Feature Selection based on Correlation and P-Value

MATH 6358

Group 8

December 5, 2020

Abstract:

Often when we get a dataset, we might find a plethora of features in the dataset. All of the features we find in the dataset might not be useful in building a statistical model to make the necessary prediction. Using some of the features might even make the predictions worse. So, feature selection plays a huge role in building a statistical model. In this report, we will explore two measures: Correlation and P-value that we can use on the data to select the right features.

The two most commonly used statistical tests for establishing relationships between variables are correlation and p-value. Correlation is a way to test if two variables have any kind of relationship, whereas the p-value tells us if the result of an experiment is statistically significant. In this report, we will be taking a look at how they are calculated and how to interpret the numbers obtained.

Table of Contents

Research Statement: Testing the Significance of the Correlation Coefficient and P-value in feature selection

Introduction:

In this report, we will use feature selection to identify features that are most relevant to our predictive modeling problem. It is possible that too much data can harm the accuracy of the model, leading to an error in prediction; therefore, it is important to identify the most important attributes of the data. Particularly, we will be looking at Correlations and P-value to select the most impactful features. Correlation defines the relationship between features, in this case, we will use it to see how linearly dependent each feature is with one another. If the correlation between features is high then we can eliminate one of the features as this won't affect the data differently. In other words, it won't change the result of whether the mass is Benign or Malignant cancer. We will also explore feature selection based on p-values (the probability value) for the model by using Backward Elimination. Backward Elimination is used to keep only the features that are significant to the dataset. We will be comparing the p-value of each feature and if it's bigger than the significance level of .05 then it will be removed, and the cycle repeats until the only features left are the ones that cause the most amount of change to the dependent variable. We will also explore model building using K-mean clustering and K-Nearest Neighbor (KNN) algorithm to select features with the highest accuracy.

Summary:

In this project, we will use data collected from the Breast Cancer Wisconsin (Diagnostic) Data set. This data set can be found on the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

It uses many features to predict whether the cancer is Benign (B) or Malignant (M). The data set consists of 18,208 observations and 32 attributes. These attributes consist of ten-valued features for each cell nucleus:

a) Radius (mean of distances from the center to points on the perimeter)
b) Texture (standard deviation of gray-scale values)
c) Perimeter
d) Area
e) Smoothness (local variation in radius lengths)
f) Compactness (perimeter^2 / area - 1.0)
g) Concavity (severity of concave portions of the contour)
h) Concave points (number of concave portions of the contour)
i) Symmetry
j) Fractal dimension ("coastline approximation" - 1)

For each of these ten values, the mean, standard error (se), and worst (mean of the largest values) were computed from each digitized image of a fine needle aspirate of a breast mass. Therefore, there are a total of 30 features in this data set. (there is another feature within the data: ID number, we are not concerned about this feature and have it dropped from our analysis.)

Analysis:

When analyzing any data, it is important to identify which variables/ features are the most impacting on the response variable. By performing Feature Selection, we can identify these features. We select the variables that contribute the most to the response variable as well as help improve our predictions of the model by eliminating any variables that are irrelevant or decrease the accuracy.

Selecting features based on the correlation

The first step in our feature selection is based on data correlation. Taking the correlation of the data is the most common and important first step to understand how strong pairs of variables are between each other. Depending on a simple linear model: Y (response variable) vs. X (predictor variable) we know the value of the correlation coefficient can tell us a lot about the data. Depending on the sign, we can tell if it's a positive or negative relationship and depending on how close it is to 1 (or -1) we know how strong the linear relationship is. Where 1 is a very strong linear relationship and 0 being there might be a relationship but not a linear one. Now that we understand what correlation means to a simple linear model, we need to understand what it means for a multiple feature model such as the 30 features in our data: Breast Cancer Wisconsin. In our project, we are trying to identify which features have the largest impact on our categorical response variable: whether the mass is Benign or Malignant cancer. To do this, we observe the correlation between the 30 features, and by doing so we might run into multicollinearity.

Multicollinearity occurs when one independent variable is highly correlated with one or more other independent variables in multiple regression. In other words, having two or more of the predictors exhibit moderately or high correlation. This is not what we hope to have since it can harm the p-value of the data, by resulting in unstable estimates as it increases the variance of regression coefficients. Also, if two independent variables contain the same information to an extent, one variable gains little by using both in the regression model. As we will see later in the report, we can identify multicollinearity heat maps and Variance Inflation Factor (or VIF). Therefore, the first step into understanding our data, we will take a look at our correlation matrix.

Identifying multicollinearity using correlations and heatmaps

Since our data is too large to display the correlation matrix of the diagnoses and all 30 features, we have displayed "diagnosis" along with the first 4 features below.

```
> data.corr[1:5,1:5]
              diagnosis radius_mean texture_mean perimeter_mean area_mean
diagnosis         1.000       0.730        0.415          0.743     0.709
radius_mean       0.730       1.000        0.324          0.998     0.987
texture_mean      0.415       0.324        1.000          0.330     0.321
perimeter_mean    0.743       0.998        0.330          1.000     0.987
area_mean         0.709       0.987        0.321          0.987     1.000
```

Even though this is a small fraction of our data, from this correlation matrix we can already spot highly correlated values (as highlighted) between ***radius_mean*** vs ***perimeter_mean*** and ***area_mean***, as well as ***perimeter_mean*** vs ***area_mean***.

Using the ggpairs() function in the R package ggplot2, we are able to display a scatter plot matrix, where our correlations are shown in the top right triangle, and numeric variables are drawn on the bottom left half of the triangle. Once again we take note of the highlighted features to demonstrate a high correlation and positively strong linear relationship between the two.

