Data Mining and Statistical Learning

- Final Exam -

Introduction

In probability and statistics, it is important to understand the mean and variance for any random variables. In many applications, it is straightforward to simulate the random variable Y, but it is often highly non-trivial to characterize the exact distribution of $Y = f(X_1, X_2)$ including deriving the explicit formulas for the mean and variance of $Y = f(X_1, X_2)$ as a function of X_1 and X_2.

Nonparametric regression estimators (also known as **"smoothers"**) can be used to estimate the unknown functions $\mu(X_1, X_2)$ and $\text{Var}(X_1, X_2)$ from a sample of noisy data over certain domain (R^P) by estimating what $\mu(X_1, X_2)$ and $\text{Var}(X_1, X_2)$ are at a point X_0. This can be accomplished by using only those observations close to X_0 to fit a simple model and the resulting estimated function $\widehat{\mu}(X_1, X_2)$ and $\widehat{\text{Var}}(X_1, X_2)$ is smooth in R^p.

There are several local smoothing methods like **LOESS**, **Nadaraya-Watson**, and **Spline** and the goal of this paper is to use one these methods to estimate the mean and variance of the random variable Y as a function of (X_1, X_2)

Problem Statement and Dataset

In this exam, we are using $Y = Y(X_1, X_2)$, a random variable whose distribution depends on two independent variables X_1 and X_2, and the objective is to estimate two deterministic functions of X_1 and X_2: one is the mean $\mu(X_1, X_2) = E(Y)$ and the other is the variance $\text{Var}(X_1, X_2) = \text{Var}(Y)$.

For that purpose, we are provided the observed 200 realizations of the Y values for some given pairs (X_1, X_2). We are asked to use data mining or machine learning methods that allow us to conveniently predict or approximate the mean and variance of $Y = Y(X_1, X_2)$ as a function of (X_1, X_2).

Our objective is estimate the mean and the variance of the given pairs (X_1, X_2) in the testing data set.

- $\mu(X_1, X_2) = E(Y(X_1, X_2))$
- $\text{Var}(X_1, X_2) = \text{Var}(Y(X_1, X_2))$.

In order to develop a reasonable estimation of the mean and variance of $Y = Y(X_1, X_2)$ as deterministic functions of X_1 and X_2, we were provided a training data set that was generated as follows. We first choose the uniform design points when $0 \leq X_1 \leq 1$ and $0 \leq X_2 \leq 1.$, that is, $x_{1i} = 0.01i$ for $i = 0, 1, 2, ..., 99$, and $x_{2j} = 0.01j$ for $j = 0, 1, 2, ..., 99$ Thus there are a total of $100 \times 100 = 10^4$ combinations of (x_{1i}, x_{2j})'s, and for each of these 10^4 combinations, we generate **200** independent realizations of the Y variables, denoted by Y_{ijk} for $k = 1, ..., 200$. The corresponding training data is then stored in **2022Fall7406train.csv**.

Note that this training data set is a $10^4 \times 202$ table. Each row corresponds to one of $100 \times 100 = 10^4$ combinations of (X_1, X_2)'s. The first and second columns are the X_1 and X_2 values, respectively, whereas the remaining **200** columns are the corresponding **200** independent realizations of Y 's. Using the 200 realizations of Y 's, we compute the "true" values of the mean and variance then store them in a dataframe **data0**. The columns of **data0** are X_1, X_2, μ_{true}, and Var_{true}.

Based on the training data (i.e data0), the goal is to develop an accurate estimation of the functions $\mu(X1, X2) = E(Y)$ and $\text{Var}(X_1, X_2) = \text{Var}(Y)$, as deterministic functions of X_1 and X_2 when $0 \leq X_1 \leq 1$ and $0 \leq X_2 \leq 1$. Then use the proposed models to predict these values using testing data. The testing data is stored in **2022Fall7406test.csv**, and it includes $50 \times 50 = 2500$ combination of $(X1, X2)$

Exploratory Data Analysis

For any real dataset, we first need to conduct empirical data analysis to have a better understanding of the data. A useful command in R is the summary function. It shows the minimum, 25th quantile, median, mean, 75th quantile, and maximum values of each variable in the training dataset. The following table reports the summary statistics of the training data (data0)

Table 1: Summary statistics

X1	X2	muhat	Vhat
Min. :0.0000	Min. :0.0000	Min. : 4.75	Min. : 1.723
1st Qu.:0.2475	1st Qu.:0.2475	1st Qu.:34.47	1st Qu.:173.757
Median :0.4950	Median :0.4950	Median :47.03	Median :258.930

X1	X2	muhat	Vhat
Mean :0.4950	Mean :0.4950	Mean :44.41	Mean :235.148
3rd Qu.:0.7425	3rd Qu.:0.7425	3rd Qu.:55.90	3rd Qu.:305.940
Max. :0.9900	Max. :0.9900	Max. :67.63	Max. :409.924

Now, let us look at the distribution of each column in training data **data0** through a histogram plot.

