# Statistical Learning Lab: Assignment-04

## Cross Validation and Bootstrapping

**Name: Semanti Ghosh**

**Roll No: 22IM10036**

1. **Loading the data and printing the first few lines**

Code snippet

```
#getting to the current working directory
setwd("C:/Study/Semester_6/Statistical_Learning_Lab")
getwd()

#reading and printing the data
data <- read.csv("manufacturing.csv")
head(data)
```

Output

```
> #getting to the current working directory
> setwd("C:/Study/Semester_6/Statistical_Learning_Lab")
> getwd()
[1] "C:/Study/Semester_6/Statistical_Learning_Lab"
>
> #reading and printing the data
> data <- read.csv("manufacturing.csv")
> head(data)
  Temperature...C. Pressure..kPa. Temperature.x.Pressure Material.Fusion.Metric
1         209.7627       8.050855               1688.769               44522.22
2         243.0379      15.812068               3842.931               63020.76
3         220.5527       7.843130               1729.823               49125.95
4         208.9766      23.786089               4970.737               57128.88
5         184.7310      15.797812               2918.345               38068.20
6         229.1788       8.498306               1947.632               53136.69
  Material.Transformation.Metric Quality.Rating
1                        9229576       99.99997
2                       14355367       99.98570
3                       10728389       99.99976
4                        9125702       99.99997
5                        6303792      100.00000
6                       12037072       99.99879
>
```

## 2. Fitting polynomials on temperature from 1 to 5 degrees and then performing LOOCV and k-Fold CV

Code Snippet

```
#varying the degree of polynomial on temperature
library(ggplot2)
models <- list()
degrees <- 1:5

for (d in degrees) {
  formula <- as.formula(paste("Quality.Rating ~ poly(Temperature...C.,", d, ", raw=TRUE)"))
  models[[d]] <- lm(formula, data = data)
}

library(boot)

#initialization
cv_results <- data.frame(Degree=integer(), LOOCV=numeric(), k5=numeric(), k10=numeric())

#performing LOOCV and k-fold CV (doing it for different degrees)
for (d in degrees) {
  formula <- as.formula(paste("Quality.Rating ~ poly(Temperature...C.,", d, ", raw=TRUE)"))
  model <- glm(formula, data = data)

  loocv_error <- cv.glm(data, model)$delta[1]
  k5_error <- cv.glm(data, model, K=5)$delta[1]
  k10_error <- cv.glm(data, model, K=10)$delta[1]

  cv_results <- rbind(cv_results, data.frame(Degree=d, LOOCV=loocv_error, k5=k5_error, k10=k10_error)
}

#Displaying the results in a table
install.packages("pander")
library(pander)
pander(cv_results)
```

Output

```
>
> for (d in degrees) {
+   formula <- as.formula(paste("Quality.Rating ~ poly(Temperature...C.,", d, ", raw=TRUE)"))
+   models[[d]] <- lm(formula, data = data)
+ }
>
> library(boot)
>
> #initialization
> cv_results <- data.frame(Degree=integer(), LOOCV=numeric(), k5=numeric(), k10=numeric())
>
> #performing LOOCV and k-fold CV (doing it for different degrees)
> for (d in degrees) {
+   formula <- as.formula(paste("Quality.Rating ~ poly(Temperature...C.,", d, ", raw=TRUE)"))
+   model <- glm(formula, data = data)
+
+   loocv_error <- cv.glm(data, model)$delta[1]
+   k5_error <- cv.glm(data, model, K=5)$delta[1]
+   k10_error <- cv.glm(data, model, K=10)$delta[1]
+
+   cv_results <- rbind(cv_results, data.frame(Degree=d, LOOCV=loocv_error, k5=k5_error, k10=k10_error))
+ }
> library(pander)
> pander(cv_results)
```