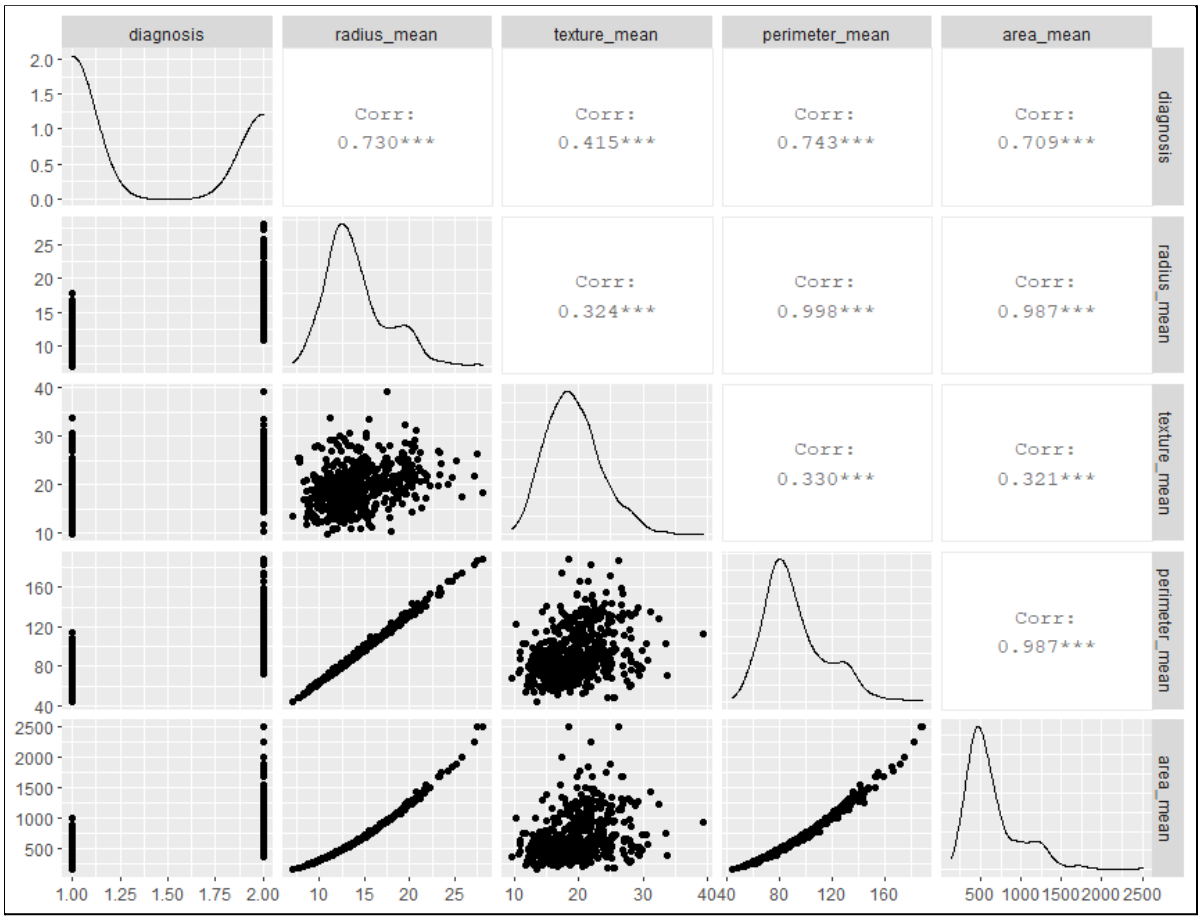


Figure 1: Scatter Plot for Visualizing Correlation between Features

Moving on, we can use the function corrplot() in the R package corrplot to generate a heatmap. By doing so we can view the data in a visually colorful way and identify possible multicollinearity. The color in the plot varies based on the sign/ direction of the correlation: blue for positive and red for negative signs. While the size of the circle demonstrated the strength of the correlation: the larger the circle, the closer it is to 1 (or -1).

