
# Predictive Modelling Of Performance Grades Based on Demographics and Physical Measurements

CLASSIFICATION MODEL

**For**
MA 5790
Qiuying Sha

**By**
Group #5
Vikramaditya Gurrapu
Vinay Palakurthy

**Submitted on**
December 14, 2023

# Abstract

Maintaining physical performance throughout life is crucial for well-being, yet age and individual differences pose challenges in optimizing exercise strategies. This study addresses this complexity by developing a predictive model to estimate or classify performance grades (A-D) based on age, gender, and diverse exercise performance data. We aim to identify key relationships between these variables and performance by analyzing a comprehensive dataset of 13,393 individuals, encompassing body composition, cardiovascular health, grip strength, and specific exercise measures like flexibility, endurance, and power. Stratification by grade and consideration of gender differences will refine our analysis, enabling the development of targeted exercise recommendations for each group. By uncovering these relationships, this study strives to empower individuals of all ages and fitness levels to achieve optimal performance and promote healthy aging

# Table Of Contents

# Background

Physical performance directly impacts our well-being and quality of life. Unfortunately, aging often leads to declining physical capabilities, exacerbating health risks and decreasing daily function. While regular exercise can mitigate these declines, finding the optimal strategy for each individual is a complex puzzle. Factors like age, gender, and baseline fitness levels all play a crucial role in determining the effectiveness of different exercise programs. This study seeks to address this gap in our understanding by building a predictive model. By analyzing a rich dataset incorporating diverse performance metrics, we aim to illuminate the intricate interplay between individual characteristics, exercise behavior, and overall performance. This knowledge will not only empower individuals to choose the most effective exercise for their unique needs but also inform the development of personalized health interventions and targeted exercise recommendations for distinct performance groups. Ultimately, promoting optimal performance across the lifespan will contribute to a healthier and more vibrant aging experience for all.

# Variable Introduction and Definitions

The raw data used in this study was collected by the Korea Sports Promotion Foundation and extensive post-processing and filtering were performed, to ensure the quality of the data. The resulting dataset is readily available on Kaggle (https://www.kaggle.com/datasets/kukuroo3/body-performance-data). The dataset comprises 13,393 records featuring 12 variables: 10 continuous predictors, 1 categorical predictor, and 1 response variable (performance grades). Below is a list of the variables, and names as used in the analysis, with their descriptions.

| **Predictors** | **Description** |
|---|---|
| age | Age of the individual (20 - 64 years) |
| height_cm | Height of the individual, measured in cm |
| weight_kg | Weight of the individual, measured in cm |
| body fat_% | Percentage of body fat the individual has |
| diastolic | Minimum Diastolic blood pressure reading |
| systolic | Maximum Systolic blood pressure reading |

| | |
|---|---|
| gripforce | Measured strength of individual |
| sit and bend forward_cm | Distance the individual can bend forward while sitting (flexibility) |
| sit-ups counts | Number of sit-ups the individual can complete |
| broad jump_cm | The distance the individual can achieve in a broad jump (power) |
| gender | Sex of the individual (female or male) |

**Table 1:** Table for variables description

The predictor 'gender' is the only categorical predictor in the dataset. To prepare the data for model building, we will perform a comprehensive pre-processing, which includes handling missing values and outliers, filtering irrelevant features, and applying appropriate transformations. Both linear and nonlinear classification models will be explored and compared.