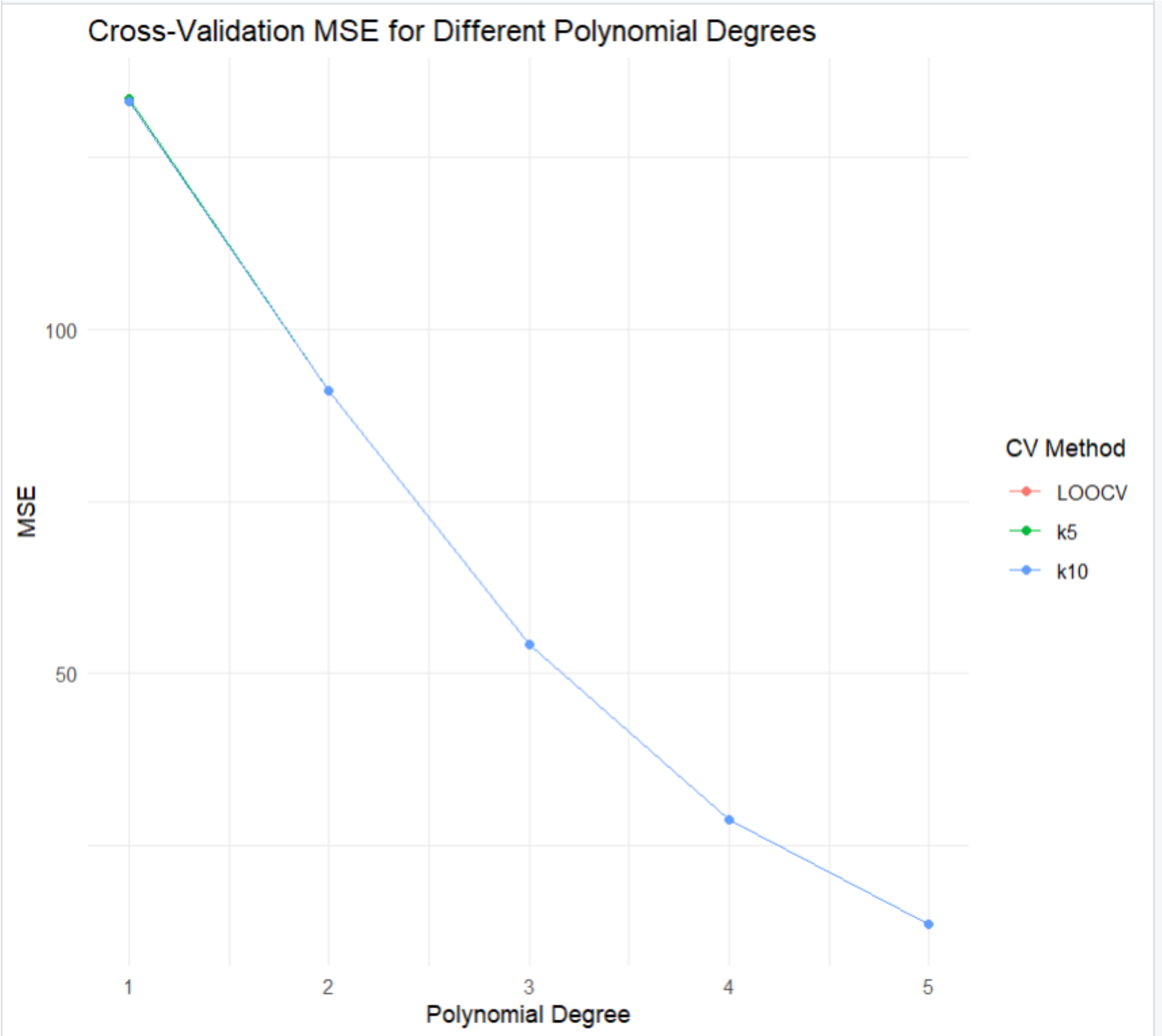
| Degree | LOOCV | k5 | k10 |
|---|---|---|---|
| 1 | 133.1 | 133.5 | 133.1 |
| 2 | 91.19 | 91.09 | 91.09 |
| 3 | 54.24 | 54.2 | 54.15 |
| 4 | 28.79 | 28.69 | 28.76 |
| 5 | 13.53 | 13.56 | 13.49 |

