

CROSS VALIDATION

Introduction: In this assignment, we are going to use k-fold cross validation on the selected models. I have selected the data of seeds with the columns “Area”, “Perimeter”, “Compactness”, “Kernal.length”, “Kernal.width”, “Asymmetry.Coeff”, “Kernel.Groove” and “Type”. Basically, In our data, We can check which type of seed is according to the area, perimeter, compactness of the seeds.

```
#Import the data
seeds <- read.csv("~/Docs/Business Analytics/Classification/seeds.csv")

#view data
View(seeds)
```

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
1	15.26	14.84	0.8710	5.763	3.312	2.2210	5.220	1
2	14.88	14.57	0.8811	5.554	3.333	1.0180	4.956	1
3	14.29	14.09	0.9050	5.291	3.337	2.6990	4.825	1
4	13.84	13.94	0.8955	5.324	3.379	2.2590	4.805	1
5	16.14	14.99	0.9034	5.658	3.562	1.3550	5.175	1
6	14.38	14.21	0.8951	5.386	3.312	2.4620	4.956	1
7	14.69	14.49	0.8799	5.563	3.259	3.5860	5.219	1
8	16.63	15.46	0.8747	6.053	3.465	2.0400	5.877	1
9	16.44	15.25	0.8880	5.884	3.505	1.9690	5.533	1
10	15.26	14.85	0.8696	5.714	3.242	4.5430	5.314	1

Showing 1 to 11 of 199 entries, 8 total columns

Objective: There are various models that need to adapt for optimal prediction performance such as regression, tree etc. We will estimate objectively the performance of different models or sub models to choose the best model according to our data. In this Assignment, I will use k-fold cross validation to test how they are likely to behave on the different models. Following are the steps to estimate the prediction error:

1. Split the data into two sets and build the model on training data set.
2. Make predictions by applying the model on a new test data.
3. Compute the prediction error.

```
#importing all the libraries
library(dplyr)
library(tidyverse)
library(ggplot2)
```

```
library(reshape2)
library(Information)
library(gridExtra)
library(stringr)
library(caret)
library(CombMSC)
```

Splitting the data into training and test set

I have splitted the data set into two sets:

Trainset: This set is used for model estimation by estimating model parameters. I have taken 80% of the data into this set.

Testset: I have selected 20% data for model assessment by estimating test error of the final chosen model.

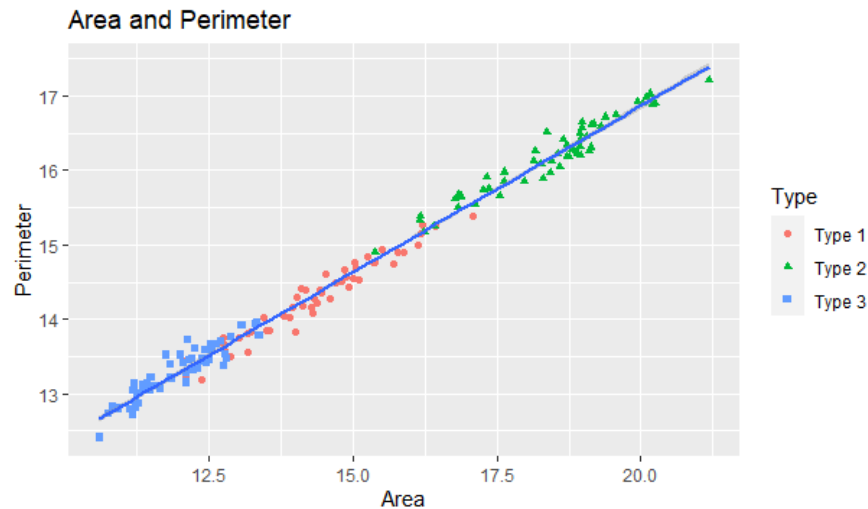
```
# Create a partition on dataset (80% training 20% testing)
data_split <- createDataPartition(seeds$Type, p = 0.8, list = FALSE)
# select 20% data for training
testset <- seeds[-data_split,]
# select 80% data to build or train the models
trainset <- seeds[data_split,]
```

Understanding the data set: Before we begin with our models, It's best to understand and analyze the variables. Now, I can see that there are no NA values and missed values in my data.

```
g <- ggplot(data=train.data, aes(x = Area, y = Perimeter))
print(g)

g <- g +
  geom_point(aes(color=Type, shape=Type)) +
  xlab("Area") +
  ylab("Perimeter") +
  ggtitle("Area and Perimeter")+
  geom_smooth(method="lm")

print(g)
```