# Preprocessing of the predictors

## Preprocessing

We initiated our analysis by examining the occurrence of missing values in the dataset. In the event of significant missing values, we would have considered either removing the corresponding samples or applying imputation techniques such as mean/median imputation or k-nearest neighbors. Fortunately, our dataset exhibited no missing values, allowing us to proceed with data preprocessing and model development without additional handling of missing values. Given that our predictors included weight and height, we introduced a continuous predictor, 'BMI,' and generated dummy variables for the categorical predictor 'gender.'

```
> sapply(pred_var,function(x) sum(is.na(x)))
                    age               height_cm               weight_kg             body.fat_.
                      0                       0                       0                       0
              diastolic                systolic               gripForce sit.and.bend.forward_cm
                      0                       0                       0                       0
        sit.ups.counts           broad.jump_cm
                      0                       0
```

**Figure 1:** NA value in the data

To address the categorical predictor, we conducted a check for near-zero variance and determined that the dataset did not contain predictors with near-zero variance. The bar plot below shows the frequency imbalances in the distribution of the categorical predictor.

```
> nzv_result <- nearZeroVar(pre_data[,categorical], saveMetrics = TRUE)
> print(nzv_result)
[1] freqRatio     percentUnique zeroVar       nzv
<0 rows> (or 0-length row.names)
```

**Figure 2:** Near Zero Variance values

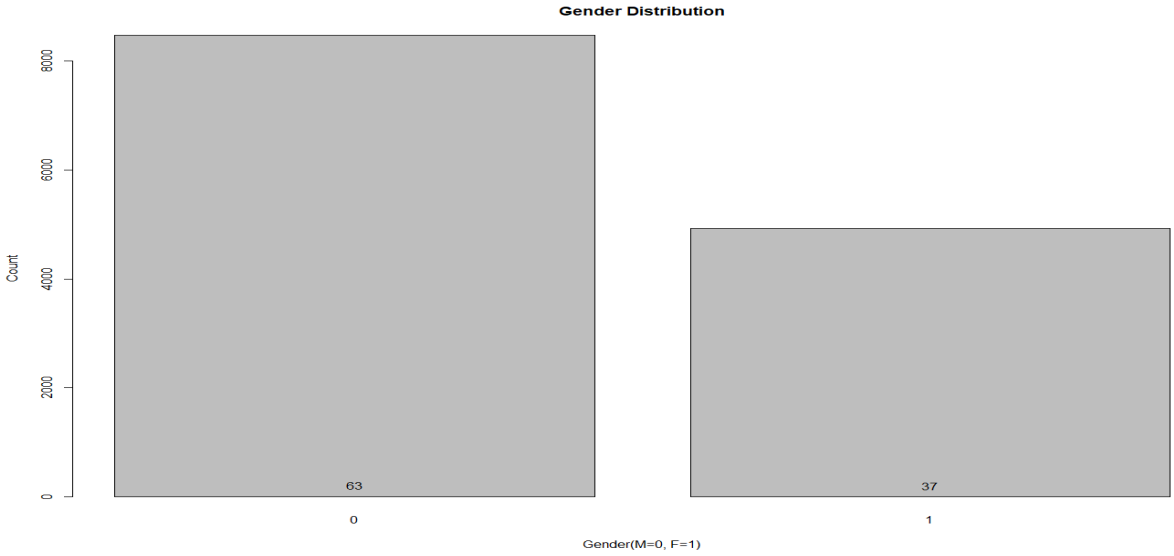


**Figure 3:** Distribution of gender predictor in percentage

For the continuous predictors, we assessed skewness and identified that only 'Age,' 'sit and bend forward,' and 'BMI' exhibited moderate skewness, while the remaining predictors displayed approximately symmetric distributions. The presence of skewness indicated the likelihood of outliers, and upon further examination, we observed that most predictors indeed had outliers.

## Transformations

Below are the histograms and box plots illustrating the distribution of continuous predictors before any transformations. The skewness values of each continuous predictor are presented in the figure below.