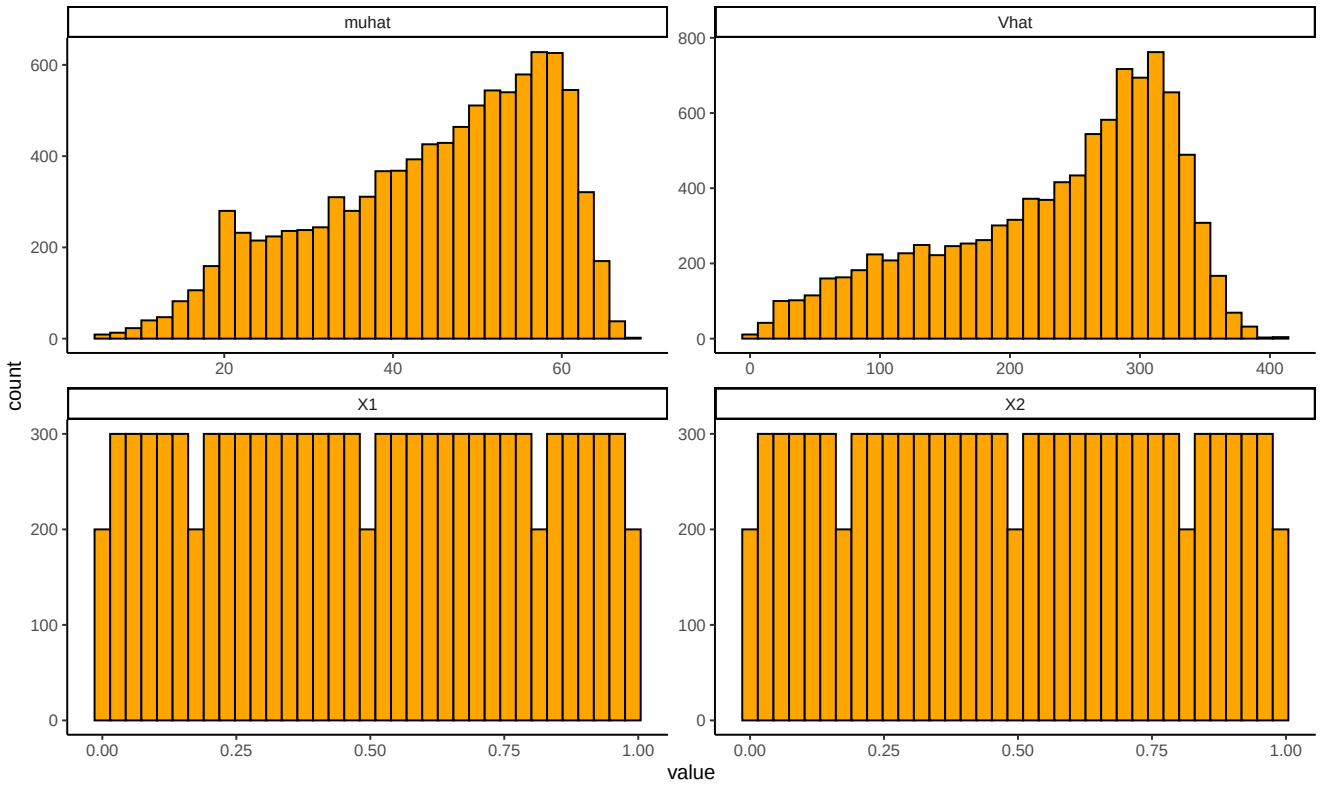

Figure 1: Histogram of each column in the data0 dataset

The histograms are as expected. Additionally, we can see roughly where the peaks of the distribution are, whether the distribution is skewed or symmetric, and if there are any outliers.

The following plots represent the true mean and variance as a function of X_1 and X_2.

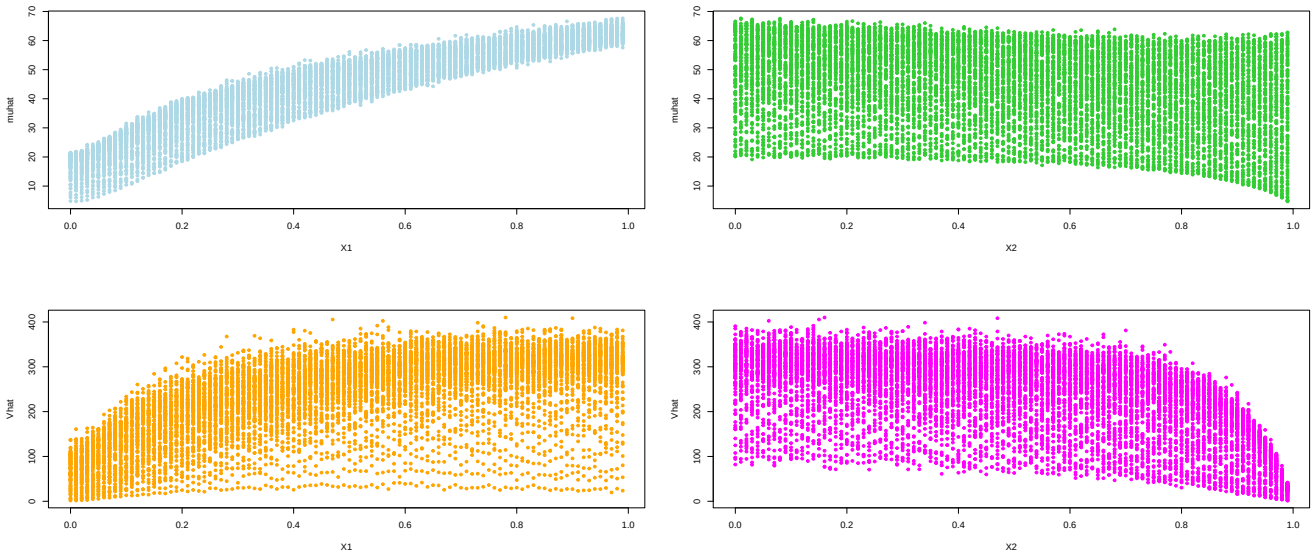

Figure 2: muhat and vhat vs X1 and X2

Proposed Methodology

As previously stated, the goal of this analysis is to estimate the mean and variance of a random variable Y as a function of X_1 and X_2 when $0 \leq X_1 \leq 1$ and $0 \leq X_2 \leq 1$. This analysis is split into several experiments.

