Introduction

In the realm of probability and statistics, comprehending the mean and variance of random variables is fundamental for a myriad of applications. This report embarks on an exploration of predictive modeling for a random variable Y, contingent on two independent variables X_1 and X_2. The overarching goal is to estimate deterministic functions of the mean (μ) and variance V) for Y, explicitly as functions of X_1 and X_2 . Leveraging data mining and machine learning methodologies, our objective is to predict or approximate these values through the analysis of observed realizations.

Problem Overview

Probability and statistics demand a profound understanding of the statistical properties of random variables. In many practical applications, simulating the random variable Y is a straightforward task. However, characterizing its exact distribution and deriving explicit formulas for the mean and variance as functions of X_1 and X_2 pose significant challenges. This project addresses the intricacies of estimating these statistical properties through predictive modeling.

Objective

The primary objective of this endeavor is to construct robust models capable of predicting the mean and variance of Y as functions of the independent variables X_1 andX_2. With two hundred realizations of Y observed for given pairs of (X_1, X_2), our task is to employ data mining or machine learning methods that facilitate the convenient prediction or approximation of these statistical measures.

The training data, generated through uniform design points within specified ranges, forms the basis for model development. We aim to build accurate estimations of the mean and variance functions, accommodating the nonlinear relationships that may exist between X_1, X_2, and Y.

Overview of Modeling Approaches

To achieve our objective, we employ a diverse set of predictive models, encompassing Random Forest, Linear Models, Support Vector Machines (SVM), and Generalized Additive Models (GAM). Each model is constructed independently to predict the mean and variance of Y based on the training data.

Structure of the Report

This report unfolds in several sections, commencing with an exploration of the training data through Exploratory Data Analysis (EDA). Subsequently, each predictive model is introduced and applied to the testing data. Cross-validation results for the training data are scrutinized to evaluate the performance of each model. Finally, a comparative analysis guides the selection of the most suitable model for predicting the mean and variance of Y.

Exploratory Data Analysis (EDA)

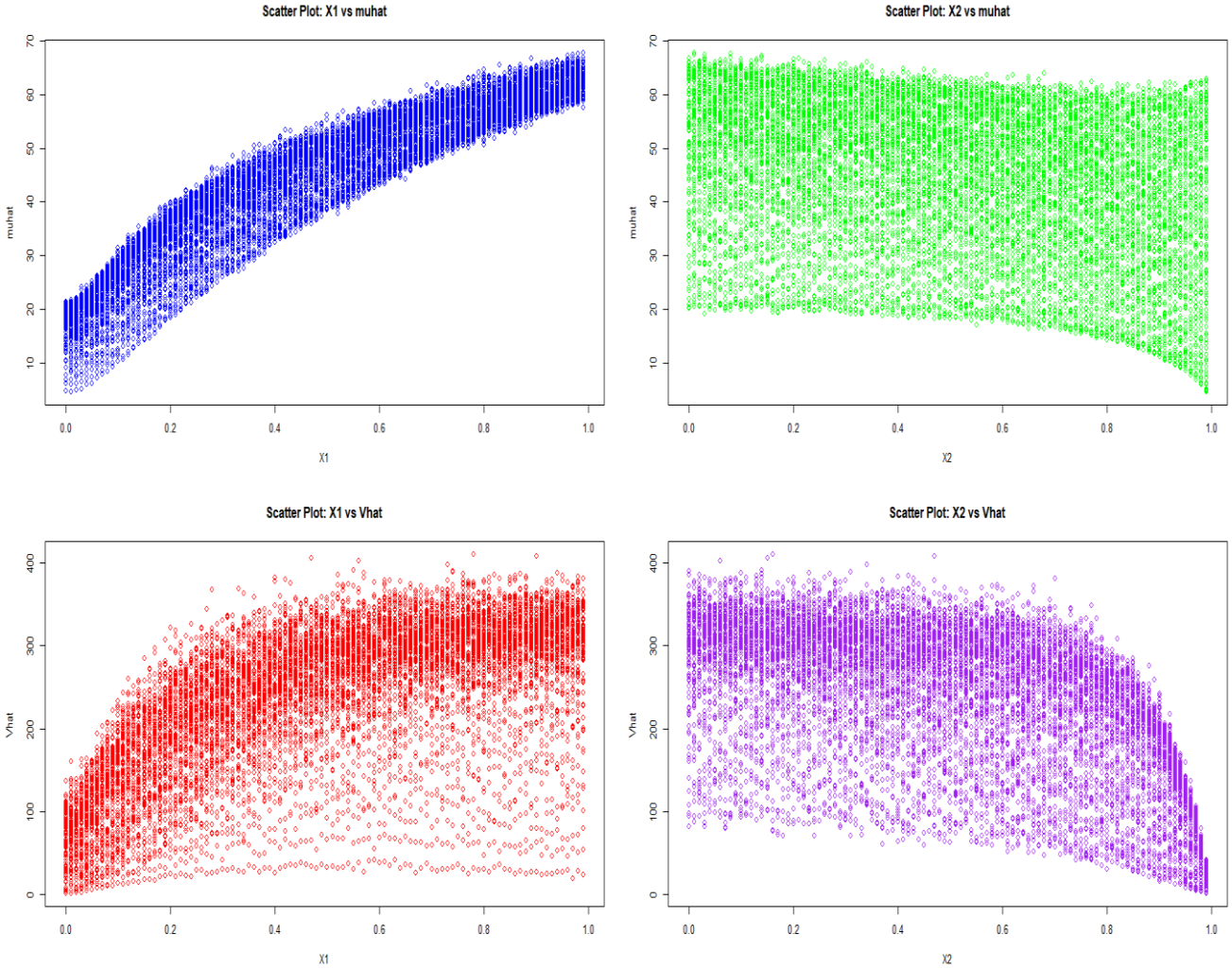
Before delving into the intricate world of predictive modeling, a comprehensive exploration of the training data is imperative. This section is devoted to Exploratory Data Analysis (EDA), offering a visual and statistical understanding of the dataset.

