1. Research Question

Objective: In probability and statistics, it is important to understand the mean and variance for any random variables. Suppose that $Y = Y(X_1, X_2)$ is a random variable whose distribution depends on two independent variables X_1 and X_2. Our objective is to estimate two deterministic functions of X_1 and X_2 : one is the mean function $\mu(X_1, X_2) = \mathbf{E}(Y)$ and the other is $V(X_1, X_2) = 100 * \mathrm{Var}(Y)$ when $0 \leq X_1 \leq 7$ and $0 \leq X_2 \leq 4$.

2. Methods

Linear Regression: A linear regression is a simple and yet powerful model that assumes that the regression function $\mathrm{E}(Y \mid X)$ is linear in the inputs $X_1, \ldots, X_p$.. The linear regression model is often preferred for it's simple structure and its interpretability [1–3]. It is also known that these models often outperform some advanced non-linear and complicated models in-terms of predictions.

The Linear regression model can be formulated as follows. For a given input vector $X^T = (X_1, X_2, \ldots, X_p)$ also known as predictors and for Y the response, the linear regression model has the form

$$Y = f(X) + \varepsilon$$

where

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Simply put, the Linear regression model's task is to estimate the deterministic function $f(X)$ through estimating the parameters from the data.

The Linear regression model assumes that the model is linear in the parameters [3, 4]. These parameters $\beta_0, \beta_1, \ldots, \beta_p$ are often estimated using the training data $(x_1, y_1) \ldots (x_N, y_N)$. Among those estimation procedures the least square estimation procedure is widely used. The least square procedure simply estimate the parameters, by considering the values of them which minimize the residual sum of squares. So, the estimates $\hat{\beta} = \left(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p\right)^T$ are obtained by minimizing the following residual sum of squares.

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

The minimization procedure can easily be carried out by writing the residual sum of squares as,

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

Then through differentiation with respect to β we can obtain the unique solution of $\hat{\beta}$.

$$\hat{\beta} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

Using the obtained $\hat{\beta}$, the fitted and predicted values in-case of new observations can be obtained from the following equation.

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

.

Polynomial regression: As we have seen earlier that linear regression model assumes the existence of the linear relationship between response and predictor variables. However, that assumption is often violated in practice. So, the polynomial regression [5, 6] is an extension of the linear regression model which is often used deal with such scenarios. The general formulation of the polynomial regression is as follows,

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j^k \beta_j$$

here the k can be any positive number. When $k = 1$ it is the regular linear regression, when $k = 2$ it becomes a quadratic regression and when $k = 3$ it becomes cubic and so on.

The estimation procedure will be same as we described earlier for the linear regression model.

Random forest: The random forest [7, 8] is ensemble of decision trees aggregated by bootstrapping procedure called bagging, with main focus in reducing the variance of an estimated prediction function. The decision tree is a method which partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one. It started by splitting the space into two regions and model the response by mean in the case of regression or by mode in the case of classification. The variable and split points are chosen by minimizing the cost function. This keep continues until it reaches a stopping point. For example, if the process partition the space into k regions $R_1, R_2, \ldots R_k$. The corresponding regression model predicts Y with a constant c_m in region R_m, that is,

$$\hat{f}(X) = \sum_{m=1}^{k} c_m I(X_1, X_2) \in R_m$$

the best $\hat{c}_m$ is chosen by minimizing of the sum of squares $\sum (y_i - \ f(x_i))^2$.

The random forest is nothing but fitting the same decision tree many times on bootstrap sampled versions of the training data. Then the results are averaged, which is thereby expected to reduce the variance of the predictions. The random forest also allows us to tune several hyper-parameters. By tuning these hyper-parameters, most desirable model is often obtained. Let's see some of these hyper-parameters.

- Number of trees in the forest
- Maximum number of features considered for splitting a node
- Maximum number of levels in each decision tree
- Minimum number of data points placed in a node before the node is split
- Minimum number of data points allowed in a leaf node.