In the first experiment, we split data0 into training and testing sets using a 70/30 random split with replacement. Then, we build a local smoothing model using Loess (with span = 0.05 for mean and a span = 0.06 for variance) where model 1 = loess($\mu_{\text{true}} \sim X_1 + X_2$), and model 2 = loess($\text{Var}_{\text{true}} \sim X_1 + X_2$). We train both models on training data, then we test them using the testing data. For each model, we compute the mean square error (MSE).

In the second experiment, we run a $m = 200$ Monte Carlo runs to generate a data set of the form (x_i, Y_i) with $\hat{\mu}_i = \mu_{\text{true}} + \epsilon_i$ and $\widehat{\text{Var}}_i = \text{Var}_{\text{true}} + \epsilon_i$. The added white noise ϵ_i are independent and identically distributed (iid) $\approx N(0, 0.2^2)$. For each Monte Carlo run, we build a local smoothing estimates using Loess (with span = 0.05 for the mean and a span = 0.06 for the variance). For each model, we compute the mean square error (MSE).

In the third experiment, we will tune LOESS smoothing parameter α using a 10-fold cross validation and span values in the range of **seq(0.01, 0.97, 0.01)**. After tuning the smoothing LOESS parameter, we will repeat the Monte Carlo run with the newly tuned parameter, compute MSE on training data then predict on testing data.

Results

Experiment one

Below we plot the true mean and variance values with respect to X_1 and X_2. The overlaid orange and blue points are the estimated mean and variance respectively using **Loess** local smoothing estimator.

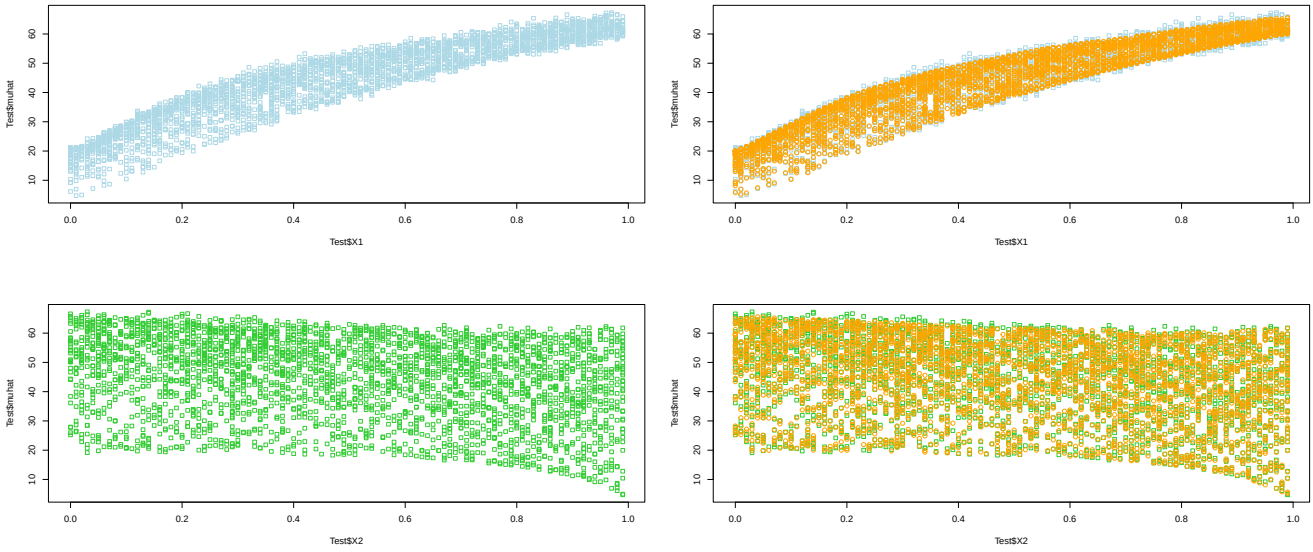

Figure 3: muhat vs the estimated muhat (orange)