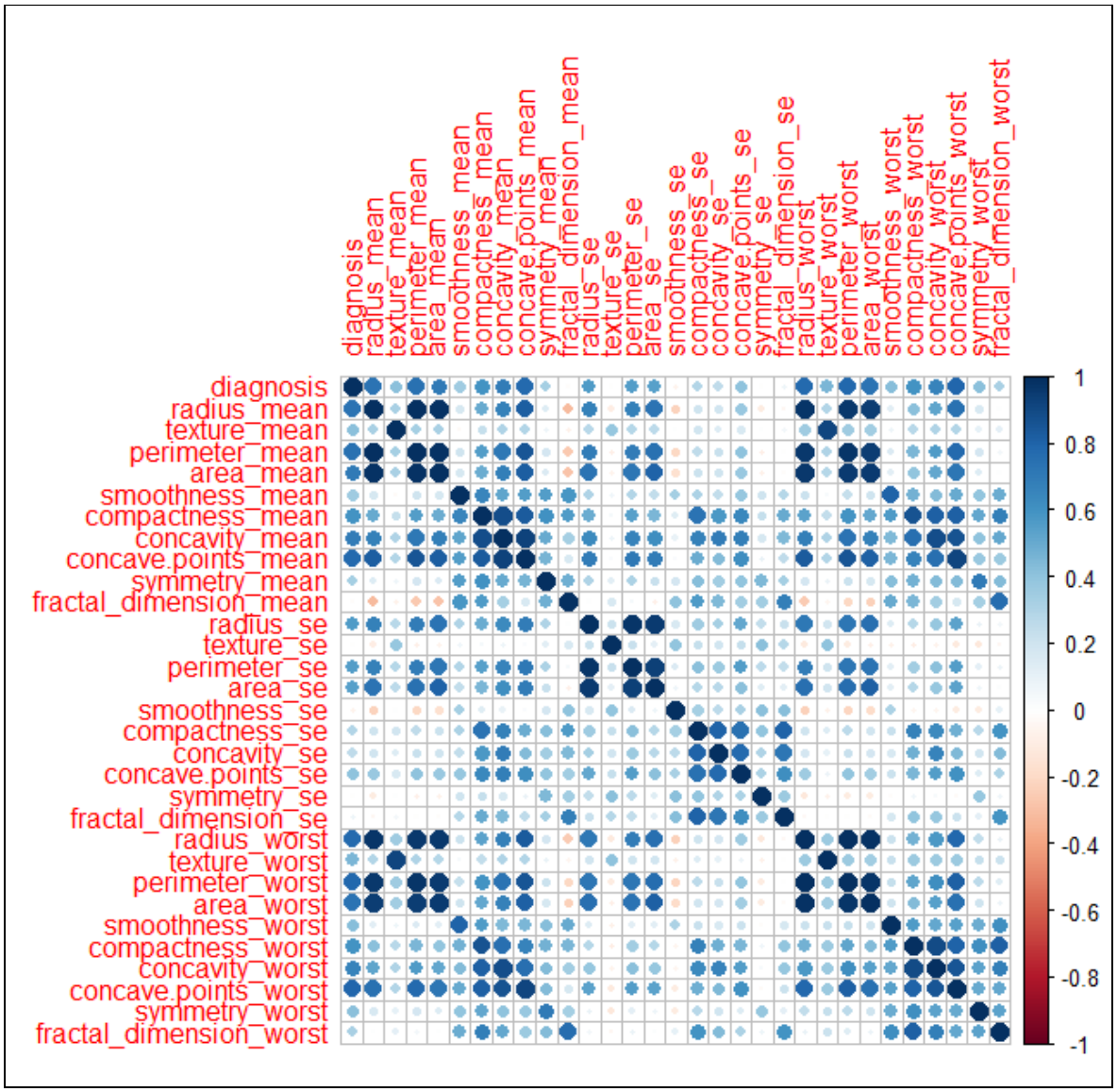


***Figure 2: Correlation Heatmap** showing a 2D Correlation Matrix between two of 30 Features*

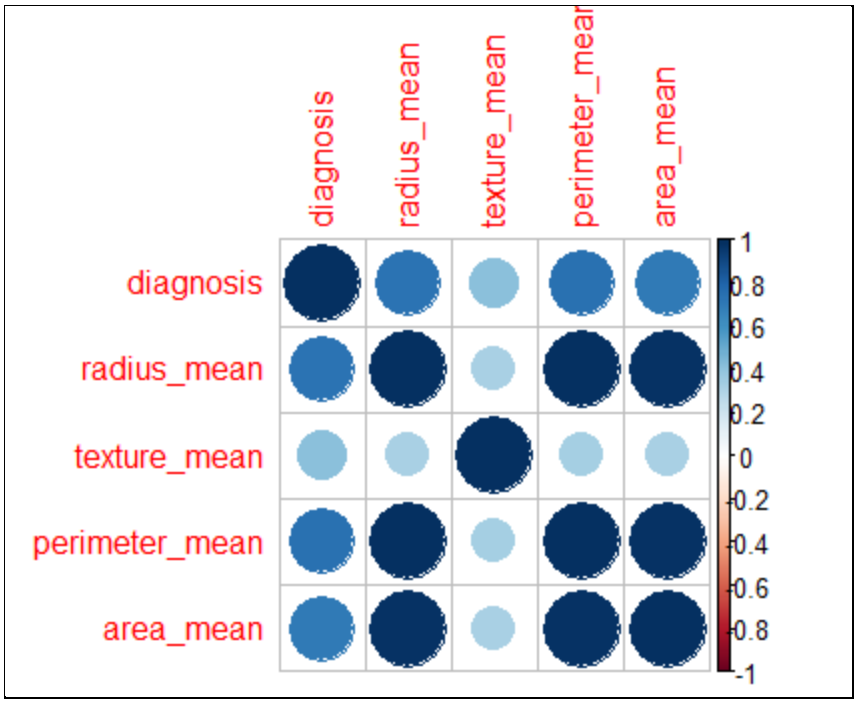


Figure 3: Corrplot, Correlation Heatmap of the first 4 features and response "Diagnosis"

This time we can view all 30 of the features. Since we appear to have little to no red circles we can move on observing the large dark blue circles, since those are the ones that are highly correlated. Just by eyeballing each row feature, we can see an average of 10 features with highly correlated values, therefore predicting to eliminate those features.
Just to name 8 from the first row:

radius_mean,
perimeter_mean,
area_mean,
concave.poins_mean,
radius_worst,
perimeter_worst,
area_worst, and
concave.points_worst

Of course, eyeballing the data is just a curious way to observe the data and not our final answer nor a smart way to select feature selection. As we can see in the next part, some of the features we suggested to eliminate ended up staying.

Next, we compare the correlation between features and remove one of the two features that have a correlation higher than 0.90. Including the response **Diagnosis**, we can see that only 20 column features were selected while 10 were eliminated. Therefore our eyeballing the data was not that off.

We ended up eliminating 3 of the predictors we believed earlier to have the greatest influence

{***radius_mean, perimeter_mean, area_mean***},

And 7 more that we did not expect

{texture_mean, compactness_mean, concave.points_mean, facial_dimension_mean, texture_se, facial_dimension_se, texture_worst}

```
> dim(data.new)
[1] 569  21
> names(data.new)
 [1] "diagnosis"               "smoothness_mean"
 [3] "compactness_mean"        "symmetry_mean"
 [5] "fractal_dimension_mean"  "texture_se"
 [7] "area_se"                 "smoothness_se"
 [9] "compactness_se"          "concavity_se"
[11] "concave.points_se"       "symmetry_se"
[13] "fractal_dimension_se"    "texture_worst"
[15] "area_worst"              "smoothness_worst"
[17] "compactness_worst"       "concavity_worst"
[19] "concave.points_worst"    "symmetry_worst"
[21] "fractal_dimension_worst"
```