3. Analysis

For given realizations of the Y values which are generated from some function depends on (X_1, X_2) 's, we ought to estimate two deterministic functions of X_1 and X_2 : one is the mean function $\mu(X_1, X_2) = \mathbf{E}(Y)$ and the other is $V(X_1, X_2) = 100 \times \text{Var}(Y)$, which then can be used to approximate the distributions of Y 's.

At the same time it is important to note that there does not exist an universal model which could be used to tackle or model all the problems. As such, the models are always depend on the data. So, in this dark world of choosing an appropriate model, an explanatory analysis of the data would definitely sheds some light.

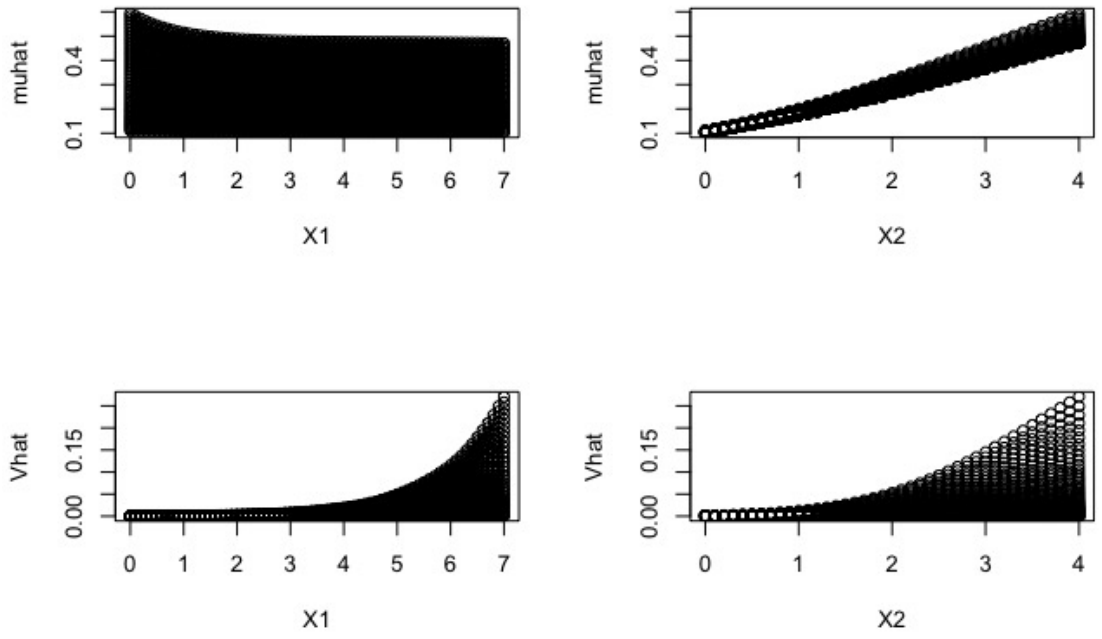

Figure 1: X1 and X2

The figure 1 given here for that purpose. If you notice the plots in that, it is clear that $X2$ shows a strong linear relationship with the estimated mean $(\hat{\mu})$. Meanwhile the $X1$ and $X2$ show clear non-linear pattern with the estimated sample variance of the data $\hat{V}$. So, it can be speculated that a linear model would not be ideal model in this case. Any model that could model a non-linear structure would work. If we want to test all kind of non-linear models, we would spend our entire life, as there exist a huge list of them. A simple an intuitive model for non-linear structure would be a polynomial regression model. The strong theoretical properties which it borrows from the linear model[2–5], is one of many reason why it is preferred for modelling non-linear patterns. It is also necessary to determine a evaluation metric that could help to confirm the efficiency or the performance of the model. This can then can be used to compare the potential models. The root mean square error and coefficient of determination are typical evaluation metrics that is highly used in the regression setting. While coefficient of determination is used to evaluate the model's

performance on the training set, the root mean square error is used to check model's predictive performance on unseen data(test data).