Plotting the results

Code snippet

```
# plotting the results
cv_results_long <- reshape2::melt(cv_results, id="Degree")

ggplot(cv_results_long, aes(x=Degree, y=value, color=variable)) +
  geom_line() + geom_point() +
  labs(title="Cross-Validation MSE for Different Polynomial Degrees",
       x="Polynomial Degree", y="MSE", color="CV Method") +
  theme_minimal()
```

Output




**Analysis:** From the above graph and results, we can say that when we fit a polynomial of degree 5, then the MSE is minimised. This is expected since the training and validation set error does decrease when the degree of the polynomial increases, since then the polynomial becomes more flexible. However, we should also keep in mind that if the degree of the polynomial is too high, then it will overfit the data, resulting in high variance. So, while the results suggest that 5 is the best choice, we might choose a lower degree polynomial (like maybe three or four degree polynomial). This will make the model more robust and ensure generalisation on unseen data.

## 3. Fitting linear polynomials for different variables

### Code Snippet

```
#trying out for linear combinations of different variables(excluding interactions)
library(reshape2)

# Define all variable combinations
var_combinations <- list(
  "Temp" = c("Temperature...C."),
  "Temp-Press" = c("Temperature...C.", "Pressure..kPa."),
  "Temp-MatFus" = c("Temperature...C.", "Material.Fusion.Metric"),
  "Temp-MatTrans" = c("Temperature...C.", "Material.Transformation.Metric"),
  "Temp-Press-MatFus" = c("Temperature...C.", "Pressure..kPa.", "Material.Fusion.Metric"),
  "Temp-Press-MatTrans" = c("Temperature...C.", "Pressure..kPa.", "Material.Transformation.Metric"),
  "Temp-MatFus-MatTrans" = c("Temperature...C.", "Material.Fusion.Metric", "Material.Transformation.Metric"),
  "Temp-Press-MatFus-MatTrans" = c("Temperature...C.", "Pressure..kPa.", "Material.Fusion.Metric", "Material.Transformation.Metric")
)

cv_results <- data.frame(Model=character(), LOOCV=numeric(), k5=numeric(), k10=numeric())

for (model_name in names(var_combinations)) {
  formula <- as.formula(paste("Quality.Rating ~", paste(var_combinations[[model_name]], collapse = " + ")))
  model <- glm(formula, data = data)

  loocv_error <- cv.glm(data, model)$delta[1]
  k5_error <- cv.glm(data, model, K=5)$delta[1]
  k10_error <- cv.glm(data, model, K=10)$delta[1]

  cv_results <- rbind(cv_results, data.frame(Model=model_name, LOOCV=loocv_error, k5=k5_error, k10=k10_error))
}

#printing the results
print(cv_results)
```