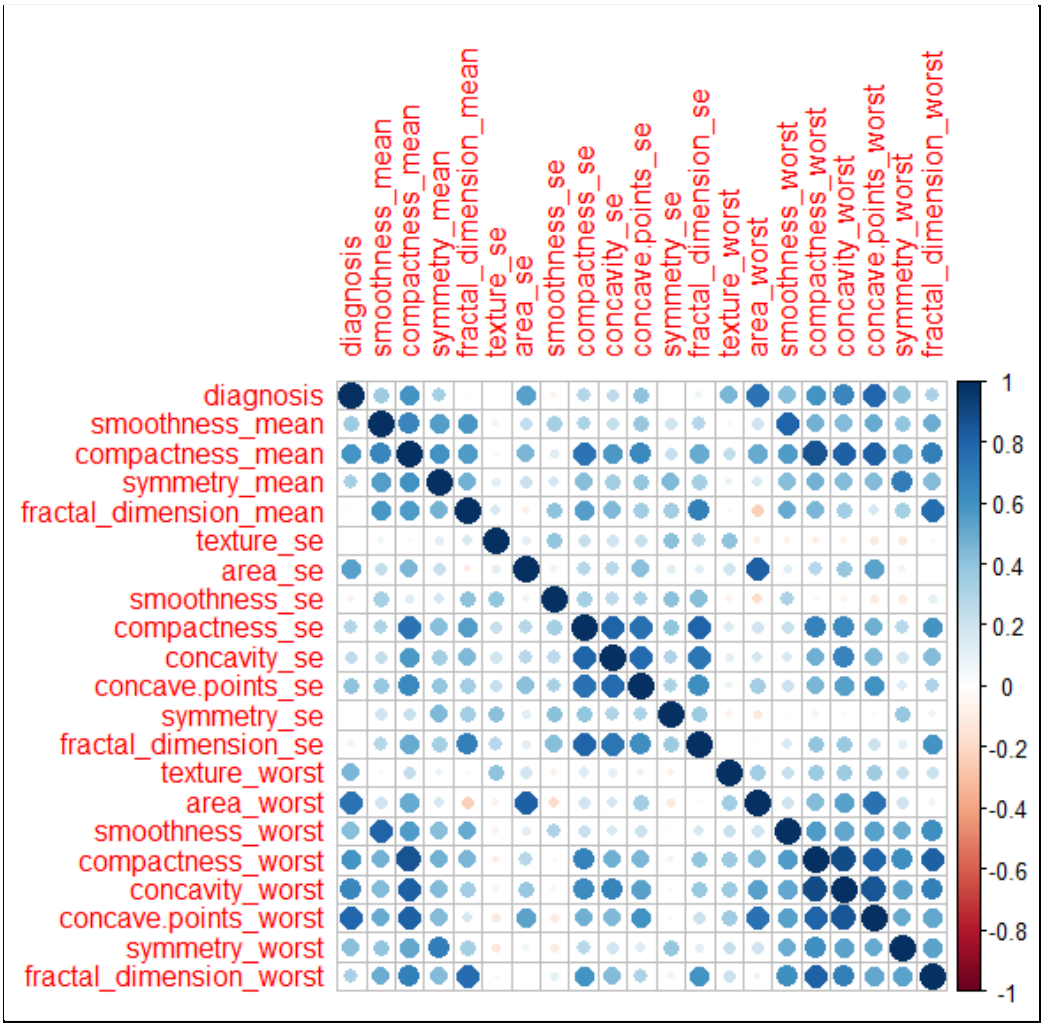


Figure 4: Correlation Matrix of 20 Features with the Response Variable

As we recall, highly correlated features are more linearly dependent; therefore, they can result in having the same effect on dependent variables. By setting any two features with a correlation higher than 0.90, we are able to drop one of the two features. By doing this we hope to have removed irrelevant or redundant attributes from the data, thus improving the accuracy of the predictive model.

Identifying multicollinearity using VIF

Another method we can use to identify multicollinearity is using the Variance Inflation Factor (VIF). This method assesses how much the variance of an estimated regression coefficient increases if features are correlated. If the VIF is 1, then there is low to no correlation and the larger the features are correlated, the larger the VIF. Using the linear model function lm(Y~. , data = data) in R, we can carry out a multiple regression model where Y is our response variable "diagnosis" vs our full model of 30 features. We also use vfi() under the package car to find the VIF. Setting the threshold to 10, we can observe if any variables go past that threshold to be removed from the model. In Figure 5, we can see the VIF values for our full model and a bar graph with threshold 10. As you can see the ***radius_mean*** and ***perimeter_mean*** have unbelievably large VIF values close to 4000. Because of these two features, and what appears to be 4 others, we can barely see our threshold of 10. Now we are certain we have a presence of multicollinearity and dropping a few features can help improve our data.

```
vif_data
          radius_mean            texture_mean
          3806.115296               11.884048
       perimeter_mean               area_mean
          3786.400419              347.878657
      smoothness_mean         compactness_mean
             8.194282               50.505168
       concavity_mean     concave.points_mean
            70.767720               60.041733
        symmetry_mean  fractal_dimension_mean
             4.220656               15.756977
           radius_se              texture_se
            75.462027                4.205423
         perimeter_se                area_se
            70.359695               41.163091
        smoothness_se          compactness_se
             4.027923               15.366324
         concavity_se       concave.points_se
            15.694833               11.520796
        symmetry_se     fractal_dimension_se
             5.175426                9.717987
        radius_worst           texture_worst
           799.105946               18.569966
      perimeter_worst              area_worst
           405.023336              337.221924
     smoothness_worst        compactness_worst
            10.923061               36.982755
      concavity_worst    concave.points_worst
            31.970723               36.763714
       symmetry_worst fractal_dimension_worst
             9.520570               18.861533
```

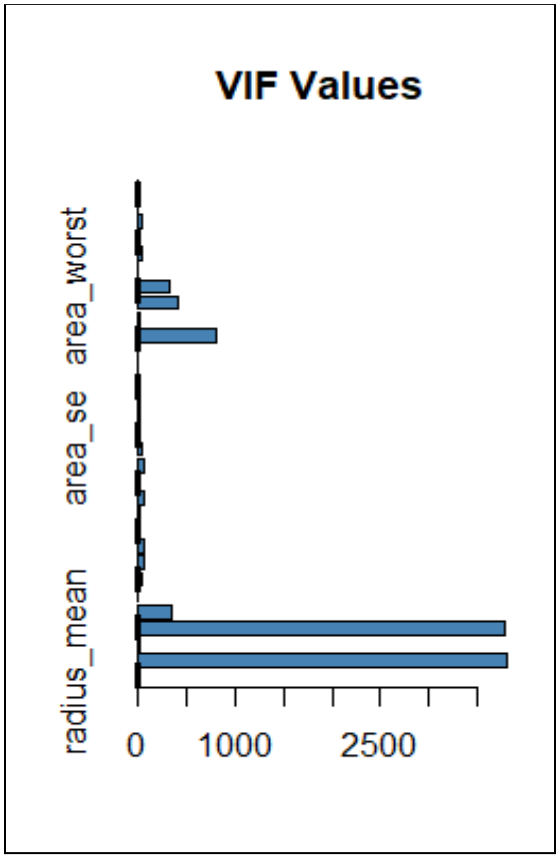


Figure 5: VIF Bar Graph for Full Model

Moving forward, we can test the VIF once again using our *data.new* of 20 features we obtained from earlier. Setting up another linear model we observe the VIF values in Figure 6 below. Right away we see an improvement of the VIF value for the features we kept. Compactness_mean showing the most drop (50.5 to 19.15) of 31.35 points to faractal_dimension_worst showing the least