For the y values, predicted $\hat{y}$ values, and for the mean of y values ($\bar{y}$),

The root mean square is defined as,

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

The coefficient of determination is defined as

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

It is worth to note that, even though a test set is given, we don't know $\mathbf{E}(Y)$ and $\text{Var}(Y)$ of the test set. Therefore, it is impossible to check the performance of the model on unseen data. So, we split the given training set in to new train set and test set and then we performed the modelling procedure on former and evaluated on latter.

So, as the base-line model we started to analyze using the regression model and then we ought to see how the polynomial regression model works, and finally we hoped to move to paradigm of non-parametric models. Theses models are evaluated based on the root mean squared error on the test set.

To frame this as a regression model we should introduce some notation. The $E(y)$ can be estimated through the mean of the sample data and lets denote that as $\hat{\mu}$. At the same time lets estimate the $100 \times \text{Var}(Y)$ by the sample variance as $100 \times Var(Y)$ and denote that as $\hat{V}$.

So the linear regression model can be formulated as,

$$\hat{\mu} = f_1(X) + \varepsilon_1$$
$$\hat{V} = f_2(X) + \varepsilon_2$$

where

$$f_1(X) = \beta_0 + X_1\beta_1 + X_2\beta_2$$
$$f_2(X) = \gamma_0 + X_1\gamma_1 + X_2\gamma_2$$

The β, γ are estimated through the least square estimation procedure described earlier. The improvements on the linear regression model have been made by looking at the values of the evaluation metrics. Once the linear and polynomial regression model has been fit, the random forest model has been considered. The detailed result and findings are given in the next section below.

4. Results and Findings

As we described earlier the model fitting is done on the training set and evaluated on the test set.

Table 1: Models for $\hat{\mu}$

Model	R^2	Train RMSE	Test RMSE
Simple Linear Regression	0.9845	0.014679	0.014416
Polynomial Regression degree=2	0.9935	0.009542	0.009312
Polynomial Regression degree=3	0.9950	0.008370	0.008099
Random forest	0.9997	0.002647	0.001294

Given the results above, all the models performs equally well. But random forest is showing a superior performance. One reason that we could think about this performance is its ability to tune the hyper-parameters. This tuning is carried out through a huge grid of these parameters.

This testing procedure is sufficient to evaluate the models' performance when we sufficiently large training and test set. But to get more information, as a general practice to evaluate the performance a cross-validation technique is preferred. Here we implemented one such procedure through this step-wise algorithm given below.

1. First take certain number of observations(N) from the train set and let it as test set.
2. Then fit the model and find its RMSE
3. Then Evaluate its performance on the test set we considered, by its RMSE
4. Carry out all these steps several times and calculate the mean of test and train RMSE

We implemented the above algorithm by considering the N=300 test observations and ran it 10 times. The results are given in the table 2. From the mean RMSE of test and train sets, RF again confirms its superior performance. But the rest of models look stable in-terms of mean RMSE. i.e the difference between test and train errors are almost the same, where the RF shows small discrepancy in that.

Table 2: Cross validation results $\hat{\mu}$

	SLR	**PR** $K = 2$	**PR** $K = 3$	**RF**
Test	0.015249	0.009835	0.008577	0.000870
Train	0.014585	0.009485	0.008321	0.002481

Now we moved on to find the deterministic function for $\hat{V}$. Same procedure mentioned above carried out, the results are given in the table 3. The train and test RMSE looks almost same for the linear regression model and the polynomial regression models. However, the R^2 values suggest that this may not be a good fit. Because the values lie around 0.50, which is very low compared to the models for the $\hat{\mu}$. But thankfully the random forest models comes in handy with reducing the both RMSEs' and increasing the R^2 values.