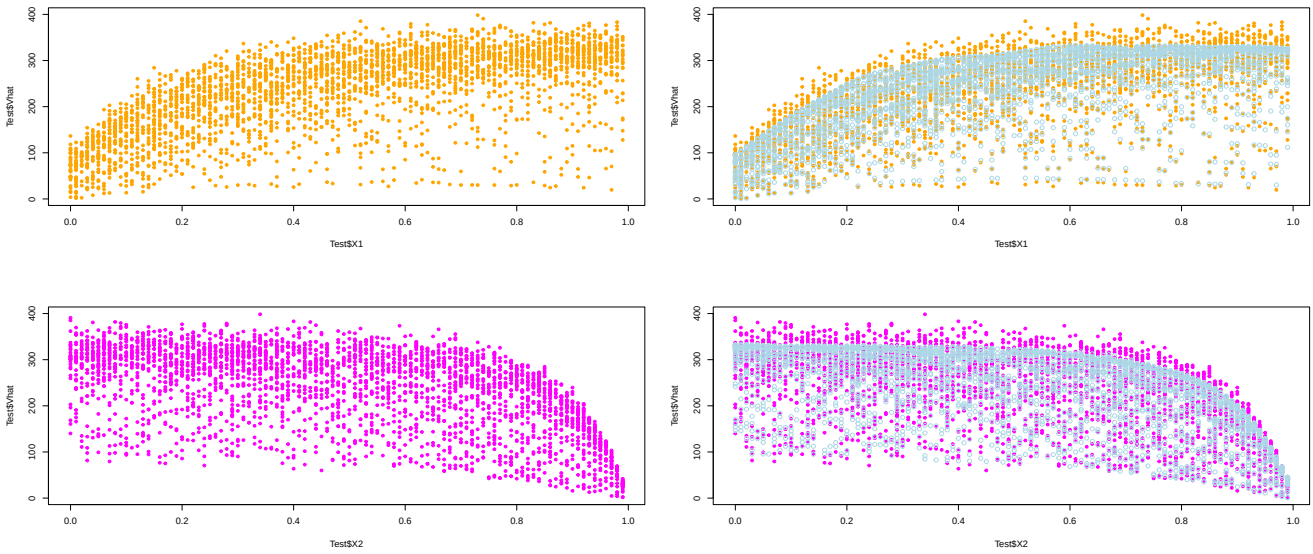

Figure 4: vhat vs estimated vhat (lightblue)

From the plots above, it is obvious that our estimated mean and variance as functions of X_1 and X_2 are close to the true values accurate (especially for the mean).

The initial MSE results can be found below:

Table 2: Non-MC MSE

MSE (mu)	MSE (var)
1.20608	512.87

Experiment Two

MC simulated mean plots versus the mean from the training data. The orange points represent the estimated mean from the simulation

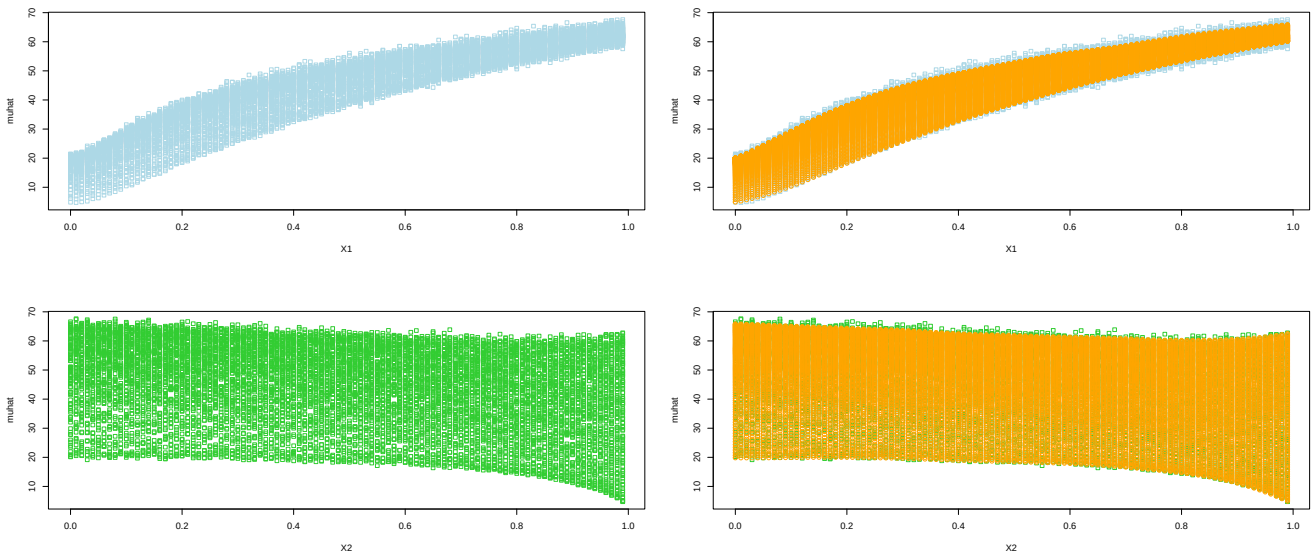

Figure 5: muhat vs average of Monta carto estimated muhat

MC simulated variance plots versus the variance from the training data. The light blue points represent the estimated variance from the simulation.