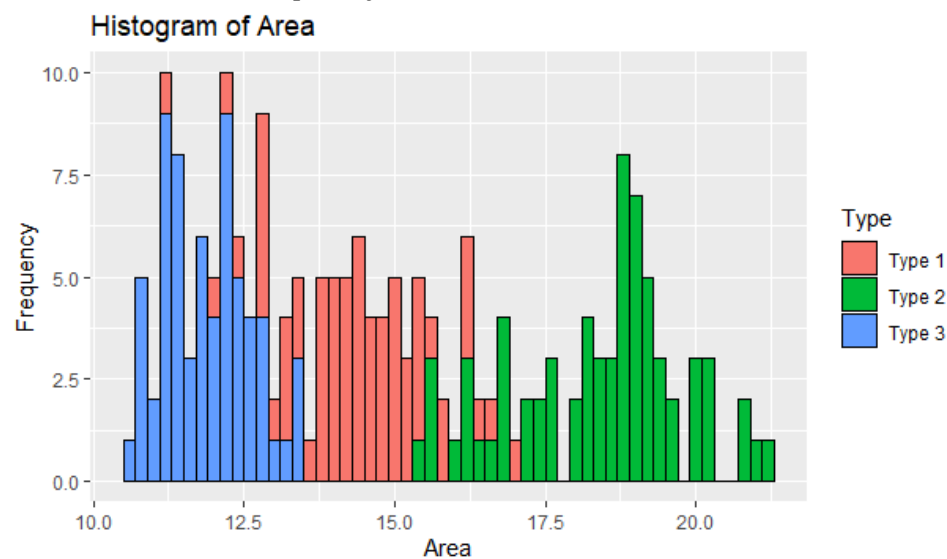


Scatter Plot helps to visualize the relationship between response and predictor variables. We have 7 predictors in this data. The scatter plot along with the smoothing line shows the linearly increasing relationship between Area and Perimeter variables.

Histogram

```
histogram <- ggplot(data=seeds, aes(x=Area)) +  
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +  
  xlab("Area") +  
  ylab("Frequency") +  
  ggtitle("Histogram of Sepal Width")  
print(histogram)
```

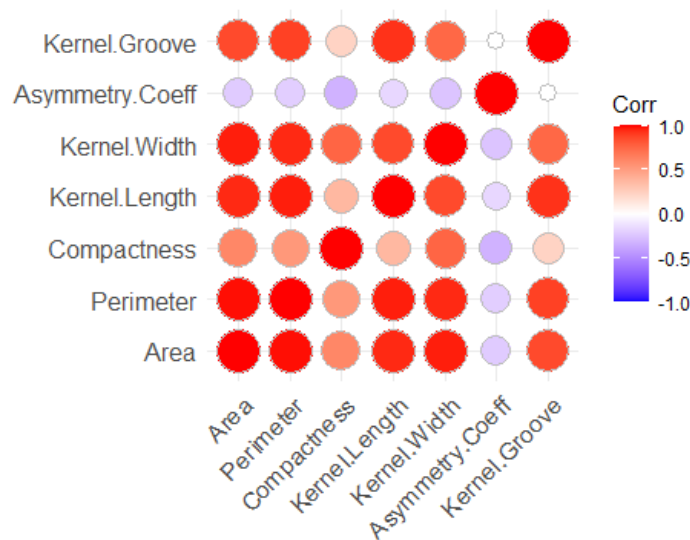
We can check the frequency distribution of data in the below screenshot.



Correlation of data

Finding the correlation between the features or predictors is an important in the model because we can use the correlation to make the predictions. Correlation takes values between 1 to -1.

```
#correlation of data
correlation_data <- cor(seeds[,1:7])
ggcorrplot(correlation_data, method = "circle")
```



We can see in the screenshot above that 5 predictors are highly correlated i.e. Area, Perimeter, Kernel.Width, Kernel.Length, and Kernel.Groove where as Asymmetry.Coeff and Compactness are less correlated.

According to our data if we increase the area of the seeds then the perimeter and kernel length, width and Groove will also increase that are showing high correlation with each other. Value of Compactness is closer to 0 that shows the weak relationship with variables whereas asymmetry.Coeff's value is nearby -0.5 shows low correlation.

In our modelling we will be using these five predictors to analyse our model.

Building linear regression

```
# build linear regression model on full data
linearMod <- lm(Type ~ Area, data=seeds)
print(linearMod)
```

We have already seen the linear relationship of predictor and response variable on scatter plot and by plotting correlation.

```
Call:
lm(formula = Type ~ Area, data = seeds)

Coefficients:
(Intercept)      Area
   3.41069    -0.09489
```

Predicting Linear Model

```
# Split the data into training and test set
training.samples <- seeds$Type %>%
createDataPartition(p = 0.8, list = FALSE)
train.data <- seeds[training.samples, ]
test.data <- seeds[-training.samples, ]
#Build model on training data
model <- lm(Type ~ Area, data=train.data)
#make predictions on test data
TypePred <- predict(model, test.data)
summary(model)
```

The best way to practice to check the performance with new data is to split the data into two parts.

I have split the data set into two sets:

Trainset: This set is used for model estimation by estimating model parameters. I have taken 80% of the data into this set.

Testset: I have selected 20% data for model assessment by estimating test error of the final chosen model.