Table 3: Models for $\hat{V}$

Model	R^2	Train RMSE	Test RMSE
Simple Linear Regression	0.4429	0.024757	0.021406
Polynomial Regression degree=2	0.5749	0.021626	0.018711
Polynomial Regression degree=3	0.5973	0.021049	0.018394
Random forest	0.9977	0.001814	0.000764

It is argued that deriving conclusion by looking at just one sample(test, train) may not be ideal and it may not represent the performance of the model. This would also leads to ill-performance of the model, when given a new test set. To avoid these complications we would like to evaluate the performance of the model through cross validation. This would generalize the performance of the model. So, the algorithm defined earlier implemented here. The results are summarized in the table.

Table 4: Cross validation results $\hat{V}$

	SLR	PR $K = 2$	PR $K = 3$	RF
Test	0.023493	0.020869	0.020452	0.000714
Train	0.024640	0.021489	00.020914	0.003559

The table 4 agrees with the conclusions we have made from $\hat{\mu}$. The RMSE of the linear regression model and the polynomial regression models have increased compared to that of $\hat{\mu}$. But they do not become substantially large so that we could call these models are inappropriate. Anyways the RF model again shows its superior performance in this case too.

5. Conclusions

If we review our research question, we are supposed to find two deterministic functions X_1 and X_2 which are used to estimate $E(y)$ and $100 \times Var(Y)$. Through this we have considered several ways to achieve this. From the simple linear model to the most efficient random forest model. The performance of the random forest outplayed all the other model we have considered here. So, we have decided to define the deterministic model through the random forest model. However, it can not be easily written in a close form like we can do it for the linear regression models. Since this research question mainly focus on the predictive performance of the model, we guess this might not be a problem. With the defined random forest model we have estimated the $\hat{\mu}$ and $\hat{V}$ of the test data given. Mean and the variance of the estimated values turned out to be,

Table 5: Summary of the estimation

Summary	$\hat{\mu}$	$\hat{\mathbf{V}}$
Mean	0.283836	0.008542
Variance	0.014495	0.000426

6. References

[1] Holger Schielzeth. "Simple means to improve the interpretability of regression coefficients". In: *Methods in Ecology and Evolution* 1.2 (), pp. 103–113. DOI: https://doi.org/10.1111/j.2041-210X.2010.00012.x.

[2] Xin Yan and Xiao Gang Su. *Linear Regression Analysis: Theory and Computing*. USA: World Scientific Publishing Co., Inc., 2009. ISBN: 9789812834102.

[3] R.H. Myers. *Classical and Modern Regression with Applications*. Bookware Companion Series. PWS-KENT, 1990. ISBN: 9780534921781. URL: https://books.google.com/books?id=oRLvAAAAMAAJ.

[4] Michael A. Poole and Patrick N. O'Farrell. "The Assumptions of the Linear Regression Model". In: *Transactions of the Institute of British Geographers* 52 (1971), pp. 145–158. ISSN: 00202754, 14755661. URL: http://www.jstor.org/stable/621706.

[5] "Modelling using Polynomial Regression". In: *Procedia Engineering* 48 (2012). Modelling of Mechanical and Mechatronics Systems, pp. 500–506. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2012.09.545. URL: https://www.sciencedirect.com/science/article/pii/S1877705812046085.

[6] Julio L. Peixoto. "A Property of Well-Formulated Polynomial Regression Models". In: *The American Statistician* 44.1 (1990), pp. 26–30. DOI: 10.1080/00031305.1990.10475687.

[7] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007%2F978-0-387-84858-7.