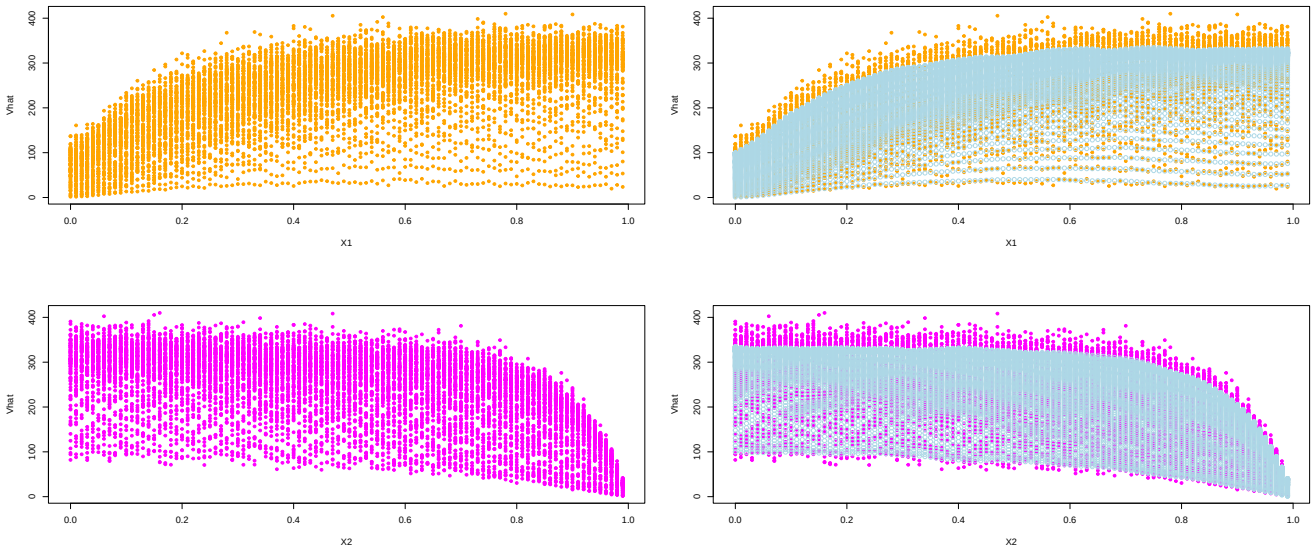

Figure 6: vhat vs average of Monta carto estimated vhat

The MSE results using the Monte Carlo simulated the mean and variance can be found below:

Table 3: Average MC MSE

MC MSE (mu)	MC MSE (var)
1.15881	509.218

The predicted mean and variance (After Monte Carlo runs) as function of X_1 and X_2 using testing data

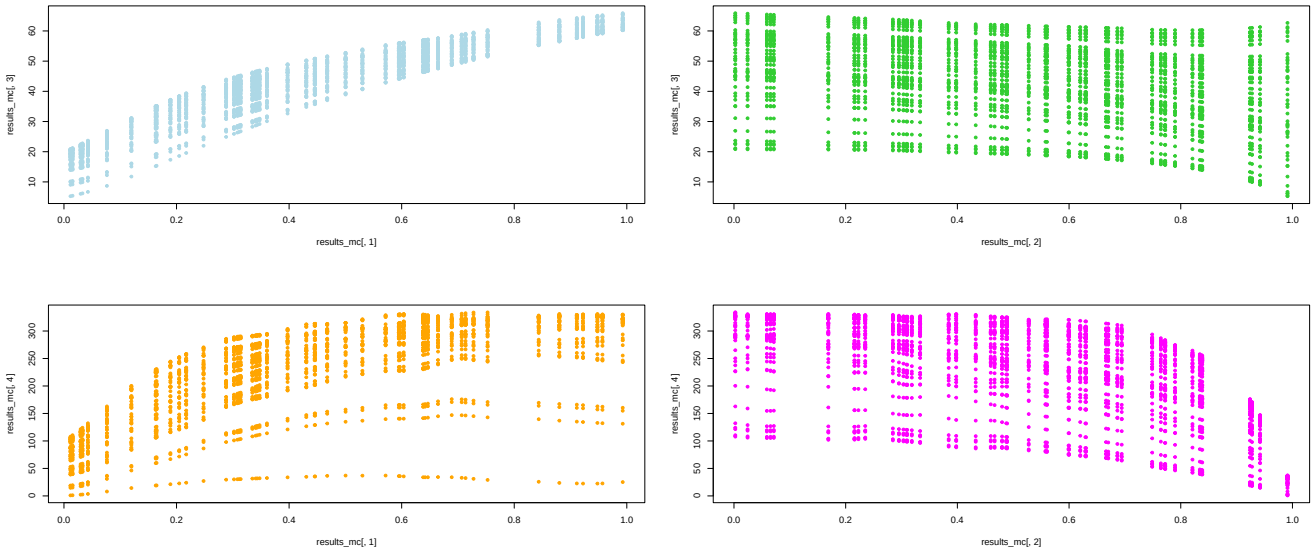

Figure 7: Predicted mean and variance using testing data

Here, we notice that the shapes of our predicted mean and variance are identical to the true mean and variance from the training data.

Experiment Three: 10-fold cross validation

For each μ and Var estimates, we compare the actual observed μ_i and Var_i with its smoothed estimate for a given α based on the $k-1$ data points and testing on the remaining subset of data. We choose the optimal tuning span by minimizing the average mean square error based (MSE) of the 10-fold cross-validation. Then, we repeat the $m = 200$ Monte Carlo run with the newly tuned parameter.