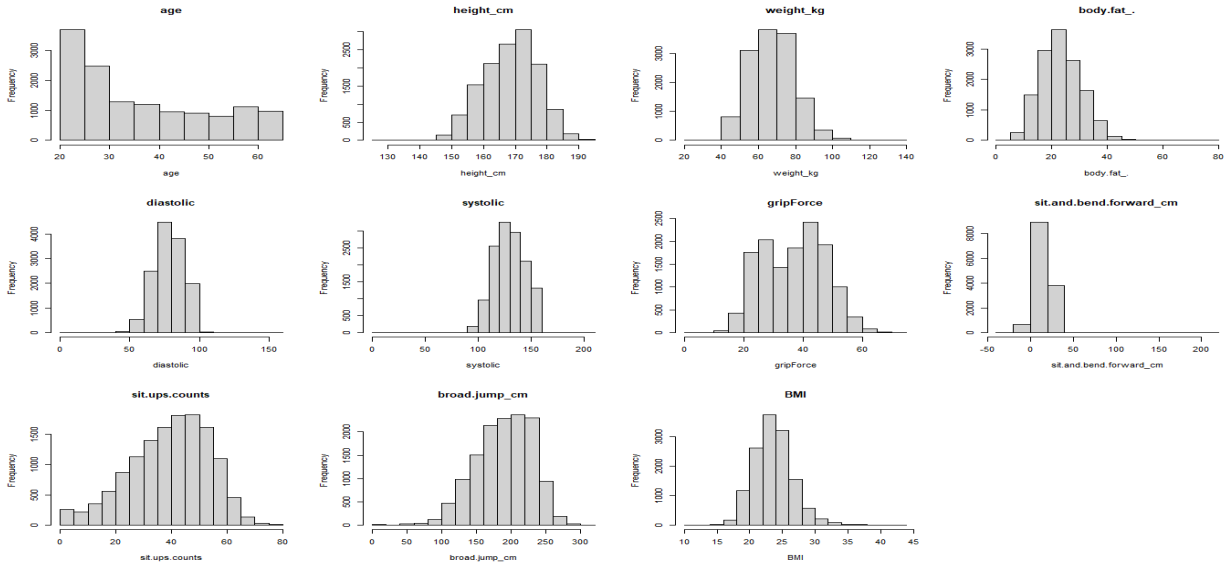


**Figure 4:** Histograms of the continuous predictors before transformations

```
> skewValues
                 age            height_cm            weight_kg            body.fat_.
         0.59976117          -0.18684049           0.34972624           0.36105136
           diastolic             systolic            gripForce sit.and.bend.forward_cm
        -0.15960141          -0.04864271           0.01845236           0.78531607
       sit.ups.counts        broad.jump_cm                  BMI
        -0.46772509          -0.42252789           0.57397573
```