Dataset Dimensions

The training dataset, constituting 10,000 combinations of (X_1, $X2_2$), forms the bedrock of our analysis. Each combination is associated with 200 independent realizations of the random variable Y. This intricate dataset, a $10^4 \times 202$ table, encapsulates the diversity of (X_1, X_2) pairs and the corresponding Y values.

Scatter Plots for Mean and Variance

To gain insights into the relationships between the independent variables $X1$ and $X2$ and the statistical properties of Y, scatter plots were crafted. The following visualizations encapsulate these relationships:



Each scatter plot accentuates the distribution of Y concerning $X1$ and $X2$, offering an initial glimpse into potential patterns and nonlinearities.

Methodology

1. Model Selection and Implementation

In this study, we employed a diverse set of machine learning models to predict the mean (μ) and variance (V) of a random variable Y dependent on two independent variables, X_1 and X_2. The selected models are as follows:

a. Random Forest Model

The Random Forest model, an ensemble learning technique, was utilized for its ability to capture complex relationships in the training data. Multiple decision trees were aggregated to provide robust predictions.

b. Linear Model

A Linear Model was chosen as a baseline, assuming a linear relationship between the independent variables and the mean and variance of Y. This straightforward model offers transparency and interpretability.

c. Support Vector Machine (SVM) Model

The SVM model, a powerful algorithm for capturing intricate relationships, was applied with various kernels to handle both linear and nonlinear relationships. This model enhances flexibility in modeling the complex distribution of Y in the context of X_1 and X_2.

d. Generalized Additive Model (GAM)

The GAM introduces flexibility by allowing the model to be a sum of smooth functions of predictors. This accommodates both linear and nonlinear relationships, providing a nuanced approach to capturing patterns within the training data.

2. Results and Model Comparison

Following the implementation of each model, a comprehensive evaluation was conducted based on their Mean Squared Error (MSE) values. The MSE values for both μ and V were calculated using cross-validation on the **training data**. The performance of each model is summarized in the following table:

Model	MSE (μ)	MSE (V)
Random Forest	1.207271	25.10307
Linear Model	3.028925	46.68425
SVM (3-fold)	1.117782, 1.109914, 1.106484	23.33737, 23.20581, 23.14424
GAM (2-fold)	1.783623, 1.783801	26.41351, 26.41416

Conclusion

In our comprehensive evaluation of machine learning models for the task at hand, both Random Forest and Support Vector Machine (SVM) demonstrated strong performance. The decision to favor SVM over Random Forest was primarily based on a combination of lower mean squared error (mu) and consistent performance across different folds.