Appendix: Rcode

```
setwd("~/Documents/Class/STAT-983")
library(kernlab)
library(caret)
library(randomForest)
library(e1071)

### Read Training Data
midterm<- read.table(file = "midtermtrain.csv", sep=",");

testIndex=sample(1:2911,size = 200,replace = F)

### Train and test split
midtermtrain<-midterm[-testIndex,]
midtermtest<-midterm[testIndex,]
dim(midtermtrain);

###  exploratory variable
```

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


## note that muhat = E(Y) and Vhat = 100*Var(Y)
muhat <- apply(midtermtrain[,3:202], 1, mean);
Vhat  <- 100*apply(midtermtrain[,3:202], 1, var);

## we can plot 4 graphs in a single plot
par(mfrow = c(2, 2));
plot(X1, muhat);
plot(X2, muhat);
plot(X1, Vhat);
plot(X2, Vhat);
dev.off()

### Train and Test set
train<-data.frame(muhat,Vhat,X1,X2)
test<-data.frame(muhat,Vhat,X1,X2)

### Models for muhat
fit1<-lm(muhat~X1+X2,data = train)
summary(fit1)
sqrt(mean((resid(fit1))^2))
RMSE(predict(fit1,test),test$muhat)

fit2<-lm(muhat~poly(X1,2)+poly(X2,2),data = train)
summary(fit2)
sqrt(mean((resid(fit2))^2))
RMSE(predict(fit2,test),test$muhat)

fit3<-lm(muhat~poly(X1,3)+poly(X2,3),data = train)
summary(fit3)
sqrt(mean((resid(fit3))^2))
RMSE(predict(fit3,test),test$muhat)


fitControl <- trainControl(
  method = "repeatedcv",
  number = 2,
  repeats = 2)

rfGrid <-  expand.grid(mtry=c(1,2))
nrow(rfGrid)
set.seed(824)
rfFit <- train(muhat ~ X1+X2, data = train,
               method = "rf",
```

```
                trControl = fitControl,
                verbose = FALSE,
                tuneGrid = rfGrid)

prerf<-predict(rfFit$finalModel,test[,3:4])
RMSE(prerf,test$muhat)
rfFit$results$Rsquared
rfFit$results$RMSE

### Models for Vhat
fita<-lm(Vhat~X1+X2,data = train)
summary(fita)
sqrt(mean((resid(fita))^2))
RMSE(predict(fita,test),test$Vhat)

fitb<-lm(Vhat~poly(X1,2)+poly(X2,2),data = train)
summary(fitb)
sqrt(mean((resid(fitb))^2))
RMSE(predict(fitb,test),test$Vhat)

fitc<-lm(Vhat~poly(X1,3)+poly(X2,3),data = train)
summary(fitc)
sqrt(mean((resid(fitc))^2))
RMSE(predict(fitc,test),test$Vhat)

set.seed(823)
rfFit2 <- train(Vhat ~ X1+X2, data = ,
                method = "rf",
                trControl = fitControl,
                verbose = FALSE,
                ## Now specify the exact models
                ## to evaluate:
                tuneGrid = rfGrid)

prerf2<-predict(rfFit2$finalModel,test[,3:4])
RMSE(prerf2,test$Vhat)
rfFit2$results$Rsquared
rfFit2$results$RMSE

### Cross-validation for muhat
fit1Train=fit2Train=fit3Train=fit1Test=fit2Test=fit3Test=fit4Test=fit4Train<-c()

for (i in 1:10){
  X1 <- midterm[,1];
  X2 <- midterm[,2];

  muhat <- apply(midterm[,3:202], 1, mean);
```

```
  Vhat  <- 100*apply(midterm[,3:202], 1, var);
  boot<-data.frame(muhat,Vhat,X1,X2)

  bootindex<-sample(1:2911,size = 300,replace = F)
  bootTrian<-boot[-bootindex,]
  bootTest<-boot[bootindex,]


  fit1<-lm(muhat~X1+X2,data = bootTrian)
  summary(fit1)
  fit1Train<-c(fit1Train,sqrt(mean((resid(fit1))^2)))
  fit1Test<-c(fit1Test,RMSE(predict(fit1,bootTest),bootTest$muhat))