drop (18.86 to 17.96) of 0.9 points. Even though our VIF has improved we still appear to have some multicollinearity since 8 features still appear to pass the 10 level threshold. Therefore, we hope to improve our model as we explore feature selection based on p-value in the next section of this report.

```
vif_data.new
      smoothness_mean        compactness_mean
             6.732140               19.153895
       symmetry_mean  fractal_dimension_mean
             3.892434               10.588979
        texture_se                   area_se
             2.945187                4.671350
      smoothness_se          compactness_se
             3.538297               14.445720
       concavity_se       concave.points_se
            11.036448                8.245383
        symmetry_se    fractal_dimension_se
             4.493753                8.707923
      texture_worst              area_worst
             2.675907                9.860048
    smoothness_worst       compactness_worst
             9.346221               30.627717
     concavity_worst    concave.points_worst
            20.306160               21.199341
      symmetry_worst fractal_dimension_worst
             8.246714               17.967631
```

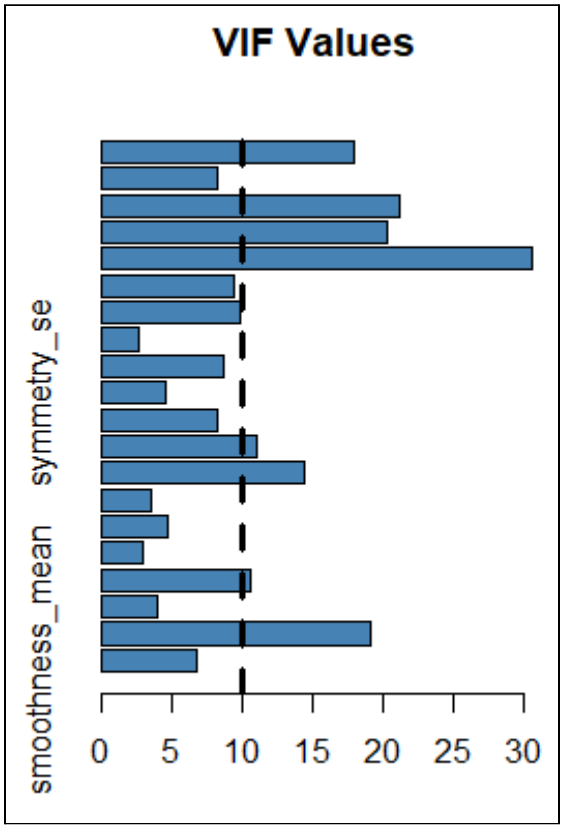


Figure 6: VIF Bar Graph of 20 Features

Selecting columns based on p-value using backward elimination

In this section, we will be selecting the columns based on their p-value (goes from 0 to 1) which refers to the hypothesis of the significance level. The significance level is the amount of change a feature will affect the final output, meaning how important is this feature and how much it affects the response variable. In this case, generally, we take the 5% (or 0.05) significance level by default. Using the linear model function lm(Y~. , data = data) in R, we can carry out a multiple regression model where Y is our response variable ***Diagnosis*** vs our full model of 30 features. Using the summary() function we observe the following information:

```
> data.lm = lm(diagnosis~.,data=data)
> summary(data.lm)

Call:
lm(formula = diagnosis ~ ., data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60241 -0.16209 -0.02705  0.12982  0.82295

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             -1.0218117  0.4280072  -2.387 0.017314 *
radius_mean             -0.2177721  0.1735089  -1.255 0.209985
texture_mean             0.0045455  0.0079439   0.572 0.567426
perimeter_mean           0.0237399  0.0250985   0.946 0.344641
area_mean                0.0003178  0.0005253   0.605 0.545391
smoothness_mean          0.0846891  2.0172759   0.042 0.966529
compactness_mean        -4.2220353  1.3336803  -3.166 0.001635 **
concavity_mean           1.3979973  1.0458611   1.337 0.181887
concave.points_mean      2.1418330  1.9791819   1.082 0.279657
symmetry_mean            0.1027092  0.7427382   0.138 0.890067
fractal_dimension_mean   0.0332616  5.5722693   0.006 0.995240
radius_se                0.4349559  0.3104682   1.401 0.161800
texture_se              -0.0067585  0.0368439  -0.183 0.854525
perimeter_se            -0.0225203  0.0411183  -0.548 0.584129
area_se                 -0.0009232  0.0013978  -0.660 0.509235
smoothness_se           15.8543207  6.6248737   2.393 0.017046 *
compactness_se           0.0649034  2.1694829   0.030 0.976145
concavity_se            -3.5654680  1.3007522  -2.741 0.006327 **
concave.points_se       10.5679513  5.4520354   1.938 0.053103 .
symmetry_se              1.6973407  2.7276006   0.622 0.534019
fractal_dimension_se    -7.1464402 11.6764096  -0.612 0.540769
radius_worst             0.1951831  0.0579677   3.367 0.000814 ***
texture_worst            0.0071594  0.0069489   1.030 0.303339
perimeter_worst         -0.0024351  0.0059360  -0.410 0.681807
area_worst              -0.0010112  0.0003197  -3.163 0.001648 **
smoothness_worst         0.5428569  1.4346393   0.378 0.705288
compactness_worst        0.0671583  0.3830830   0.175 0.860902
concavity_worst          0.3811912  0.2686173   1.419 0.156453
concave.points_worst     0.4643099  0.9142253   0.508 0.611751
symmetry_worst           0.5567875  0.4943014   1.126 0.260493
fractal_dimension_worst  4.3034831  2.3832134   1.806 0.071517 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2362 on 538 degrees of freedom
Multiple R-squared:  0.7743,    Adjusted R-squared:  0.7617
F-statistic: 61.53 on 30 and 538 DF,  p-value: < 2.2e-16
```

Table 1: Summary of the linear model for the full data model

From our summary in Table 1, we observe that the feature ***fractal_dimension_mean*** has a p-value of 0.995, concluding that the null hypothesis is true: this column does not provide any noticeable change to the dependent variable and can be easily removed without consequences.