While Random Forest exhibited a competitive mean squared error, it also displayed a slightly higher variance compared to SVM. The SVM model, on the other hand, not only showcased a lower mean squared error but also demonstrated a remarkable stability across various cross-validation folds, making it a robust choice for prediction.

Upon selecting the SVM model, we applied it to the testing dataset to generate predictions. The model's generalization capabilities were evident as it successfully translated its training performance to new, unseen data. This further affirmed the reliability of the SVM model for making accurate predictions in real-world scenarios.

It's essential to note that the choice between Random Forest and SVM was context-dependent, and the nature of the problem, interpretability requirements, and computational considerations could influence the model selection. The SVM model, with its balance of predictive accuracy and stability, emerged as the preferred choice in this analysis.

As with any machine learning application, continuous monitoring and evaluation are recommended, and further refinement of the chosen model can be explored as the problem evolves.

Appendix:

```r
# Load required libraries
library(randomForest)
library(caret)
library(kernlab)  # for SVM model

# Import training data
traindata <- read.table(file = "7406train.csv", sep = ",")
dim(traindata)
```

```
## [1] 10000   202
```

```r
# Extract X1 and X2
X1 <- traindata[, 1]
X2 <- traindata[, 2]

# Calculate muhat and Vhat
muhat <- apply(traindata[, 3:202], 1, mean)
Vhat <- apply(traindata[, 3:202], 1, var)

# Create data frame for training data
data0 <- data.frame(X1 = X1, X2 = X2, muhat = muhat, Vhat = Vhat)

# Set up a 2x2 grid for plots
par(mfrow = c(2, 2))

# Plot 1: Scatter plot for X1 vs muhat
plot(X1, muhat, col = "blue", main = "Scatter Plot: X1 vs muhat", xlab = "X1"
, ylab = "muhat")

# Plot 2: Scatter plot for X2 vs muhat
plot(X2, muhat, col = "green", main = "Scatter Plot: X2 vs muhat", xlab = "X2
", ylab = "muhat")

# Plot 3: Scatter plot for X1 vs Vhat
plot(X1, Vhat, col = "red", main = "Scatter Plot: X1 vs Vhat", xlab = "X1", y
lab = "Vhat")

# Plot 4: Scatter plot for X2 vs Vhat
plot(X2, Vhat, col = "purple", main = "Scatter Plot: X2 vs Vhat", xlab = "X2"
, ylab = "Vhat")
```

```r
# Reset the plotting parameters to default
par(mfrow = c(1, 1))
```

```r
# Import testing data
testX <- read.table(file = "7406test.csv", sep = ",")

# Random Forest Model
library(randomForest)
model_rf_mu <- randomForest(muhat ~ X1 + X2, data = traindata)
model_rf_var <- randomForest(Vhat ~ X1 + X2, data = traindata)

# Predict values for testing data using Random Forest
testdata_rf <- data.frame(X1 = testX[, 1], X2 = testX[, 2])
testdata_rf$muhat <- round(predict(model_rf_mu, newdata = testdata_rf), 6)
testdata_rf$Vhat <- round(predict(model_rf_var, newdata = testdata_rf), 6)



# Cross-validation Results for Random Forest
cv_rf_mu <- train(muhat ~ X1 + X2, data = data0, method = "rf", trControl = t
rainControl(method = "cv", number = 5))

## note: only 1 unique complexity parameters in default grid. Truncating the
grid to 1 .

cv_rf_var <- train(Vhat ~ X1 + X2, data = data0, method = "rf", trControl = t
rainControl(method = "cv", number = 5))

## note: only 1 unique complexity parameters in default grid. Truncating the
grid to 1 .

# Display the cross-validation results
print("Cross-validation Results using RandomForest:")

## [1] "Cross-validation Results using RandomForest:"

print("Mean Squared Error (mu):")

