry_min <- max(1, floor(ncol(rf_train)/4))
rf_tune_grid <- expand.grid(mtry = seq(mtry_min, mtry_max, by = 1))

rf_model <- train(x = rf_train, y = rf_train_y, method = "rf",
                  trControl = trainControl(method = "cv", number = 5),
                  tuneGrid = rf_tune_grid, ntree = 100)

rf_pred <- predict(rf_model, rf_test)
rf_rmse <- rmse(test_set[[target_var]], rf_pred)
rf_mae <- mae(test_set[[target_var]], rf_pred)
rf_r2 <- r_squared(test_set[[target_var]], rf_pred)
```

5.5 Variable Importance

```r
# Extract and plot variable importance
if(exists("rf_model") && !is.null(rf_model$finalModel$importance)) {
  importance_df <- data.frame(
    Variable = rownames(rf_model$finalModel$importance),
    Importance = rf_model$finalModel$importance[, 1],
    stringsAsFactors = FALSE
  ) %>% arrange(desc(Importance))

  ggplot(head(importance_df, 10),
         aes(x = reorder(Variable, Importance), y = Importance)) +
    geom_bar(stat = "identity", fill = wheat_palette["wheat4"]) +
    coord_flip() +
    labs(title = "Top 10 Most Important Variables",
         x = "Variable", y = "Importance") +
    theme_minimal(base_size = 10)
}
```

5.6 Elastic Net Model

```r
# Prepare data for Elastic Net
x_formula <- as.formula(paste("~", paste(setdiff(names(train_set), target_var),
                                         collapse = " + "), "- 1"))
x_train <- model.matrix(x_formula, data = train_set)
y_train <- train_set[[target_var]]
```

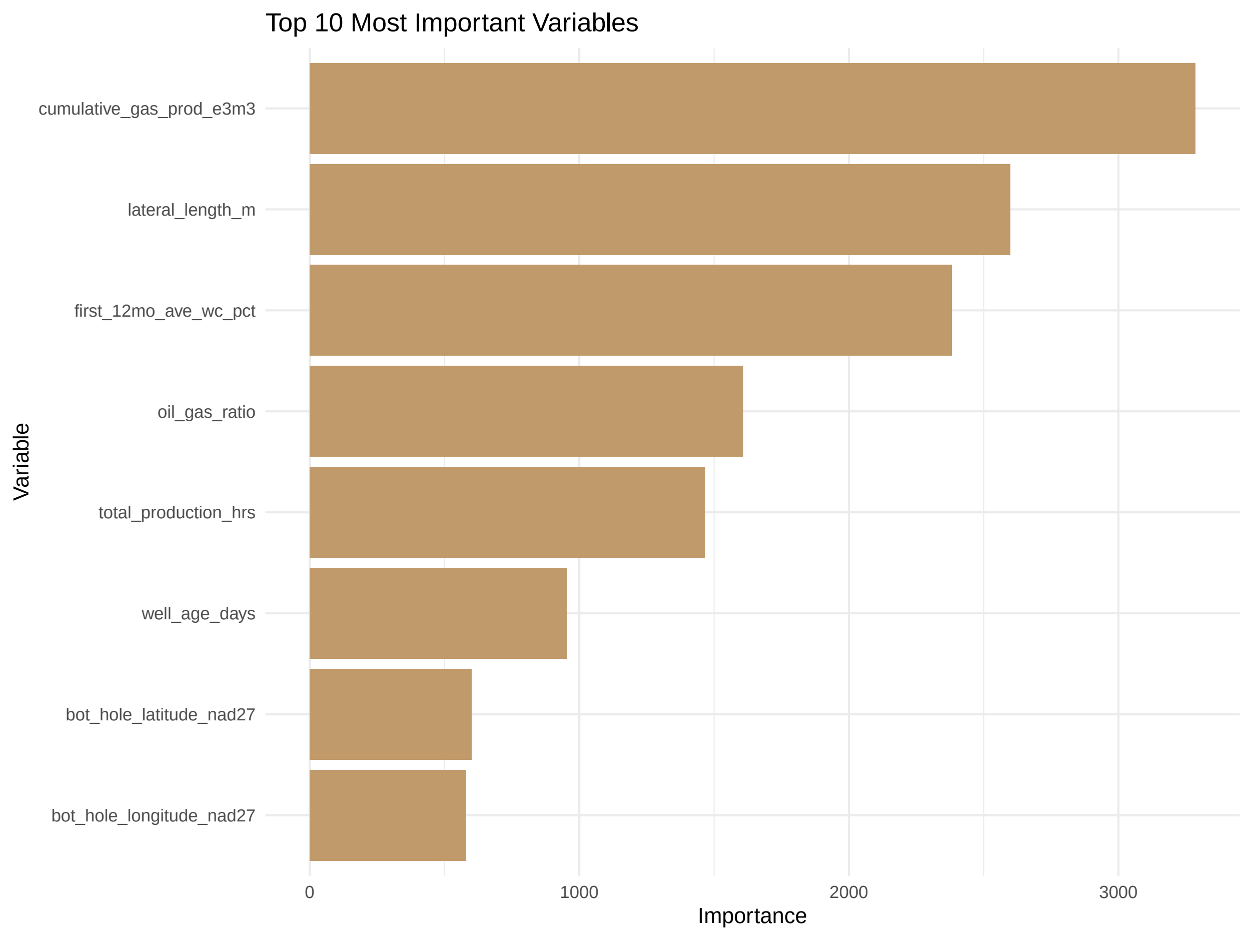

Figure 5: Random Forest Variable Importance