The steps we have just done is based on the **Backward Elimination** method to keep only those features that are significant to the dataset, which we will apply for the rest of the features:

1. Select a significance level (in this case, we choose 0.05)
2. Fit the model with all features.
3. Check the p-values of different features with summary() function.
4. If the p-value is higher than the significance level, remove the feature.
5. Repeat steps 2 to 4 with the reduced features till only the features having p-values ≤ significance level remain.

As a final result, we obtain the following summary table:

```
> data_new.lm = lm(diagnosis~.,data=data_new)
> summary(data_new.lm)

Call:
lm(formula = diagnosis ~ ., data = data_new)

Residuals:
     Min       1Q   Median       3Q      Max
-0.90672 -0.17681 -0.03524  0.13964  0.91517

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              0.143338   0.262257   0.547  0.58490
radius_mean              0.036135   0.008051   4.488 8.74e-06 ***
texture_mean             0.016296   0.002689   6.061 2.49e-09 ***
concavity_mean           1.722750   0.431816   3.990 7.50e-05 ***
fractal_dimension_mean -19.381427   3.516659  -5.511 5.45e-08 ***
radius_se                0.198241   0.060363   3.284  0.00109 **
compactness_se          -3.554004   1.275953  -2.785  0.00553 **
concavity_se            -2.642525   0.806482  -3.277  0.00112 **
concave.points_se       16.757584   3.326379   5.038 6.37e-07 ***
smoothness_worst         2.994482   0.704780   4.249 2.52e-05 ***
symmetry_worst           1.069151   0.215731   4.956 9.56e-07 ***
fractal_dimension_worst  7.848703   1.278089   6.141 1.56e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.251 on 557 degrees of freedom
Multiple R-squared:  0.7361,    Adjusted R-squared:  0.7309
F-statistic: 141.2 on 11 and 557 DF,  p-value: < 2.2e-16
```

Table 2: Summary of the linear model for the full data_new model

We observe that when we reduce the number of features from 30 features to the most significant 11 features with their p-values all below significance level 0.05, which highly impact

our response variable ***Diagnosis.*** Notice that some of the 11 features found to be most significant using the Backward Elimination are the same features that were dropped in our > 0.90 correlation selection test, such as the ***radius_mean, texture_mean,*** and ***facial_dimension_mean.*** This shows that based on these two feature selection methods, we are able to reduce the number of features down to the most important; however the results can be contradictory. In our correlation test we only performed once with a set threshold of **0.90**. While with the p-value method, we repeated our steps by removing each feature that shows a value above the assumed **0.05**. For both methods we received results that are not quite similar with each other. Therefore. we wouldn't expect using only Correlation and P-value to be the best methods in feature selection.

It is important to understand that removing or even keeping a selected number of features can have a big impact on the outcome of data prediction. That is why it is also important to test the data using models or algorithms that can further improve predicting, categorizing, or classifying our data.

Model Building

In order to check how much our selecting features makes an impact on model performance in the dataset, we use the K-Means model (an unsupervised learning model) and KNN model (a supervised learning model) to predict the results.

First of all, we partition our data set to a training set and a test set. 20% of the data set is used to create the TESTSET and 80% to create the TRAINSET. Next, we apply the K-Means model and KNN model with both our 30 original features and our 11 selected features.

K-Means Model:

As a definition, K-Means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as far as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) data points are within the same cluster.

In our case, at first, we apply K-Means Model to our original dataset with 30 features to predict our response variable ***Diagnosis*** which yields the below results with the confusion matrix and the accuracy:

```
> cm_kmeans

pre_kmeans  Benign Malignant
  Benign         1        28
  Malignant    119        23
> sum(diag(cm_kmeans))/sum(cm_kmeans)
[1] 0.1403509
```

As our observation, the K-Means Model with original 30 features only has an accuracy of ~14% which is pretty low. In the real scenarios, we most likely would not apply this model to predict the results based on its fairly poor model.

As a comparison, we apply K-Means Model to the dataset with only our 11 features selected from Correlation Coefficient and P-value, to predict our response variable ***Diagnosis*** which yields the below results with the confusion matrix and the accuracy:

```
> cm_kmeans1

pre_kmeans  Benign Malignant
  Benign        96        10
  Malignant     14        51
> sum(diag(cm_kmeans1))/sum(cm_kmeans1)
[1] 0.8596491
```

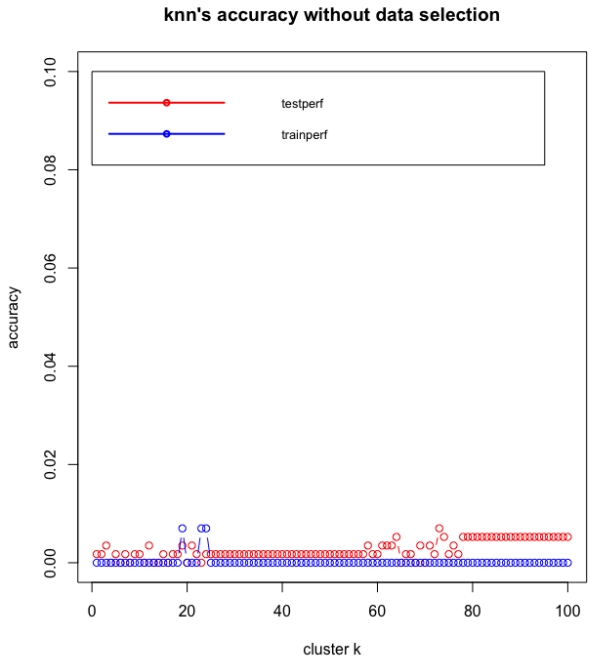
We observe that the K-Means Model with our selected 11 features has a fairly better accuracy of ~85% which is roughly 71% better than the K-Means Model with the original 30 features. At this stage, we can be confident to use the K-Means Model with our selected 11 features to predict the results.