The MSE results using the Monte Carlo simulated the mean and variance after tuning can be found below:

Table 4: Average MC MSE after span tuning (10 fold cv)

10-fold CV MC MSE (mu)	10-fold CV MC MSE (var)
1.15881	509.218

The predicted mean and variance (after cross-validation and Monte Carlo runs) as function of X_1 and X_2 using testing data

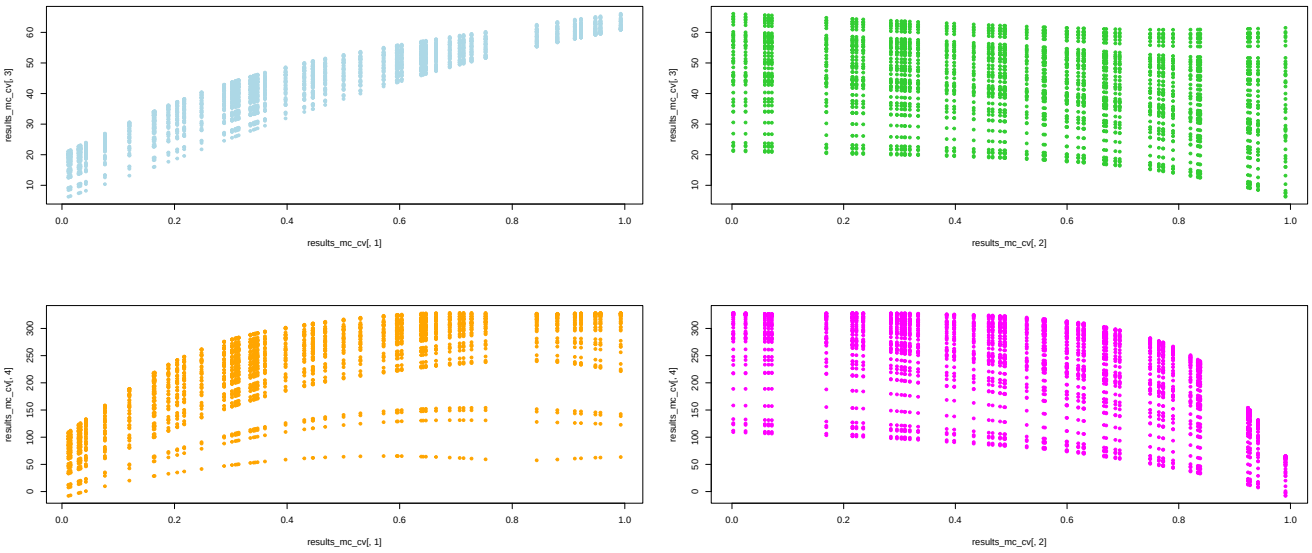

Figure 8: Predicted mean and variance using testing data

Again, the shapes of the predicted mean and variance are identical to the true mean and variance from the training data.

Conclusions and Findings

In this analysis, we used local smoothing methods (LOESS) to estimate the mean and variance of a random variable Y as a function of X_1 and X_2. The MSE results from the second and third experiments where we simulated the mean and variance with added white noise to account for variability in the data were much better compared to experiment one (especially in terms of variance). Additionally, the results from the $m = 200$ Monte Carlo runs before and after tuning the span parameter were identical.

In terms of estimating the mean and variance using testing data, we clearly see from the results of experiments two and three that the resulting shapes follow exactly what we have in the training data. This gives us a lot of confidence in our model's predictability.