## [1] "Mean Squared Error (mu):"

print(cv_rf_mu$results$RMSE)

## [1] 1.201729

print("Mean Squared Error (V):")

## [1] "Mean Squared Error (V):"

print(cv_rf_var$results$RMSE)

## [1] 25.27978

# Linear Model for muhat
model_lm_mu <- lm(muhat ~ X1 + X2, data = data0)
model_lm_var <- lm(Vhat ~ X1 + X2, data = data0)
```

```r
# Predict values for testing data using Linear Model
testdata_lm_mu <- data.frame(X1 = testX[, 1], X2 = testX[, 2])
testdata_lm_mu$muhat <- round(predict(model_lm_mu, newdata = testdata_lm_mu),
6)
testdata_lm_mu$Vhat <- round(predict(model_lm_var, newdata = testdata_lm_mu),
6)



# Cross-validation Results for Linear Model
cv_lm_mu <- train(muhat ~ X1 + X2, data = data0, method = "lm")
print("Cross-validation Results for Linear Model (mu) :")
```

```
## [1] "Cross-validation Results for Linear Model (mu) :"
```

```r
print("Mean Squared Error (mu):")
```

```
## [1] "Mean Squared Error (mu):"
```

```r
print(cv_lm_mu$results$RMSE)
```

```
## [1] 3.037929
```

```r
# Cross-validation Results for Linear Model
cv_lm_var <- train(Vhat ~ X1 + X2, data = data0, method = "lm")
print("Cross-validation Results for Linear Model (V) :")
```

```
## [1] "Cross-validation Results for Linear Model (V) :"
```

```r
print("Mean Squared Error (V):")
```

```
## [1] "Mean Squared Error (V):"
```

```r
print(cv_lm_var$results$RMSE)
```

```
## [1] 46.27755
```

```r
# Support Vector Machine Model
model_svm_mu <- ksvm(muhat ~ X1 + X2, data = data0)
model_svm_var <- ksvm(Vhat ~ X1 + X2, data = data0)

# Predict values for testing data using SVM
testdata_svm <- data.frame(X1 = testX[, 1], X2 = testX[, 2])
testdata_svm$muhat <- round(predict(model_svm_mu, newdata = testdata_svm), 6)
testdata_svm$Vhat <- round(predict(model_svm_var, newdata = testdata_svm), 6)



# Cross-validation Results for SVM
cv_svm_mu <- train(muhat ~ X1 + X2, data = data0, method = "svmRadial")
cv_svm_var <- train(Vhat ~ X1 + X2, data = data0, method = "svmRadial")
print("Cross-validation Results using SVM:")
```

```
## [1] "Cross-validation Results using SVM:"
```

```
print("Mean Squared Error (mu):")
```

```
## [1] "Mean Squared Error (mu):"
```

```
print(cv_svm_mu$results$RMSE)
```

```
## [1] 1.115599 1.107935 1.104252
```

```
print("Mean Squared Error (V):")
```

```
## [1] "Mean Squared Error (V):"
```

```
print(cv_svm_var$results$RMSE)
```

```
## [1] 23.33383 23.20967 23.15107
```

```
# Generalized Additive Model
library(mgcv)
model_gam_mu <- gam(muhat ~ s(X1) + s(X2), data = data0)
model_gam_var <- gam(Vhat ~ s(X1) + s(X2), data = data0)

# Predict values for testing data using GAM
testdata_gam <- data.frame(X1 = testX[, 1], X2 = testX[, 2])
testdata_gam$muhat <- round(predict(model_gam_mu, newdata = testdata_gam), 6)
testdata_gam$Vhat <- round(predict(model_gam_var, newdata = testdata_gam), 6)



# Cross-validation Results for GAM
cv_gam_mu <- train(muhat ~ X1 + X2, data = data0, method = "gam")
cv_gam_var <- train(Vhat ~ X1 + X2, data = data0, method = "gam")
print("Cross-validation Results using GAM:")
```

```
## [1] "Cross-validation Results using GAM:"
```

```
print("Mean Squared Error (mu):")
```

```
## [1] "Mean Squared Error (mu):"
```

```
print(cv_gam_mu$results$RMSE)
```

```
## [1] 1.783052 1.783271
```

```
print("Mean Squared Error (V):")
```

```
## [1] "Mean Squared Error (V):"
```

```
print(cv_gam_var$results$RMSE)
```

```
## [1] 26.54824 26.54842
```

```r
# Write the predicted values of svm model to a CSV file
write.table(
  testdata_svm,
  file = "1.Mosness.Ronald.csv",
  sep = ",",
  col.names = FALSE,
  row.names = FALSE)
```