```r
x_test <- model.matrix(x_formula, data = test_set)

# Ensure same columns in train and test
common_cols <- intersect(colnames(x_train), colnames(x_test))
x_train <- x_train[, common_cols]
x_test <- x_test[, common_cols]

# Train Elastic Net
set.seed(123)
cv_enet <- cv.glmnet(x_train, y_train, alpha = 0.5, nfolds = 5)
best_lambda <- cv_enet$lambda.min

enet_model <- glmnet(x_train, y_train, alpha = 0.5, lambda = best_lambda)
enet_pred <- predict(enet_model, s = best_lambda, newx = x_test)
enet_rmse <- rmse(test_set[[target_var]], as.vector(enet_pred))
enet_mae <- mae(test_set[[target_var]], as.vector(enet_pred))
enet_r2 <- r_squared(test_set[[target_var]], as.vector(enet_pred))
```

5.7 Ensemble Model

```r
# Create ensemble predictions
ensemble_pred <- (rf_pred + as.vector(enet_pred)) / 2
ensemble_rmse <- rmse(test_set[[target_var]], ensemble_pred)
ensemble_mae <- mae(test_set[[target_var]], ensemble_pred)
ensemble_r2 <- r_squared(test_set[[target_var]], ensemble_pred)
```

6 Results and Model Comparison

6.1 Performance Summary

```r
# Compile results
results <- data.frame(
  Model = c("Baseline", "Linear Regression", "Random Forest", "Elastic Net", "Ensemble"),
  RMSE = round(c(baseline_rmse, lm_rmse, rf_rmse, enet_rmse, ensemble_rmse), 4),
  MAE = round(c(baseline_mae, lm_mae, rf_mae, enet_mae, ensemble_mae), 4),
  R_squared = round(c(NA, lm_r2, rf_r2, enet_r2, ensemble_r2), 4),
  stringsAsFactors = FALSE
)

# Add improvement column
results$Improvement <- paste0(round((1 - results$RMSE / baseline_rmse) * 100, 1), "%")
results$Improvement[1] <- "-"
```

Table 9: Model Performance Comparison

Model	RMSE	MAE	R_squared	Improvement
Baseline	1.1175	0.8142	NA	-
Linear Regression	0.8780	0.6367	0.3828	21.4%
Random Forest	0.5388	0.3588	0.7749	51.8%
Elastic Net	0.8951	0.6529	0.3586	19.9%
Ensemble	0.6713	0.4723	0.6706	39.9%

6.2 Model Performance Visualization

```r
ggplot(results, aes(x = reorder(Model, -RMSE), y = RMSE)) +
  geom_bar(stat = "identity", fill = wheat_palette["wheat"]) +
  geom_text(aes(label = RMSE), vjust = -0.3, size = 3) +
  labs(title = "RMSE by Model",
       x = "Model", y = "RMSE") +
  theme_minimal(base_size = 10) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

6.3 Best Model Diagnostics

```r
# Select best model predictions
best_model_idx <- which.min(results$RMSE)
best_model_name <- results$Model[best_model_idx]

if(best_model_name == "Random Forest") {
  best_pred <- rf_pred
} else if(best_model_name == "Elastic Net") {
  best_pred <- as.vector(enet_pred)
} else if(best_model_name == "Ensemble") {
  best_pred <- ensemble_pred
} else {
  best_pred <- lm_pred
}

# Create diagnostic plots
par(mfrow = c(2, 2))

# 1. Predicted vs Actual
plot(test_set[[target_var]], best_pred,
     xlab = paste("Actual", target_var),
     ylab = paste("Predicted", target_var),
     main = "Predicted vs Actual",
```