According to this way, we can find the model predicted values for the test data(i.e. 20%) as well as actual values. We can calculate the prediction accuracy by calculating accuracy measures and error rates.

```
Call:
lm(formula = Type ~ Area, data = train.data)

Residuals:
    Min       1Q   Median       3Q      Max
-1.3359 -0.9867  0.3466  0.6671  0.8600

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.36399    0.31580  10.652  < 2e-16 ***
Area        -0.09155    0.02070   -4.422  1.81e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.769 on 158 degrees of freedom
Multiple R-squared:  0.1101, Adjusted R-squared:  0.1045
```

```
F-statistic: 19.55 on 1 and 158 DF,  p-value: 1.814e-05
```

Now, We have found the p-value, F-statistic, Residual standard error, Adjusted R square. P-value is very important in analysis, we can check if our linear model is statistically significant or not that generally when the p-value is less than 0.05.

F-statistic is basically on the ratio of mean squares. The more the value, the better the model.

```
# make predictions and compute the values of R2, RMSE, MAE
predictions <- model %>% predict(test.data)
data.frame( R2 = R2(predictions, test.data$Type),
  RMSE = RMSE(predictions, test.data$Type), MAE = MAE(predictions, test.data$Type))

RMSE(predictions, test.data$Type)/mean(test.data$Type)
```

R2, RMSE and MAE are used to check the accuracy of regression model. MAE (Mean absolute error) is the sum of absolute differences between our observed outcome and predicted values.

RMSE is the standard deviation of the prediction errors.

R2 represents squared correlation between observed outcome values and predicted values. The higher the R2, better the model.

```
OUTPUT:
      R2      RMSE      MAE
1 0.1569117 0.7238567 0.6433479
```

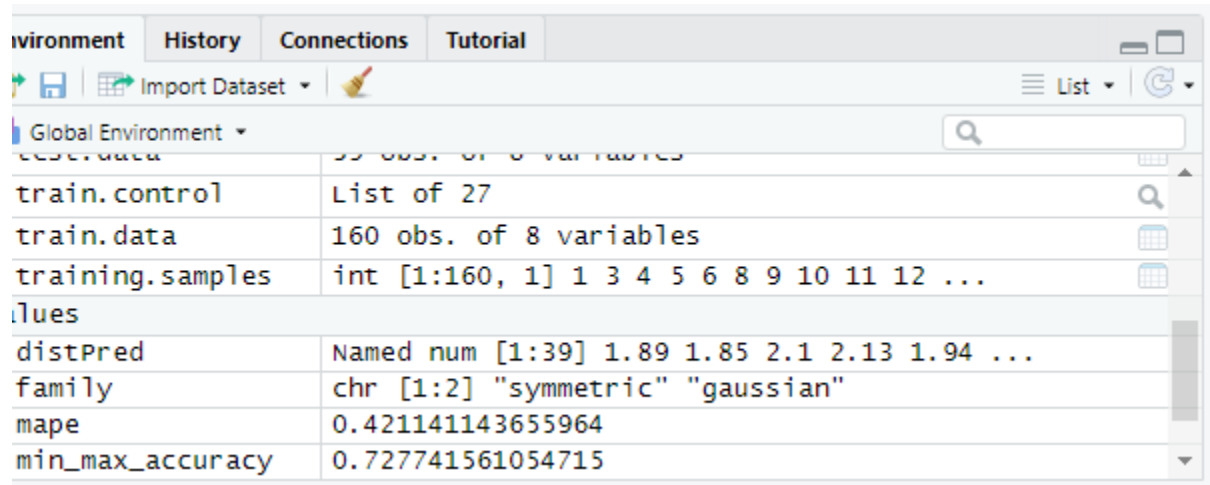
Both MAE and RMSE represents average prediction error, range from 0 to ∞ , and negatively oriented scores, which means lower the R2 and MAE, better the model.

```
#Calculate prediction accuracy
actuals_preds <- data.frame(cbind(actuals=test.data$Type, predicted=TypePred)) #
make actuals_predicted dataframe.
correlation_accuracy <- cor(actuals_preds)
head(actuals_preds)
```

```
#check the accuracy of model
```

```
min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
```

```
mape <- mean(abs((actuals_preds$predicted -  
actuals_preds$actuals))/actuals_preds$actuals)
```



The screenshot shows the RStudio Global Environment pane. It contains a table of objects in the workspace. The objects and their types/values are:

Object	Type/Value
train.control	List of 27
train.data	160 obs. of 8 variables
training.samples	int [1:160, 1] 1 3 4 5 6 8 9 10 11 12 ...
distPred	Named num [1:39] 1.89 1.85 2.1 2.13 1.94 ...
family	chr [1:2] "symmetric" "gaussian"
mape	0.421141143655964
min_max_accuracy	0.727741561054715

Now, we can see that the accuracy of our model is 72%.