### Output

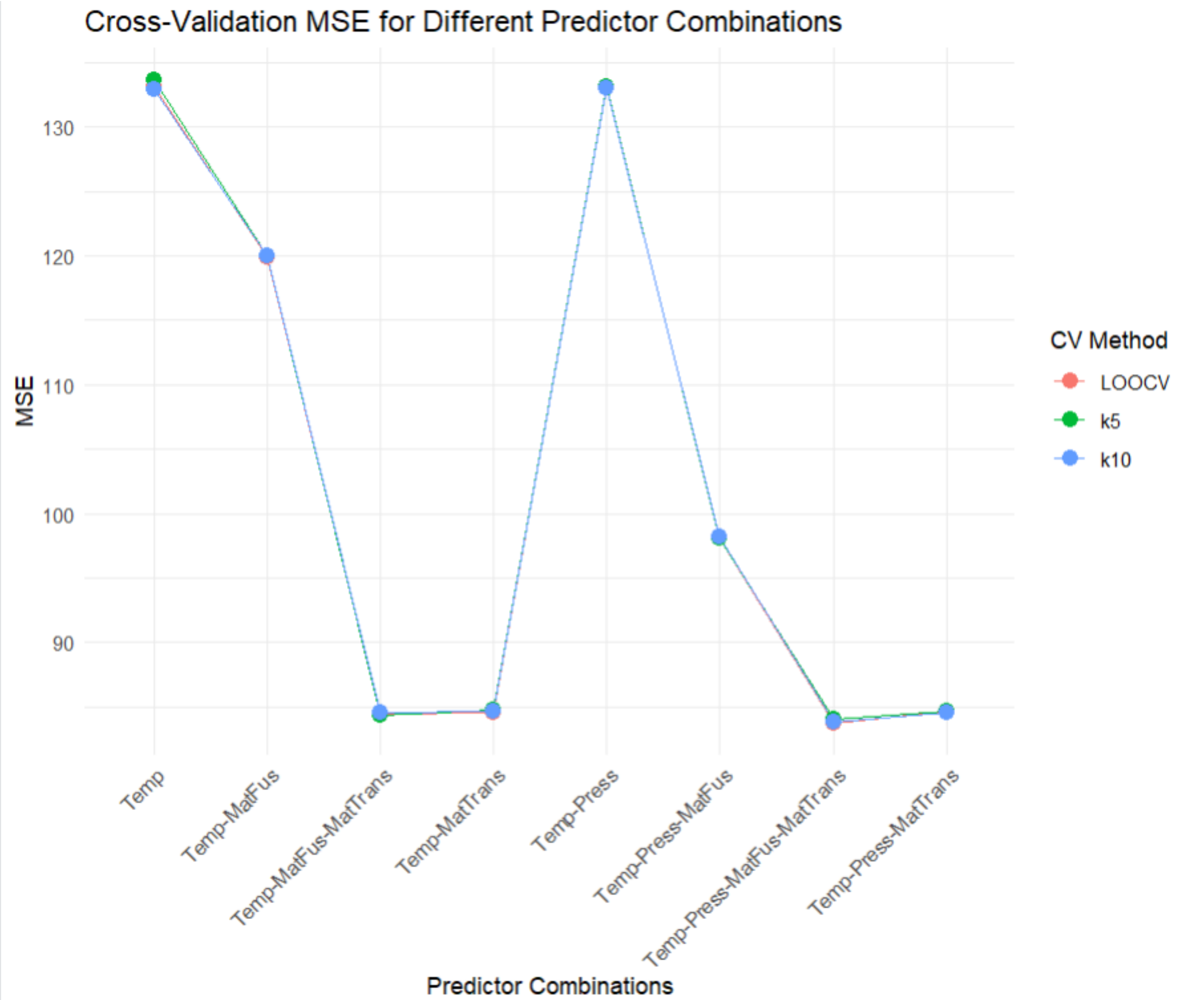
```
#trying out for linear combinations of different variables(excluding interactions)
library(ggplot2)
library(reshape2)

# Define all variable combinations
var_combinations <- list(
  "Temp" = c("Temperature...C."),
  "Temp-Press" = c("Temperature...C.", "Pressure..kPa."),
  "Temp-MatFus" = c("Temperature...C.", "Material.Fusion.Metric"),
  "Temp-MatTrans" = c("Temperature...C.", "Material.Transformation.Metric"),
  "Temp-Press-MatFus" = c("Temperature...C.", "Pressure..kPa.", "Material.Fusion.Metric"),
  "Temp-Press-MatTrans" = c("Temperature...C.", "Pressure..kPa.", "Material.Transformation.Metric"),
  "Temp-MatFus-MatTrans" = c("Temperature...C.", "Material.Fusion.Metric", "Material.Transformation.Metric"),
  "Temp-Press-MatFus-MatTrans" = c("Temperature...C.", "Pressure..kPa.", "Material.Fusion.Metric", "Material.Transformation.Metric")
)

cv_results <- data.frame(Model=character(), LOOCV=numeric(), k5=numeric(), k10=numeric())

for (model_name in names(var_combinations)) {
  formula <- as.formula(paste("Quality.Rating ~", paste(var_combinations[[model_name]], collapse = " + ")))
  model <- glm(formula, data = data)

  loocv_error <- cv.glm(data, model)$delta[1]
  k5_error <- cv.glm(data, model, K=5)$delta[1]
  k10_error <- cv.glm(data, model, K=10)$delta[1]

  cv_results <- rbind(cv_results, data.frame(Model=model_name, LOOCV=loocv_error, k5=k5_error, k10=k10_error))
}

# Print the CV results
print(cv_results)
                       Model     LOOCV        k5       k10
                        Temp 133.07880 133.64967 132.91792
                  Temp-Press 133.14410 133.09729 133.04600
                 Temp-MatFus 119.93438 119.97232 119.96603
               Temp-MatTrans  84.59484  84.80396  84.69679
           Temp-Press-MatFus  98.12928  98.06263  98.20927
         Temp-Press-MatTrans  84.63338  84.62915  84.53809
        Temp-MatFus-MatTrans  84.50945  84.31112  84.56092
  Temp-Press-MatFus-MatTrans  83.76925  84.04034  83.83754
```