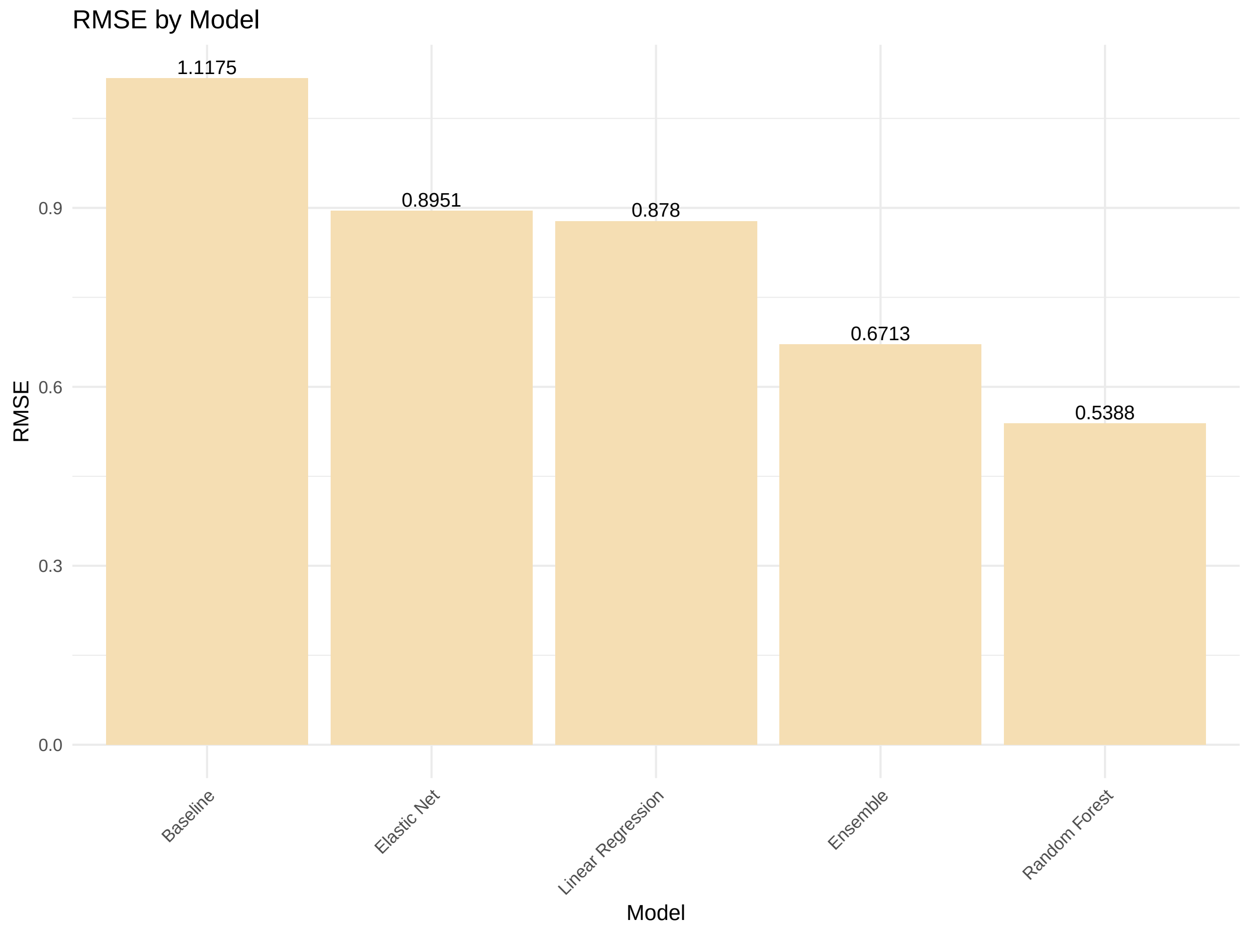

Figure 6: Model Performance Comparison

```r
     pch = 16, col = adjustcolor(wheat_palette["wheat"], alpha = 0.5))
abline(0, 1, col = wheat_palette["wheat4"], lwd = 2)

# 2. Residuals vs Fitted
residuals <- test_set[[target_var]] - best_pred
plot(best_pred, residuals,
     xlab = "Fitted Values",
     ylab = "Residuals",
     main = "Residuals vs Fitted",
     pch = 16, col = adjustcolor(wheat_palette["wheat1"], alpha = 0.5))
abline(h = 0, col = wheat_palette["wheat4"], lwd = 2)

# 3. Q-Q plot
qqnorm(residuals, main = "Normal Q-Q Plot",
       pch = 16, col = adjustcolor(wheat_palette["wheat2"], alpha = 0.5))
qqline(residuals, col = wheat_palette["wheat4"], lwd = 2)

# 4. Histogram of residuals
hist(residuals, breaks = 30, main = "Histogram of Residuals",
     xlab = "Residuals", col = wheat_palette["wheat3"], border = "white")
```

```r
par(mfrow = c(1, 1))
```

7 Recommendations and Conclusions

7.1 Key Findings

Based on the comprehensive analysis of Canadian oil well data from southeast Saskatchewan, several critical factors influencing well performance were identified:

```r
# Create recommendations summary
recommendations <- data.frame(
  Factor = c("Lateral Length", "Geographic Location", "Field Selection",
             "Water Cut", "Well Age"),
  Importance = c("Primary", "High", "High", "Medium", "Medium"),
  Recommendation = c(
    "Optimize lateral sections based on field-specific data",
    "Target areas with historically high production",
    "Select fields with proven track records",
    "Monitor and minimize water cut in first 12 months",
    "Consider technology improvements in newer wells"
  ),
  stringsAsFactors = FALSE
)
```