K-Fold Cross Validation for linear regression

K-fold cross validation is procedure to evaluate the performance of the algorithm on different dataset. In this technique, we reserve the sample of dataset and then train the model on remaining part of the dataset.

K-fold algorithm is as follow:

1. We have selected k fold number=10 and randomly split the data into k-subsets (here value=10)
2. Reserve one set and train the model on all other sets. In first iteration, the first fold is used to test the model and others are used to train the model. In second iteration, second fold is used to test the model and others are used to train the model.
3. After testing the model on reserve subset, we can record the prediction error. This process will repeat 10 times until all the folds have been used as testing set.

1. USING 5 PREDICTORS

```
#k-fold cross validation
```

```
set.seed(9)
```

```
# Define training control
```

```
train.control <- trainControl(method = "cv", number = 10)
```

```
# Train the model
```

```
model <- train(Type ~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove
```

```
, data = seeds, method = "lm", trControl = train.control)
# Summarize the results
print(model)
```

OUTPUT:

Linear Regression

199 samples
5 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 179, 180, 179, 178, 179, 179, ...

Resampling results:

RMSE	Rsquared	MAE
0.4760658	0.6565677	0.3766026

Tuning parameter 'intercept' was held constant at a value of TRUE

CP, AIC and BIC values:

AIC and BIC are commonly used metrics to check the model evaluation and selection. By adding additional variables, AIC adds a penalty to increase the error. BIC adds stronger penalty for including additional variables.

```
#CP, AIC and BIC Criterion values
```

```
s2 <- sigma(lm(Type ~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove,  
data=train.data)) #extracts residual standard error from lm
```

```
n = nrow(train.data)
```

```
c(Cp(model,S2=(s2^2)), AIC(model,k=2),AIC(model,k=log(n)))
```

OUTPUT:

```
[1] -85.51838 281.27124 302.79746
```

We have calculated the AIC,BIC and CP values using 5 predictors. Now, we can change the predictors every time and analyze the results. Lower the AIC and BIC values, better the model. Cp should be close to the number of predictors in the model.

2. USING 4 PREDICTORS


```

#k-fold cross validation
set.seed(9)
# Define training control
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Type ~ Area + Perimeter + Kernel.Length + Kernel.Width , data = seeds,
method = "lm",trControl = train.control)

# Summarize the results

print(model)
summary(model)

#AIC Values
model <- lm(Type ~ Area + Perimeter + Kernel.Length + Kernel.Width , data = seeds)

AIC(model)
BIC(model)

```

OUTPUT:

Linear Regression

199 samples
4 predictor

No pre-processing
Resampling: Cross-validated (10 fold)
Summary of sample sizes: 179, 180, 179, 178, 179, 179, ...
Resampling results:

RMSE	Rsquared	MAE
0.6777101	0.3370008	0.5739575

Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:

Min	1Q	Median	3Q	Max
-1.6144	-0.5589	0.1769	0.4855	1.5964

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	31.04819	5.63136	5.513	1.11e-07 ***
Area	1.75970	0.35310	4.984	1.38e-06 ***
Perimeter	-2.35810	0.67451	-3.496	0.000585 ***
Kernel.Length	-0.03397	0.72388	-0.047	0.962618
Kernel.Width	-6.33756	1.00314	-6.318	1.78e-09 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 0.6773 on 194 degrees of freedom
Multiple R-squared: 0.3206, Adjusted R-squared: 0.3066
F-statistic: 22.89 on 4 and 194 DF, p-value: 1.672e-15
> AIC(model)
[1] 416.6065
> BIC(model)
[1] 435.0576

```

3. USING 3 PREDICTORS

```

#k-fold cross validation
set.seed(9)
# Define training contro
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Type ~ Area + Perimeter + Kernel.Length, data = seeds, method = "lm",
               trControl = train.control)

# Summarize the results
print(model)
summary(model)
model <- lm(Type ~ Area + Perimeter + Kernel.Length, data = seeds)
AIC(model)
BIC(model)

```

```

Linear Regression

199 samples
  3 predictor

No pre-processing
Resampling: Cross-validated (10 fold)
Summary of sample sizes: 179, 180, 179, 178, 179, 179, ...
Resampling results:

    RMSE      Rsquared    MAE
0.7377916  0.2064194  0.6644843

Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-1.3330 -0.8483  0.3318  0.5969  1.1402

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.68451    3.72323   0.721  0.4718

```

```

Area          -0.02474    0.23208   -0.107    0.9152
Perimeter     -0.85368    0.69118   -1.235    0.2183
Kernel.Length  2.15125    0.69646    3.089    0.0023 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7418 on 195 degrees of freedom
Multiple R-squared:  0.1808,    Adjusted R-squared:  0.1682
F-statistic: 14.35 on 3 and 195 DF,  p-value: 1.746e-08
> AIC(model)
[1] 451.8383
> BIC(model)
[1] 467.2142

```

4. USING 2 PREDICTORS

```

#k-fold cross validation
set.seed(9)
# Define training control
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Type ~ Area + Perimeter, data = seeds, method = "lm", trControl =
train.control)