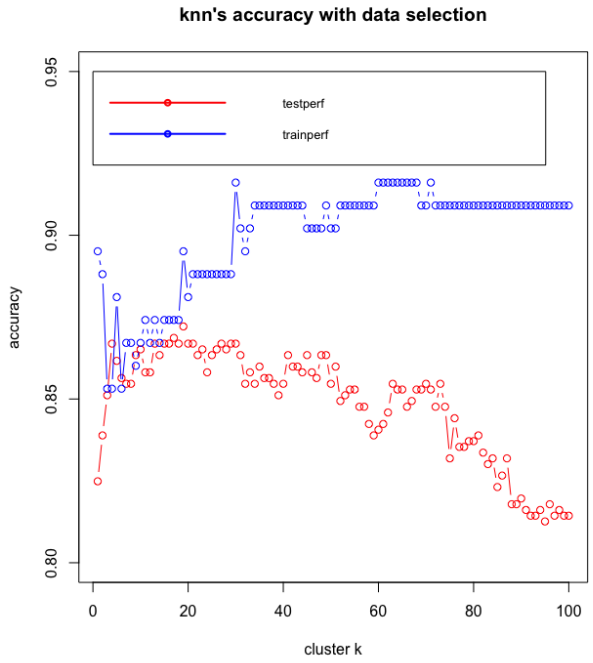
K-nearest neighbors Model:

K-nearest neighbors algorithm is also the simple and intuitive algorithm in machine learning that uses a data set in which each case has been classified into several classes by Euclidean distance's function to predict the classification of the new sample points.

In this step, we repeat the preceding operation for the cluster k = 1, 2, 3, …, 100; using the define function *testperf* and *trainperf* to compute the associated percentages *testperf* and *trainperf* of correct classifications on the TESTSET and TRAINSET. The two graphs below are a plot of both accuracies of the *testpref* in red and the *trainperf* in blue.

We observe that the KNN model with 11 features selected has a better performance of approximately from 85% to 90% than the one with the 30 intact features having a worse performance of near zero. Therefore, we can conclude that it is essential to eliminate the irrelevant features and select the rest of them before further data-driven analysis, such as KNN, K-Means, Linear Regression, and etc.





Conclusion:

All in all, as the results from our research, we can conclude that Correlation Coefficient and P-value does have an impact on feature selection. In our dataset, after selecting the most significant features by using correlation coefficient and p-value, we observe improvements of roughly 71 percent in our K-Means Model (unsupervised learning) and approximately 5 percent in our KNN Model (supervised learning) if compared with the same models using our original features.

In general, the existence of irrelevant features in the data set may degrade learning quality and consume more memory and computational time that could be saved if these features were removed. From the clustering point of view, removing irrelevant features will not negatively affect clustering accuracy whilst reducing required storage and computational time. In addition, different relevant features may produce different clustering. Therefore, different subsets of relevant features may result in different clustering, which greatly help in discovering different hidden patterns in the data. Motivated by these facts, different clustering techniques were proposed to utilize feature selection methods that eliminate irrelevant and redundant features while keeping relevant features in order to improve clustering efficiency and quality.

APPENDIX

Reference

archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
linkedin.com/pulse/what-multicollinearity-how-affects-model-performance-machine-cheruku/
en.wikipedia.org/wiki/K-means_clustering
en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

R-code

```
rm(list = ls())
cat("\f")
DATA <- read.csv(file = 'DATA.csv')
head(DATA)
dim(DATA)  # rows=569, cols=32
str(DATA)  # ID the structure of the data, num/factor/int/chr
        # all are numeric except column "diagnosis" (M = malignant, B = benign)
# convert the categorical column 'diagnosis' to numeric values
# 1 = B = benign and 2 = M = malignant
data <- data.frame(data.matrix(data.frame(unclass(DATA))))
data<-data[,-1]
str(data)

# generate the correlation matrix
data.corr = round(cor(data),3)  # remove the id column, and round to 3 decimal places
data.corr[1:5, 1:5]  #5x5 correlation matrix of the data

# ggpairs of the first 5 columns
library(GGally)
library(ggplot2)
ggpairs(data[,1:5])

# heatmap
library(corrplot)
corrplot(data.corr)
corrplot(data.corr[1:5,1:5])
# Next, we compare the correlation between features and
```

```
# remove one of two features that have a correlation
# higher than 0.9
# set the upper triangle to be zero and then remove any rows
# that have values over 0.90
tmp = data.corr
tmp[upper.tri(tmp)] <- 0
diag(tmp) <- 0
# Above two commands can be replaced with
# tmp[!lower.tri(tmp)] <- 0
data.new <- data[,!apply(tmp,2,function(x) any(x > 0.90))]
head(data.new)
dim(data.new)
names(data.new)
corrplot(cor(data.new))
# we can see that only 20 columns were selected.
```

using VIF to identify (further) multicollinearity

```
# take a linear model of the full data
data.lm = lm(diagnosis~.,data=data)
summary(data.lm)
library(car)
#create vector of VIF values
vif_data = vif(data.lm)
#create horizontal bar chart to display each VIF value
barplot(vif_data,main = "VIF Values", horiz = TRUE, col = "steelblue" )
#add vertical line at 10
abline(v = 10, lwd = 3, lty = 2)
####
#### now using the new data found after removing feature with correlation higher than 0.90
data.new.lm = lm(diagnosis~.,data = data.new)
summary(data.lm)
#create vector of VIF values
vif_data.new = vif(data.new.lm)
#create horizontal bar chart to display each VIF value
barplot(vif_data.new ,main = "VIF Values", horiz = TRUE, col = "steelblue" )
#add vertical line at 10
abline(v = 10, lwd = 3, lty = 2)
```