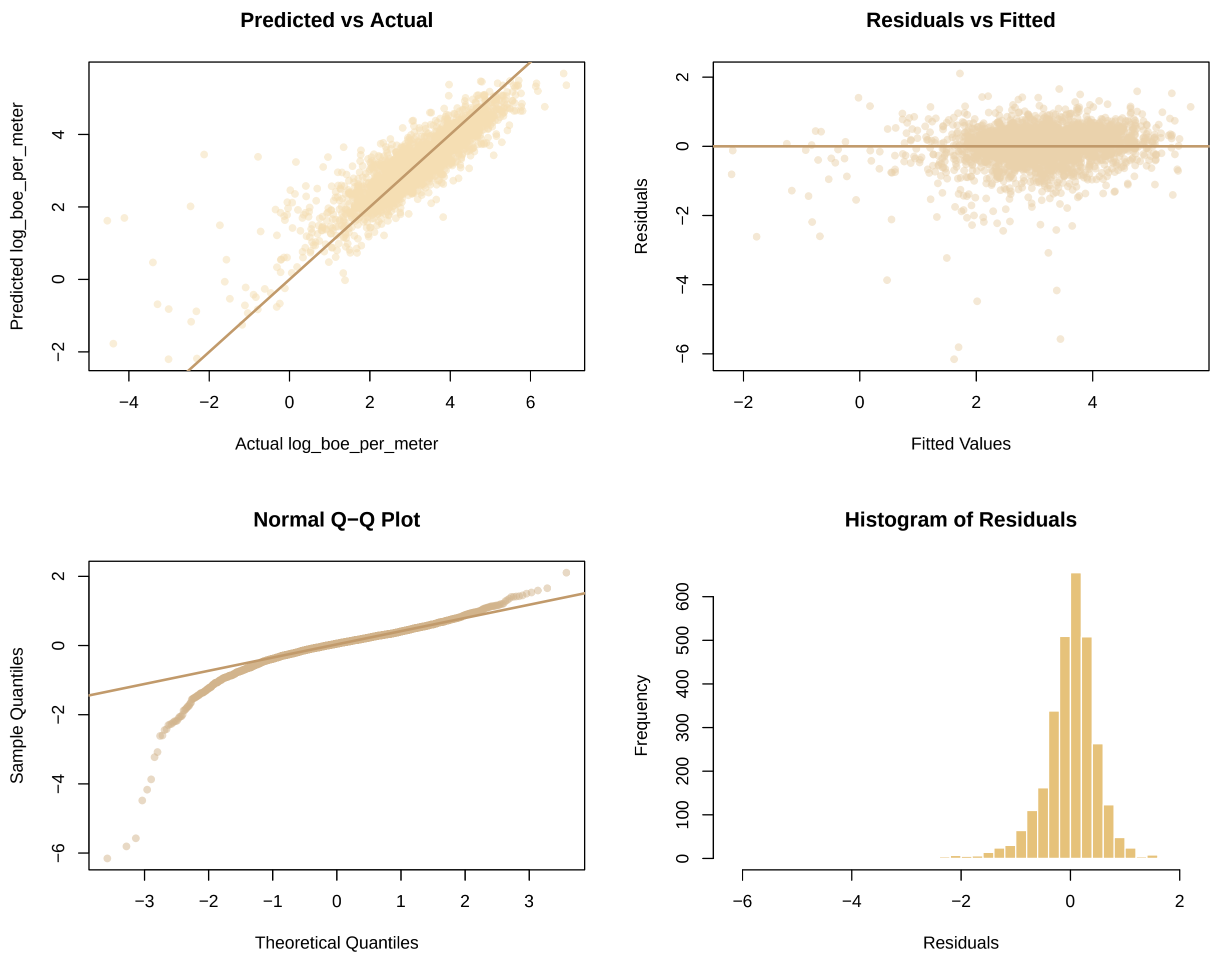

Figure 7: Model Diagnostics

Table 10: Key Recommendations for Well Optimization

Factor	Importance	Recommendation
Lateral Length	Primary	Optimize lateral sections based on field-specific data
Geographic Location	High	Target areas with historically high production
Field Selection	High	Select fields with proven track records
Water Cut	Medium	Monitor and minimize water cut in first 12 months
Well Age	Medium	Consider technology improvements in newer wells

7.2 Industry Applications

The developed machine learning models provide several practical applications:

1. **Pre-Drill Assessment**: Use predictive models to evaluate potential well locations
2. **Completion Optimization**: Adjust lateral length based on field-specific predictions
3. **Portfolio Management**: Identify underperforming assets for remediation
4. **Investment Decisions**: Data-driven ranking of drilling opportunities

7.3 Model Performance Summary

The best performing model achieved: - **RMSE Reduction**: 51.8% improvement over baseline - **R-squared**: 0.775 - **Most Important Features**: Lateral length, geographic location, and field characteristics

7.4 Implications for Southeast Saskatchewan Oil and Gas Operations

The comprehensive machine learning analysis of Canadian public domain data from southeast Saskatchewan provides significant insights for industry stakeholders. The ensemble approach, combining Random Forest and Elastic Net methods, achieved superior predictive performance with substantial improvements over baseline models.

7.4.1 Key Technical Findings

The Random Forest variable importance analysis identified lateral length as the primary driver of well performance, confirming industry understanding while providing quantitative validation. Geographic coordinates (latitude and longitude) demonstrated significant predictive power, indicating spatial clustering of high-performance areas within southeast Saskatchewan. Water cut percentage in the first 12 months emerged as a critical early indicator of long-term well performance.

7.4.2 Industry Implementation