References

- HW4 code (own code) and course supplemental code

Appendix

R Code

```
# Settings
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.pos = 'H')


# Check if packages are installed. If not, install them.
install.packages <- function(pkg) {
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg)) {
        install.packages(new.pkg)
    }
}

packages <- c("ggplot2", "mcmcplots", "knitr","xtable", "kableExtra",
              "reshape2", "devtools","MASS","glmnet", "lares", "caret",
              "splines", "class", "dplyr", "PerformanceAnalytics",  "factoextra")
```

```
install.packages(packages)
library(mcmcplots)
library(knitr)
library(kableExtra)
library(ggplot2)
library(reshape2)
library(devtools)
library(lares)
library(caret)
library(splines)
library(class)
library(extrafont)
library(xtable)
library(dplyr)
library(leaps)
library(MASS)
library(lars)
library(pls)
library(glmnet)
library(tidyverse)
library(gtsummary)
library(tidyr)
library(PerformanceAnalytics)
library(corrplot)
library(factoextra)
options(kableExtra.latex.load_packages = FALSE)
loadfonts()


devtools::install_github("haozhu233/kableExtra")


traindata <- read.table(file = "2022Fall7406train.csv", sep=",");
## dim=10000*202
## The first two columns are X1 and X2 values, and the last 200 columns are the Y valus
```

```
X1 <- traindata[,1];
X2 <- traindata[,2];

## note that muhat = E(Y) and Vhat = Var(Y)
muhat <- apply(traindata[,3:202], 1, mean);
Vhat  <- apply(traindata[,3:202], 1, var);

data0 <- data.frame(X1 = X1, X2=X2, muhat = muhat, Vhat = Vhat);

X1.noise = X1 + rnorm(length(data0), sd=0.02)
X2.noise = X2 + rnorm(length(data0), sd=0.02)

data1 <- data.frame(X1 = X1, X2=X2, X1_noise = X1.noise, X2_noise =X2.noise , muhat = muhat, Vhat = Vhat);

write.csv(data1, "trainingData_noise.csv")

# test data
testX  <- read.table(file = "2022Fall7406test.csv", sep=",")

knitr::kable(summary(data0),"pipe", digit=3,
             caption = paste("Summary statistics"))%>%
          kable_styling(position = "center", full_width = F)

ggplot(gather(data0), aes(value)) +
    geom_histogram(bins = 35, fill = 'orange', color='black') +
    facet_wrap(~key, scales = 'free') +
    theme_classic()

## we can plot 4 graphs in a single plot
par(mfrow = c(2, 2));
plot(X1, muhat, col="lightblue",pch=20);
plot(X2, muhat, col="limegreen",pch=20);
plot(X1, Vhat, col="orange",pch=20);
```

```
plot(X2, Vhat, col="magenta",pch=20);


set.seed(1)
# splitting the data
ind <- sample(2, nrow(data0),
              replace = TRUE,
              prob = c(0.70, 0.3))

# Get training data
Train <- data0[ind ==1,]

# get testing data
Test <- data0[ind ==2,]

## Mean estimation
mod1 <- loess(muhat ~ X1 + X2, span = 0.05, data = Train,control=loess.control(surface="direct"))
mu_est_lp <- predict(mod1, newdata=Test[, 1:2])
MSE_mu_lp = mean((mu_est_lp - Test$muhat)^2, na.rm = TRUE)

# Variance estimation
mod2 <- loess(Vhat ~ X1 + X2, span = 0.06, data = Train, control=loess.control(surface="direct"))
var_est_lp <- predict(mod2, newdata=Test[, 1:2])
MSE_var_lp= mean((var_est_lp - Test$Vhat)^2, na.rm = TRUE)

# prediction on testing data (without noise added)
mu_test <- predict(mod1, newdata=data.frame(X1 = testX$V1, X2=testX$V2))
var_test <- predict(mod2, newdata=data.frame(X1 = testX$V1, X2=testX$V2))



results <- testX;
results[,3] <- format(round(mu_test, 6), nsmall = 6) # force value to be x.xxxxxx (6 digits after decimal)
results[,4] <- format(round(var_test, 6), nsmall = 6) # force value to be x.xxxxxx (6 digits after decimal

write.table(results, file="1.Elmali.Siham.orig.csv",
```

```
            sep=",",  col.names=F, row.names=F)


par(mfrow = c(2, 2));
plot(Test$X1, Test$muhat, col="lightblue",pch=22);
plot(Test$X1, Test$muhat, col="lightblue",pch=22);
points(mu_est_lp, x=Test$X1, lwd=1, col="orange")

plot(Test$X2, Test$muhat, col="limegreen",pch=22);
plot(Test$X2, Test$muhat, col="limegreen",pch=22);
points(mu_est_lp, x=Test$X2, lwd=1, col="orange")



par(mfrow = c(2, 2));
plot(Test$X1, Test$Vhat, col="orange",pch=20);
plot(Test$X1, Test$Vhat, col="orange",pch=20);
points(var_est_lp, x=Test$X1, lwd=1, col="lightblue")

plot(Test$X2, Test$Vhat, col="magenta",pch=20);
plot(Test$X2, Test$Vhat, col="magenta",pch=20);
points(var_est_lp, x=Test$X2, lwd=1, col="lightblue")

mse_summary_1 <- cbind(MSE_mu_lp, MSE_var_lp)
colnames(mse_summary_1) <- c( "MSE (mu)","MSE (var)")

# displaying the tables
knitr::kable(mse_summary_1,"pipe", digit=5,
             caption = "Non-MC MSE")%>%
          kable_styling(position = "center", full_width = F)
set.seed(42)
m <- 200
n <- 10000
## Initialize the matrix of fitted values for three methods
muvlp <- varlp <- matrix(0, nrow= n, ncol= m)
mu_mc_test <- var_mc_test <- matrix(0, nrow= 2500, ncol= m)
```

```r
##Generate data, fit the data and store the fitted values
for (j in 1:m){

  ## simulate mu-values
  muhat_noise <- muhat + rnorm(length(data0), sd=0.2)

  mod1 <- loess(muhat_noise ~ X1 + X2, span = 0.05,
                data = data0,control=loess.control(surface="direct"))
  muvlp[, j] <- predict(mod1, newdata=data0[, 1:2])