Plotting the results

Code Snippet

```
cv_results_long <- melt(cv_results, id="Model")
|
ggplot(cv_results_long, aes(x=Model, y=value, color=variable)) +
  geom_line(aes(group=variable)) +
  geom_point(size=3) +
  labs(title="Cross-Validation MSE for Different Predictor Combinations",
       x="Predictor Combinations", y="MSE", color="CV Method") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1))  # Rotate labels for readability
```

Output




**Analysis**: From the results obtained, we can say that the error is minimised when all four parameters (Temperature, Pressure, Material Fusion Metric and Material

Transformation Metric) are taken into consideration. However, like in the previous case, this can lead to overfitting resulting in high variance. So, even if the error is slightly greater, it's safer to go for three parameters, like in Temperature, Material Fusion Metric, Material Transformation Metric. This will make the model more robust and ensure generalisation on unseen data.

## 4. Bootstrapping

Code snippet

```
#Bootstrapping

#generating the random variables that follow Gaussian distribution
set.seed(123)
data <- rnorm(50, mean = 50, sd = sqrt(2))

bootstrap_means <- numeric(100)
bootstrap_vars <- numeric(100)

#random sampling with replacement
for (i in 1:100) {
  sample_data <- sample(data, size = 20, replace = TRUE)
  bootstrap_means[i] <- mean(sample_data)
  bootstrap_vars[i] <- var(sample_data)
}

#estimating mean and variance from the
boot_mean_estimate <- mean(bootstrap_means)
boot_var_estimate <- mean(bootstrap_vars)

cat("Estimated Population Mean from Bootstrap Samples:", boot_mean_estimate, "\n")
cat("Estimated Population Variance from Bootstrap Samples:", boot_var_estimate, "\n")
```

Output

```
> #Bootstrapping
>
> #generating the random variables that follow Gaussian distribution
> set.seed(123)
> data <- rnorm(50, mean = 50, sd = sqrt(2))
>
> bootstrap_means <- numeric(100)
> bootstrap_vars <- numeric(100)
>
> #random sampling with replacement
> for (i in 1:100) {
+   sample_data <- sample(data, size = 20, replace = TRUE)
+   bootstrap_means[i] <- mean(sample_data)
+   bootstrap_vars[i] <- var(sample_data)
+ }
>
> #estimating mean and variance from the
> boot_mean_estimate <- mean(bootstrap_means)
> boot_var_estimate <- mean(bootstrap_vars)
>
> cat("Estimated Population Mean from Bootstrap Samples:", boot_mean_estimate, "\n")
Estimated Population Mean from Bootstrap Samples: 50.05668
> cat("Estimated Population Variance from Bootstrap Samples:", boot_var_estimate, "\n")
Estimated Population Variance from Bootstrap Samples: 1.733931
```