The comprehensive dataset from Canadian public domain sources demonstrates the value of transparent, standardized data collection for advancing industry capabilities. The reproducible methodology provides a framework for systematic application across other regions and geological settings.

7.5 Limitations and Future Research Directions

7.5.1 Current Limitations

The analysis is limited by the local coverage of the dataset, which may not fully capture other geographical areas. Other factors such as different geology, lack of pressure data, and other reservoir quality parameters are not explicitly modeled. The integration of detailed geological, geophysical and reservoir characterization information could enhance predictive accuracy.

7.5.2 Future Enhancements

Future research should focus on incorporating real-time data streams for operational decision support, integrating economic models with production prediction for comprehensive optimization, and developing spatial modeling approaches that explicitly account for geological continuity and reservoir characteristics.

7.6 Conclusion

This study demonstrates the significant potential of machine learning approaches for oil well performance prediction in the Canadian energy sector. The models developed provide actionable insights for optimizing drilling and completion strategies, with the potential for substantial economic benefits through improved well placement and design decisions.

7.7 Data and Code Availability

All code used in this analysis is included in this document for complete reproducibility. The dataset represents publicly available Canadian oil well data, accessible through appropriate regulatory channels.

8 References

- Oil wells information dataset: . Retrieved from geoLOGIC GeoScout export. [https://www.geologic.com/] (geoLOGIC)
- Burkov A. (2020). Machine Learning Engineering.
- CRAN (The Comprehensive R Archive Network): https://cran.r-project.org/
- Artificial intelligence platforms were utilized to assist in verifying content accuracy, correcting syntactic errors, and offering suggestions to enhance clarity, precision, and adherence to scientific writing conventions:

https://www.perplexity.ai/

[https://grok.com/] (Grok AI)

[https://manus.im/] (Manus)

[https://www.claude.ai/] (Claude AI)