# Summarize the results

print(model)
summary(model)

model <- lm(Type ~ Area + Perimeter, data = seeds)

AIC(model)

BIC(model)

```

Linear Regression

199 samples
2 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 179, 180, 179, 178, 179, 179, ...

Resampling results:

RMSE	Rsquared	MAE
------	----------	-----

```

0.7546633 0.1675374 0.682637
Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-1.3797 -0.9223  0.3567  0.6175  1.0338

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.9371      3.1098  -1.266  0.20700
Area          -0.5080      0.1751  -2.901  0.00414 **
Perimeter      0.9257      0.3902   2.372  0.01864 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7578 on 196 degrees of freedom
Multiple R-squared:  0.1407, Adjusted R-squared:  0.132
F-statistic: 16.05 on 2 and 196 DF, p-value: 3.506e-07

> AIC(model)
[1] 459.3441
> BIC(model)
[1] 471.6448

```

5. USING 1 PREDICTOR

```

#k-fold cross validation
set.seed(9)
# Define training control
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Type ~ Area, data = seeds, method = "lm", trControl = train.control)

# Summarize the results
print(model)
summary(model)

model <- lm(Type ~ Area, data = seeds)

AIC(model)

BIC(model)

```

```

199 samples
 1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 179, 180, 179, 178, 179, 179, ...
Resampling results:

    RMSE      Rsquared   MAE
0.7634371  0.1428287  0.6985526

Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-1.3450 -0.9797  0.3657  0.6564  0.8581

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.41069    0.28363   12.025  < 2e-16 ***
Area        -0.09489    0.01866   -5.086  8.5e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7667 on 197 degrees of freedom
Multiple R-squared:  0.1161, Adjusted R-squared:  0.1116
F-statistic: 25.86 on 1 and 197 DF, p-value: 8.503e-07
> AIC(model)
[1] 462.9784
> BIC(model)
[1] 472.2039

```

Comparing AIC and BIC

No of Predictors	AIC	BIC	CP
1	462.9784	472.2039	-113.0867
2	459.3441	471.6448	-111.8577
3	451.8383	467.2142	-107.9311
4	416.6065	435.0576	-92.53254
5	281.2712	302.7975	2.86618

After checking the table above, we can say that AIC and BIC values are decreasing after adding predictor every time.

Values of CP should be closer to predictor's value to check the best model i.e. model with 5 predictors.

We can also include that the difference in the AIC, BIC and CP values are not that much. Hence, These are not the actual errors, just the training errors that have been modified every time to reflect the true test errors. We can calculate the MSE to check the errors in the model.

That means our model with 5 predictors is the best among others because lower the AIC and BIC values, better the model.

LOOCV for linear regression

As we know that our sub-model with 5 predictors i.e. Area, Perimeter, Kernel.length, Kernel.width, Kernel.groove is better so we will do the LOOCV on the same model. We will try to compare our results with the K-fold Cross validation.

```
# Define training control
train.control <- trainControl(method = "LOOCV")

# Train the model
model <- train(Type~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove,
data = seeds, method = "lm", trControl = train.control)

# Summarize the results
print(model)
summary(model)
```

Output:

Linear Regression

199 samples
5 predictor

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 198, 198, 198, 198, 198, 198, ...

Resampling results:

RMSE	Rsquared	MAE
0.4866294	0.6404739	0.3770358

Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:

```
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-1.56676 -0.28301  0.04416  0.28641  0.98844

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   14.9013     4.1648   3.578 0.000438 ***
Area           0.6288     0.2637   2.385 0.018045 *
Perimeter     -1.2749     0.4852  -2.627 0.009298 **
Kernel.Length -2.4033     0.5417  -4.437 1.53e-05 ***
Kernel.Width  -1.9135     0.7806  -2.451 0.015120 *
Kernel.Groove  2.9759     0.2149  13.850 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4809 on 193 degrees of freedom
Multiple R-squared:  0.6593, Adjusted R-squared:  0.6504
F-statistic: 74.69 on 5 and 193 DF, p-value: < 2.2e-16
```

Now, If we compare the results with K-fold cross validation, we can see that there is not much difference between the p-values, F-statistics or Adjusted R square. Only there is smallest change in the values of of RMSE and R square. If we check in LOOCV, RMSE and R square are 0.4866294, and 0.6404739 whereas in K-fold, values are 0.4760658 and 0.6565677 respectively.

R Square- Higher the better- (K-fold R square is higher than LOOCV)

RMSE- Lower the better- (K-fold RMSE is lower than LOOCV)

We do not have enough predictions to check the performance of both the models but K-fold model is better than LOOCV because it validates the performance on multiple folds of our data and gives the more stable answer as to how our model performs.

Repeated K-fold CV

```
#Repeated k-fold
# Define training control
train.control <- trainControl(method = "repeatedcv",
                             number = 10, repeats = 3)
# Train the model
model <- train(Type~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove,
data = seeds, method = "lm",trControl = train.control)