  fit2<-lm(muhat~poly(X1,2)+poly(X2,2),data = bootTrian)
  summary(fit2)
  fit2Train<-c(fit2Train,sqrt(mean((resid(fit2))^2)))
  fit2Test<-c(fit2Test,RMSE(predict(fitb,bootTest),bootTest$muhat))

  fit3<-lm(muhat~poly(X1,3)+poly(X2,3),data = bootTrian)
  summary(fit3)
  fit3Train<-c(fit3Train,sqrt(mean((resid(fit3))^2)))
  fit3Test<-c(fit3Test,RMSE(predict(fitc,bootTest),bootTest$muhat))

  rfFitb <- train(muhat ~ X1+X2, data = train,
                  method = "rf",
                  trControl = fitControl,
                  verbose = FALSE,
                  tuneGrid = rfGrid)

  prerfb<-predict(rfFitb$finalModel,bootTest[,3:4])
  fit4Test<-c(fit4Test,RMSE(prerfb,bootTest$muhat))
  fit4Train<-c(fit4Train,rfFitb$results$RMSE)
}

### Cross-validation for Vhat
fitaTrain=fitbTrain=fitcTrain=fitaTest=fitbTest=fitcTest=fitdTest=fitdTrain<-c()

for (i in 1:10){
  X1 <- midterm[,1];
  X2 <- midterm[,2];

  muhat <- apply(midterm[,3:202], 1, mean);
  Vhat  <- 100*apply(midterm[,3:202], 1, var);
  boot<-data.frame(muhat,Vhat,X1,X2)

  bootindex<-sample(1:2911,size = 300,replace = F)
  bootTrian<-boot[-bootindex,]
```

```
  bootTest<-boot[bootindex,]


  fita<-lm(Vhat~X1+X2,data = bootTrian)
  summary(fita)
  fitaTrain<-c(fitaTrain,sqrt(mean((resid(fita))^2)))
  fitaTest<-c(fitaTest,RMSE(predict(fita,bootTest),bootTest$Vhat))

  fitb<-lm(Vhat~poly(X1,2)+poly(X2,2),data = bootTrian)
  summary(fitb)
  fitbTrain<-c(fitbTrain,sqrt(mean((resid(fitb))^2)))
  fitbTest<-c(fitbTest,RMSE(predict(fitb,bootTest),bootTest$Vhat))

  fitc<-lm(Vhat~poly(X1,3)+poly(X2,3),data = bootTrian)
  summary(fitc)
  fitcTrain<-c(fitcTrain,sqrt(mean((resid(fitc))^2)))
  fitcTest<-c(fitcTest,RMSE(predict(fitc,bootTest),bootTest$Vhat))

  rfFitb <- train(Vhat ~ X1+X2, data = train,
                  method = "rf",
                  trControl = fitControl,
                  verbose = FALSE,
                  tuneGrid = rfGrid)

  prerfb<-predict(rfFitb$finalModel,bootTest[,3:4])
  fitdTest<-c(fitdTest,RMSE(prerfb,bootTest$Vhat))
  fitdTrain<-c(fitdTrain,rfFitb$results$RMSE)
}
mean(fitcTest)
mean(fitcTrain)

### Test set invoking
midtermtest  <- read.table(file = "midtermtest.csv", sep=",");
names(midtermtest)<-c("X1","X2")

### Prediction
prerf<-predict(rfFit$finalModel,midtermtest)
prerf2<-predict(rfFit2$finalModel,midtermtest)

### Predictions and predictors
Results<-data.frame(midtermtest,muhat=round(prerf,6),Vhat=round(prerf2,6))

mean(prerf)
#0.283836
var(prerf)
#0.014495
mean(prerf2)
```

```
#0.008542836
var(prerf2)
#0.000426

### Output is written to a file
write.table(Results, "results.csv", row.names=F, col.names=F, sep=",")
```