building the model

##KNN##

```
DATA = data.frame(read.csv("data.csv", na.strings = ""))
true_set = DATA[,2]
X = DATA[,c(-1,-2)]

#knn without data selection
## data cloning
cl1 = X[true_set == "B",]
cl2 = X[true_set == "M",]

D1 = ifelse(nrow(cl1)<=nrow(cl2), list(cl1), list(cl2))[[1]]
D0 = ifelse(nrow(cl1)>nrow(cl2), list(cl1), list(cl2))[[1]]

set.seed(123)
x = sample(x = nrow(D1), size = as.integer(0.8*nrow(D1)), replace = FALSE)
time1 = ceiling(as.integer(0.8*nrow(D0))/length(x))
i = sample(x = setdiff(c(1:(length(x)*time1)), seq(1, length(x)*time1, time1)),
        size = length(x)*time1 - as.integer(0.8*nrow(D0)),
        replace = FALSE)
pd_D1train = rep(x, each = time1)[-i]
setdiff(x, pd_D1train) # verification for all the original 80% of cases are not missed, so we
expect the output be integer(0)
# setdiff(x, y) is the material that is in x, that is not in y

y = setdiff(c(1:nrow(D1)), x)
TIME1 = ceiling((nrow(D0) - as.integer(0.8*nrow(D0)))/length(y))
j = sample(x = setdiff(c(1:(length(y)*TIME1)), seq(1, (length(y)*TIME1), TIME1)),
        size = length(y)*TIME1 - (nrow(D0) - as.integer(0.8*nrow(D0))),
        replace = FALSE)
pd_D1test = rep(y, each = TIME1)[-j]
setdiff(y, pd_D1test) # verification for all the original 20% of cases are not missed, so we expect
the output be integer(0)

intersect(pd_D1train, pd_D1test) # verification for test set and training set do not intersect

trainD1 = D1[pd_D1train,]
testD1 = D1[pd_D1test,]

pd_D0train = sample(x = nrow(D0), size = round(0.8*nrow(D0)), replace = FALSE)
trainD0 = D0[pd_D0train,]
testD0 = D0[-pd_D0train,]
```

```
TRAINSET = rbind(trainD1, trainD0)[,-1]
TESTSET = rbind(testD1, testD0)[,-1]

TRAINSET_TARGET= as.factor(rbind(trainD1, trainD0)[,1])
TESTSET_TARGET = as.factor(rbind(testD1, testD0)[,1])

##
library(class)
test.pred = function(k) {
  set.seed(1)
  test.pred = knn(train = TESTSET, test = TRAINSET, cl = TESTSET_TARGET, k = k)
  return(test.pred)
}
test.table = function(k) {
  test.pred = test.pred(k)
  test.table = table(TRAINSET_TARGET, test.pred)
  return(test.table)
}
testconf = function(k) {
  test.table = test.table(k)
  testconf = prop.table(test.table, margin = 1)
  return(testconf)
}
testperf = function(k) {
  test.table = test.table(k)
  testperf = sum(diag(test.table))/sum(test.table)
  return(testperf)
}
train.pred = function(k) {
  set.seed(1)
  train.pred = knn(train = TRAINSET, test = TESTSET, cl = TRAINSET_TARGET, k = k)
  return(train.pred)
}
train.table = function(k) {
  train.pred = train.pred(k)
  train.table = table(TESTSET_TARGET, train.pred)
  return(train.table)
}
trainconf = function(k) {
```

```
  train.table = train.table(k)
  trainconf = prop.table(train.table, margin  = 1)
  return(trainconf)
}
trainperf = function(k) {
  train.table = train.table(k)
  trainperf = sum(diag(train.table))/sum(train.table)
  return(trainperf)
}

layout(matrix(c(1), 1, 1))
k = c(1:100)
testperf_k = mapply(testperf, k)
trainperf_k = mapply(trainperf, k)
plot(k, testperf_k, type = "b", col = "red", xlab = "cluster k",
     ylab = "accuracy", main = "knn's accuracy without data selection",
     ylim = c(0, 0.1))
lines(k, trainperf_k, type = "b", col = "blue")
legend(0, 0.1, c("testperf", "trainperf"), lwd = 2,
       col = c("red", "blue"), pch = 1, cex = 0.8)
```

#knn with data selection

```
data_new = subset(DATA, select=c("diagnosis", "radius_mean", "texture_mean",
"concavity_mean", "fractal_dimension_mean",  "radius_se", "compactness_se", "concavity_se",
"concave.points_se", "smoothness_worst","symmetry_worst", "fractal_dimension_worst"))

## data cloning
cl1 = data_new[data_new[,1] == "B",]
cl2 = data_new[data_new[,1] == "M",]

E1 = ifelse(nrow(cl1)<=nrow(cl2), list(cl1), list(cl2))[[1]]
E0 = ifelse(nrow(cl1)>nrow(cl2), list(cl1), list(cl2))[[1]]

set.seed(123)
x2 = sample(x = nrow(E1), size = as.integer(0.8*nrow(E1)), replace = FALSE)
time2 = ceiling(as.integer(0.8*nrow(E0))/length(x2))
i2 = sample(x = setdiff(c(1:(length(x2)*time2)), seq(1, length(x2)*time2, time2)),
           size = length(x2)*time2 - as.integer(0.8*nrow(E0)),
           replace = FALSE)
```

```
pd_E1train = rep(x2, each = time2)[-i2]
setdiff(x2, pd_E1train) # verification for all the original 80% of cases are not missed, so we
expect the output be integer(0)

y2 = setdiff(c(1:nrow(E1)), x2)
TIME2 = ceiling((nrow(E0) - as.integer(0.8*nrow(E0)))/length(y2))
j2 = sample(x = setdiff(c(1:(length(y2)*TIME2)), seq(1, (length(y2)*TIME2), TIME2)),
        size = length(y2)*TIME2 - (nrow(E0) - as.integer(0.8*nrow(E0))),
        replace = FALSE)
pd_E1test = rep(y2, each = TIME2)[-j2]
setdiff(y2, pd_E1test) # verification for all the original 20% of cases are not missed, so we
expect the output be integer(0)

intersect(pd_E1train, pd_E1test) # verification for test set and training set do not intersect


trainE1 = E1[pd_E1train,]
testE1 = E1[pd_E1test,]

pd_E0train = sample(x = nrow(E0), size = round(0.8*nrow(E0)), replace = FALSE)
trainE0 = E0[pd_E0train,]
testE0 = E0[-pd_E0train,]

TRAINSET = rbind(trainE1, trainE0)[,-1]
TESTSET = rbind(testE1, testE0)[,-1]

TRAINSET_TARGET = as.factor(rbind(trainE1, trainE0)[,1])
TESTSET_TARGET = as.factor(rbind(testE1, testE0)[,1])

layout(matrix(c(1), 1, 1))
k = c(1:100)
testperf_k = mapply(testperf, k)
trainperf_k = mapply(trainperf, k)
plot(k, testperf_k, type = "b", col = "red", xlab = "cluster k",
    ylab = "accuracy", main = "knn's accuracy with data selection",
    ylim = c(0.80, 0.95))
lines(k, trainperf_k, type = "b", col = "blue")
legend(0, 0.95, c("testperf", "trainperf"), lwd = 2,
      col = c("red", "blue"), pch = 1, cex = 0.8)
```