# Summarize the results

print(model)

summary(model)
```

Linear Regression

199 samples
5 predictor

No pre-processing

Resampling: Cross-validated (10 fold, repeated 3 times)

Summary of sample sizes: 178, 178, 179, 180, 179, 180, ...

Resampling results:

RMSE	Rsquared	MAE
0.481578	0.6507801	0.3774992

Tuning parameter 'intercept' was held constant at a value of TRUE
> summary(model)

Call:

```
lm(formula = .outcome ~ ., data = dat)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.56676	-0.28301	0.04416	0.28641	0.98844

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	14.9013	4.1648	3.578	0.000438	***
Area	0.6288	0.2637	2.385	0.018045	*
Perimeter	-1.2749	0.4852	-2.627	0.009298	**
Kernel.Length	-2.4033	0.5417	-4.437	1.53e-05	***
Kernel.Width	-1.9135	0.7806	-2.451	0.015120	*
Kernel.Groove	2.9759	0.2149	13.850	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4809 on 193 degrees of freedom

Multiple R-squared: 0.6593, Adjusted R-squared: 0.6504

F-statistic: 74.69 on 5 and 193 DF, p-value: < 2.2e-16

Extract MSE in LOOCV and K-Fold

```
#Build General Linear Regression model
```

```
model1 <- glm(Type~ Area + Perimeter + Kernel.Length + Kernel.Width  
+Kernel.Groove,data=train.data)
```

```
summary(model1)
```

```
# Run 10-fold cross-validation of the model1 on the data set
```

```
cv.model = cv.glm(data=train.data, glmfit=model1, K=10)
```



```
# Extract the MSE
cat('10-fold cross-validation MSE:', cv.model$delta[1])
10-fold cross-validation MSE: 0.2456297
```

```
# Run LOOCV
loocv.model = cv.glm(data=train.data, glmfit=model1, K=nrow(train.data))
# Extract the MSE
cat('\nLeave-one-out-cross-validation MSE:', loocv.model$delta[1])
Leave-one-out-cross-validation MSE: 0.2430991
```

Mean squared error shows the average squared difference between the estimated and test values. Now our MSE in LOOCV is 0.2430 and K-fold is 0.2456. MSE values are closer to zero that means both model are better.

Linear Discriminant Analysis

```
# Split the data into training (80%) and test set (20%)
set.seed(123)
training.samples <- seeds$Type %>%
createDataPartition(p = 0.8, list = FALSE)
train.data <- seeds[training.samples, ]
test.data <- seeds[-training.samples, ]
# Fit the model
model <- lda(Type~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove,
data = train.data)
# Make predictions
predictions <- model %>% predict(test.data)
# Model accuracy
mean(predictions$class==test.data$Type)
model <- lda(Type~ Area + Perimeter + Kernel.Length + Kernel.Width + Kernel.Groove,
data = train.data)
model
```

```

Call:
lda(Type ~ Area + Perimeter + Kernel.Length + Kernel.Width +
     Kernel.Groove, data = train.data)

Prior probabilities of groups:
      1      2      3
0.34375 0.33125 0.32500

Group means:
      Area Perimeter Kernel.Length Kernel.Width Kernel.Groove
1 14.33582  14.29273    5.507800    3.246800    5.085891
2 18.25906  16.10604    6.129962    3.671132    6.000075
3 11.87731  13.25096    5.238000    2.852135    5.131154

Coefficients of linear discriminants:
              LD1      LD2
Area          -0.2309424  1.733959
Perimeter      3.9080962 -2.678184
Kernel.Length -7.1614900 -7.337262
Kernel.Width  -0.6406629 -6.052518
Kernel.Groove  3.2083770  8.043019

Proportion of trace:
      LD1      LD2
0.7541 0.2459

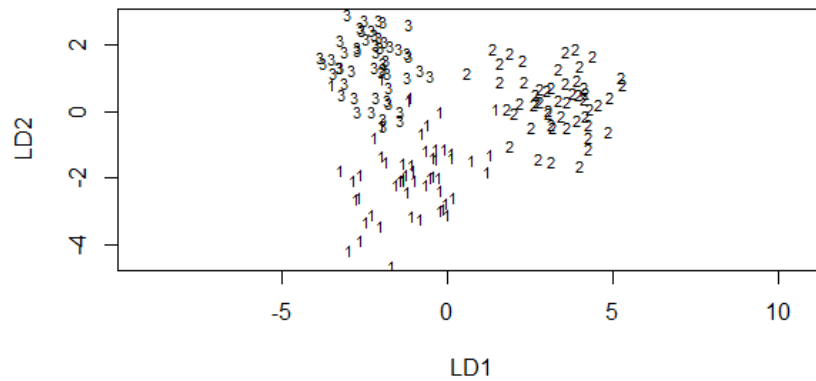
```

Our LDA model is 97% accurate according to our data. Linear Discriminant Analysis provide the Group mean, probabilities and coefficient for each individual parameter.