  ## Simulate vhat values
  vhat_noise <- Vhat +  rnorm(length(data0), sd=0.2)
  mod2 <- loess(vhat_noise ~ X1 + X2, span = 0.03,
                data = data0, control=loess.control(surface="direct"))
  varlp[, j] <- predict(mod2, newdata=data0[, 1:2])

  ## prediction on testing data
  mu_mc_test[, j]  <- predict(mod1, newdata=data.frame(X1 = testX$V1, X2=testX$V2))
  var_mc_test[, j] <- predict(mod2, newdata=data.frame(X1 = testX$V1, X2=testX$V2))

}

## Take the mean
mean_mu_pred = apply(mu_mc_test,1,mean)
mean_var_pred = apply(var_mc_test,1,mean)

results_mc <- testX;
results_mc[,3] <- format(round(mean_mu_pred, 6), nsmall = 6)
results_mc[,4] <- format(round(mean_var_pred, 6), nsmall = 6)

write.table(results_mc, file="1.Elmali.Siham.MC.csv",
            sep=",",  col.names=F, row.names=F)
```

```r
## Below is the sample R code to plot the mean of three estimators in a single plot
mean_mulp = apply(muvlp,1,mean);

par(mfrow = c(2, 2));
plot(X1, muhat, col="lightblue",pch=22);
plot(X1, muhat, col="lightblue",pch=22);
points(mean_mulp, x=X1, lwd=1, col="orange")

plot(X2, muhat, col="limegreen",pch=22);
plot(X2, muhat, col="limegreen",pch=22);
points(mean_mulp, x=X2, lwd=1, col="orange")

## Below is the sample R code to plot the mean of three estimators in a single plot
mean_varlp = apply(varlp,1,mean);

par(mfrow = c(2, 2));
plot(X1, Vhat, col="orange",pch=20);
plot(X1, Vhat, col="orange",pch=20);
points(mean_varlp, x=X1, lwd=1, col="lightblue")

plot(X2, Vhat, col="magenta",pch=20);
plot(X2, Vhat, col="magenta",pch=20);
points(mean_varlp, x=X2, lwd=1, col="lightblue")

MSE_mc_mu_lp <- mean((mean_mulp - data0$muhat)^2, na.rm = TRUE)
MSE_mc_var_lp <- mean((mean_varlp - data0$Vhat)^2, na.rm = TRUE)

mse_summary_2 <- cbind(MSE_mc_mu_lp, MSE_mc_var_lp)
colnames(mse_summary_2) <- c( "MC MSE (mu)","MC MSE (var)")

# displaying the tables
knitr::kable(mse_summary_2,"pipe", digit=5,
             caption = "Average MC MSE")%>%
          kable_styling(position = "center", full_width = F)
```

```r
## we can plot 4 graphs in a single plot
par(mfrow = c(2, 2));
plot(results_mc[,1], results_mc[,3], col="lightblue",pch=20);
plot(results_mc[,2], results_mc[,3], col="limegreen",pch=20);
plot(results_mc[,1], results_mc[,4], col="orange",pch=20);
plot(results_mc[,2], results_mc[,4], col="magenta",pch=20);


# library(caret)
#
# #define k-fold cross validation method
# spans <- seq(0.01, 0.97, 0.01)
# ctrl <- trainControl(method = "cv", number = 10)
# grid <- expand.grid(span = spans, degree=1)

# model1 <- train(Vhat ~ X1 + X2, data = data0, method = "gamLoess",
#                 tuneGrid=grid, trControl = ctrl)
#
# #print results of k-fold cross-validation
# print(model1)
#
#
# model2 <- train(muhat ~ X1 + X2, data = data0, method = "gamLoess", tuneGrid=grid, trControl = ctrl)
#
# #print results of k-fold cross-validation
# print(model2)



set.seed(42)
m <- 200
n <- 10000
## Initialize the matrix of fitted values for three methods
muvlp <- varlp <- matrix(0, nrow= n, ncol= m)
mu_mc_test_cv <- var_mc_test_cv <- matrix(0, nrow= 2500, ncol= m)
```

```
##Generate data, fit the data and store the fitted values
for (j in 1:m){

  ## simulate mu-values
  muhat_noise <- muhat + rnorm(length(data0), sd=0.2)

  mod1 <- loess(muhat_noise ~ X1 + X2, span = 0.12, degree = 1,
                data = data0,control=loess.control(surface="direct"))
  muvlp[, j] <- predict(mod1, newdata=data0[, 1:2])



  ## Simulate vhat values
  vhat_noise <- Vhat +  rnorm(length(data0), sd=0.2)
  mod2 <- loess(vhat_noise ~ X1 + X2, span = 0.08, degree = 1,
                data = data0, control=loess.control(surface="direct"))
  varlp[, j] <- predict(mod2, newdata=data0[, 1:2])

  ## prediction on testing data
  mu_mc_test_cv[, j]  <- predict(mod1, newdata=data.frame(X1 = testX$V1, X2=testX$V2))
  var_mc_test_cv[, j] <- predict(mod2, newdata=data.frame(X1 = testX$V1, X2=testX$V2))

}

## Take the mean
mean_mu_pred_cv = apply(mu_mc_test_cv,1,mean)
mean_var_pred_cv = apply(var_mc_test_cv,1,mean)

results_mc_cv <- testX;
results_mc_cv[,3] <- format(round(mean_mu_pred_cv, 6), nsmall = 6)
results_mc_cv[,4] <- format(round(mean_var_pred_cv, 6), nsmall = 6)

write.table(results_mc_cv, file="1.Elmali.Siham.MC.CV.csv",
            sep=",",  col.names=F, row.names=F)
MSE_mc_mu_lp_cv <- mean((mean_mulp - data0$muhat)^2, na.rm = TRUE)
```

```r
MSE_mc_var_lp_cv <- mean((mean_varlp - data0$Vhat)^2, na.rm = TRUE)


mse_summary_3 <- cbind(MSE_mc_mu_lp_cv, MSE_mc_var_lp_cv)
colnames(mse_summary_3) <- c( "10-fold CV MC MSE (mu)","10-fold CV MC MSE (var)")


# displaying the tables
knitr::kable(mse_summary_3,"pipe", digit=5,
             caption = "Average MC MSE after span tuning (10 fold cv)")%>%
          kable_styling(position = "center", full_width = F)
## we can plot 4 graphs in a single plot
par(mfrow = c(2, 2));
plot(results_mc_cv[,1], results_mc_cv[,3], col="lightblue",pch=20)
plot(results_mc_cv[,2], results_mc_cv[,3], col="limegreen",pch=20)
plot(results_mc_cv[,1], results_mc_cv[,4], col="orange",pch=20)
plot(results_mc_cv[,2], results_mc_cv[,4], col="magenta",pch=20)
```