**Result**

**Estimated Population Mean = 50.05668**

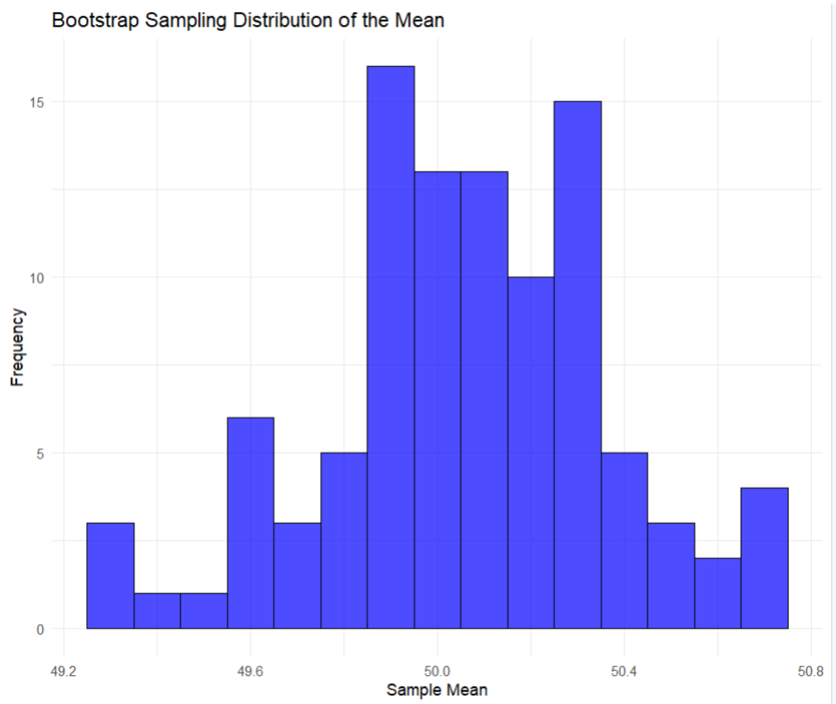
**Estimated Population Variance = 1.733931**

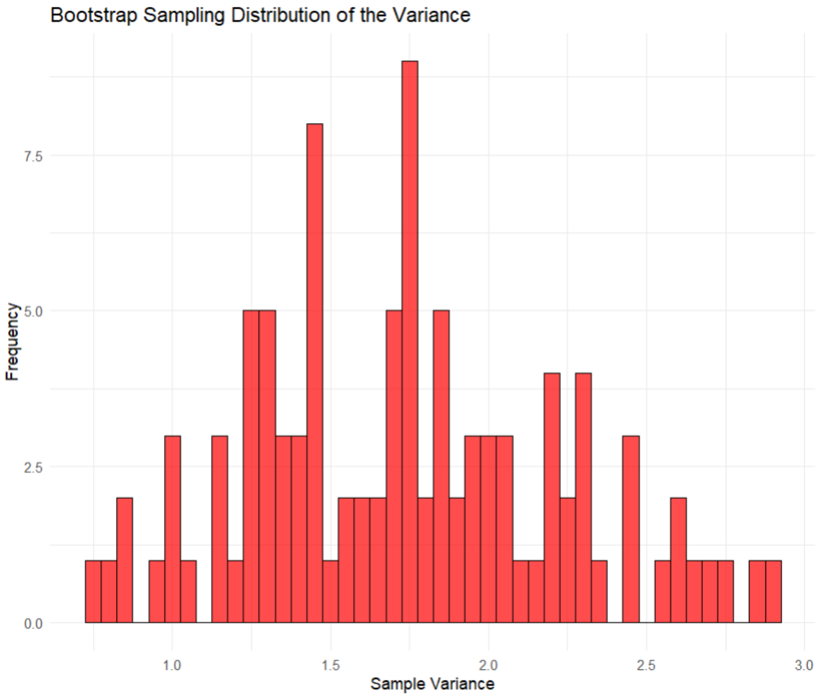
Plotting the graph of means and variances in bootstrapping for better visualisation

Code Snippet

```
#plotting the bootstrapping means and variances (just for my own understanding)

p1 <- ggplot(data.frame(Means = bootstrap_means), aes(x = Means)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Bootstrap Sampling Distribution of the Mean",
       x = "Sample Mean", y = "Frequency") +
  theme_minimal()

p2 <- ggplot(data.frame(Variances = bootstrap_vars), aes(x = Variances)) +
  geom_histogram(binwidth = 0.05, fill = "red", color = "black", alpha = 0.7) +
  labs(title = "Bootstrap Sampling Distribution of the Variance",
       x = "Sample Variance", y = "Frequency") +
  theme_minimal()

print(p1)
print(p2)
```

Output




Bootstrap Sampling Distribution of the Variance
Frequency
Sample Variance
0.0
2.5
5.0
7.5
1.0
1.5
2.0
2.5
3.0