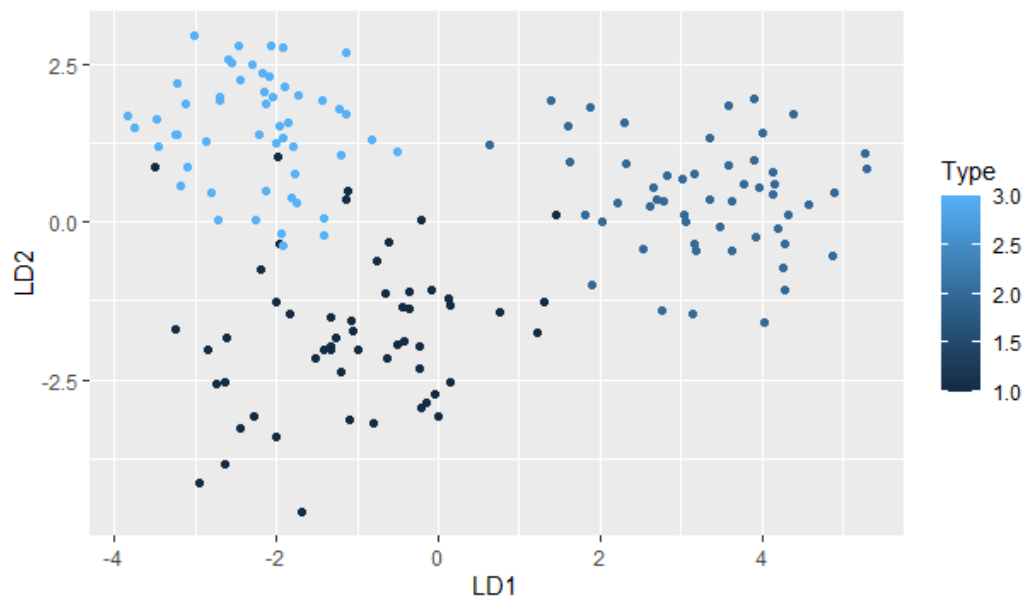
1. Probabilities: We can see that there are 34.3%, 33.1%, 32.5% of Type 1, Type 2 and Type 3 in the Group.
2. Group Mean: It shows the group center of gravity of all the variables.
3. Coefficients: It provide us the combination of all the variable that are required to form the LDA decision. For eg: $LD1 = \text{Area} \times -0.2309424 + \text{Perimeter} \times 3.9080962 + \text{Kernel.Length} \times -7.1614900 + \text{Kernel.Width} \times -0.6406629 + \text{Kernel.Groove} \times 3.2083770$ and $LDA2 = \text{Area} \times 1.733959 + \text{Perimeter} \times -2.678184 + \text{Kernel.Length} \times -7.337262 + \text{Kernel.Width} \times -6.052518 + \text{Kernel.Groove} \times 8.043019$

LDA Plot

Plot(model)



```
lda.data <- cbind(train.data, predict(model)$x)
ggplot(lda.data, aes(LD1, LD2)) +
  geom_point(aes(color = Type))
```



In above graph, You can see the values of Type1, 2 and 3. LDA is used to reduce the 2-D graph to 1-D in order to maximize the separability between the two classes. LDA uses two following criteria to create a new axis:

1. Maximize the distance between means of the three classes.
2. Minimize the variation.

K-Fold Cross Validation for Linear Discriminant Analysis

1. Using 1 Predictor

```
#k-fold for linear Discriminant Analysis
set.seed(9)
# Define training control
control1 <- trainControl(method="cv", number=10)
seeds$Type <- as.character(seeds$Type)
#Train the model
ldamodel1 <- train(Type ~Area, data=seeds, method="lda", trControl=control1)
#Summarize
print(ldamodel1)
```

```
Linear Discriminant Analysis
199 samples
  1 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 178, 181, 179, 178, 179, 179, ...
Resampling results:

    Accuracy    Kappa
0.8751378  0.8128222
```

2. Using 2 predictor

```
#k-fold for linear Discriminant Analysis (predictor=2)
set.seed(9)
# Define training control
control1 <- trainControl(method="cv", number=10)
seeds$Type <- as.character(seeds$Type)
```

```
#Train the model
ldamodel2 <- train(Type ~Area+Perimeter, data=seeds, method="lda",
trControl=control1)

#Summarize
print(ldamodel2)
```

Linear Discriminant Analysis

```
199 samples
  2 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 178, 181, 179, 178, 179, 179, ...
Resampling results:

    Accuracy    Kappa
0.8848705  0.8275308
```

3. Using 3 Predictor

```
#k-fold for linear Discriminant Analysis (predictor=3)
set.seed(9)

# Define training control
control1 <- trainControl(method="cv", number=10)
seeds$Type <- as.character(seeds$Type)

#Train the model
ldamodel3 <- train(Type ~Area+Perimeter+Kernel.Length, data=seeds, method="lda",
trControl=control1)

#Summarize
print(ldamodel3)
```

Linear Discriminant Analysis

```
199 samples
  3 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 178, 181, 179, 178, 179, 179, ...
Resampling results:

    Accuracy    Kappa
0.8793442  0.8195419
```

4. Using 4 predictor

```
#k-fold for linear Discriminant Analysis(predictor=4)
set.seed(9)
# Define training control
control1 <- trainControl(method="cv", number=10)
seeds$Type <- as.character(seeds$Type)
#Train the model
ldamodel4 <- train(Type ~Area+Perimeter +Kernel.Length+Kernel.Width, data=seeds,
method="lda", trControl=control1)
#Summarize
print(ldamodel4)
```

```
Linear Discriminant Analysis

199 samples
  4 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 178, 181, 179, 178, 179, 179, ...
Resampling results:

    Accuracy    Kappa
0.8844236  0.8269536
```

5. Using 5 predictor

```
#k-fold for linear Discriminant Analysis (predictor=5)
```

```

set.seed(9)

# Define training control

control1 <- trainControl(method="cv", number=10)

seeds$Type <- as.character(seeds$Type)

#Train the model

ldamodel5 <- train(Type ~Area+Perimeter +Kernel.Length+Kernel.Width
+Kernel.Groove, data=seeds, method="lda", trControl=control1)

#Summarize

print(ldamodel5)

```

Linear Discriminant Analysis

```

199 samples
  5 predictor
  3 classes: '1', '2', '3'

```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 178, 181, 179, 178, 179, 179, ...

Resampling results:

Accuracy	Kappa
0.9546867	0.9322384

No of Predictors	Kappa	Accuracy
1	0.8128222	0.8751378
2	0.8275308	0.8848705
3	0.8195419	0.8793442
4	0.8269536	0.8844236
5	0.9322384	0.9546867

Now, we can see that Accuracy and Kappa of the model is increasing by adding more variables. Kappa is generally used to measure the score of homogeneity or consensus exists in the ratings. It records the possibility of the agreement occurring by chance. From 0.81-

1.0 gives the most perfect agreement. Though our values of Kappa are more than .80 but highest in 5 predictors. Also, Our accuracy of the model is 95% using 5 predictors so we can say that our model with 5 predictors is best among others.

LOOCV for Linear Discriminant Analysis

```
#LOOCV FOR LDA
set.seed(9)
# Define training control
control1 <- trainControl(method="LOOCV", number=10)
seeds$Type <- as.character(seeds$Type)
#Train the model
ldamodel5 <- train(Type ~Area+Perimeter +Kernel.Length+Kernel.Width
+Kernel.Groove, data=seeds, method="lda", trControl=control1)
#Summarize
print(ldamodel5)
```

```
Linear Discriminant Analysis

199 samples
 5 predictor
 3 classes: '1', '2', '3'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 198, 198, 198, 198, 198, 198, ...
Resampling results:

   Accuracy   Kappa
0.959799    0.9396855
```

I have selected the model with 5 predictors and performed the LOOCV. After comparing the results with K-fold CV, I can say that the results are almost similar with the 5 predictor model. Hence, both the model are giving the accuracy of 95%.

Repeated K-fold CV for LDA

```
#Repeated CV FOR LDA

set.seed(9)

# Define training control

control1 <- trainControl(method="repeatedcv", number=10, repeats=3)

#Train the model

ldamodel5 <- train(Type ~Area+Perimeter +Kernel.Length+Kernel.Width
+Kernel.Groove, data=seeds, method="lda", trControl=control1)

#Summarize

print(ldamodel5)
```

Linear Discriminant Analysis

```
199 samples
  5 predictor
  3 classes: '1', '2', '3'
```

No pre-processing

Resampling: Cross-validated (10 fold, repeated 3 times)

Summary of sample sizes: 181, 180, 178, 179, 179, 179, ...

Resampling results:

Accuracy	Kappa
0.9636257	0.9452175

After comparing the 5 predictor k-fold model, LOOCV and repeated CV, I can say that accuracy of Repeated CV is more than other models. Other models are giving the 95% of accuracy whereas RepeatedCV is proving the 96% of accuracy of model. Kappa coefficient value is also more than 1%.

Conclusion: In this assignment, we built the Linear regression model and Linear Discriminant analysis. We have understood the dataset and checked the correlation of all the predictors. 5 predictors i.e. Area, Perimeter, Kernel.Length, Kernel.Width and Kernel.Groove showed the high correlation with each other.

We performed the K-fold CV on both the models (LDA and LM) by adding new variable each time and calculated the test errors and accuracy of model. We also calculated the AIC, BIC and Cp values of all the models. Hence, we can say that CP, AIC and BIC are just training errors, they are not the actual errors in the model.

Accuracy of the model is higher when we use the 5 predictors whereas lowest with 1 predictor.