**Figure 5**: Skewness values before transformation

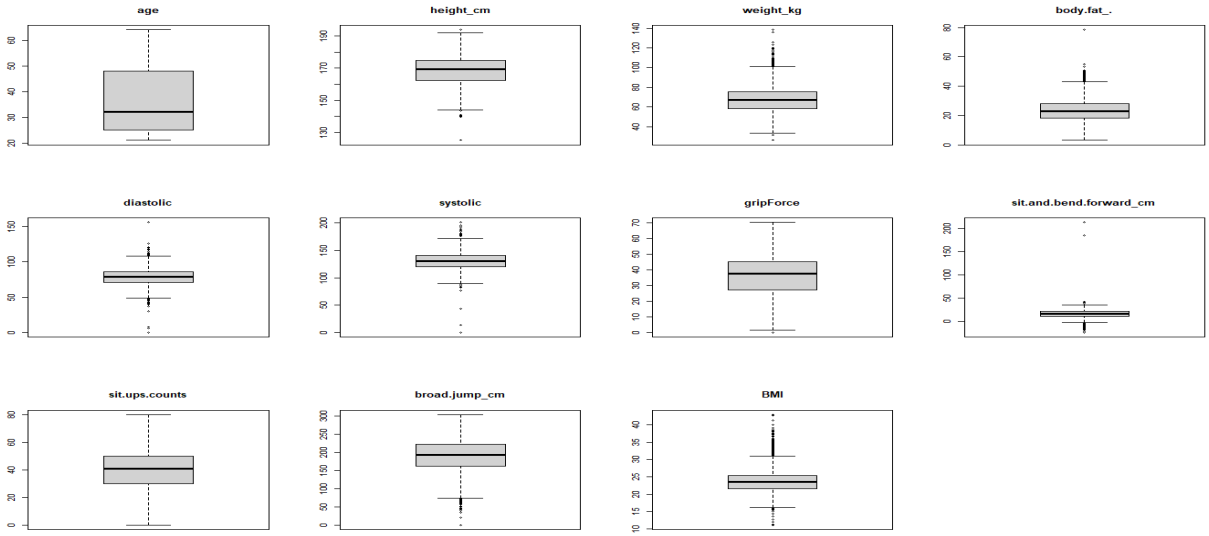


**Figure 6:** Boxplots of continuous predictors before transformations

We undertook a two-step transformation process. Initially, the data underwent centering and scaling, followed by the application of a Box-Cox transformation to mitigate skewness. To address outliers, a spatial sign transformation was then implemented.

The histograms and box plots of the transformed data are provided below, revealing a noticeable improvement in symmetry compared to the original distributions. The figure displaying skewness values further supports this observation, indicating a reduction in skewness. The transformed data also exhibits a reduction in the presence of outliers, contributing to a more robust dataset for subsequent analyses.

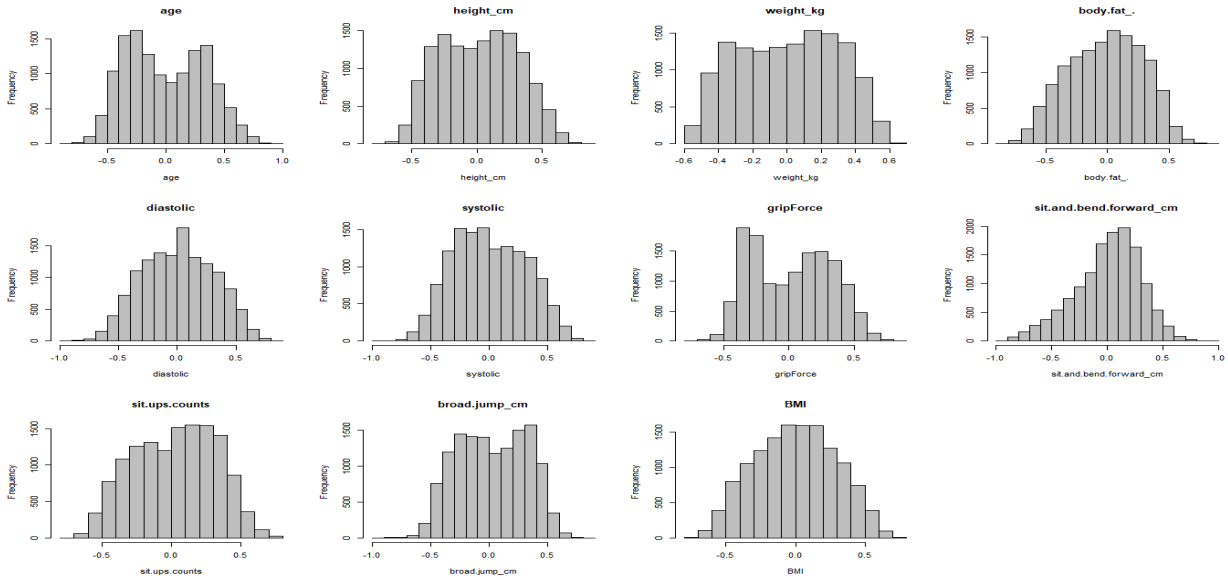


**Figure 7:** Histogram of continuous predictors after transformations

```
> apply(transformed,2,skewness)
                 age            height_cm            weight_kg            body.fat_.
          0.15681346           0.04393598          -0.04547868          -0.13596248
            diastolic             systolic            gripForce sit.and.bend.forward_cm
          0.01185281           0.10635833           0.08339816          -0.45491834
        sit.ups.counts        broad.jump_cm                  BMI
         -0.09139895          -0.04789460          -0.03331211
```

**Figure 8:** Skewness values after transformations

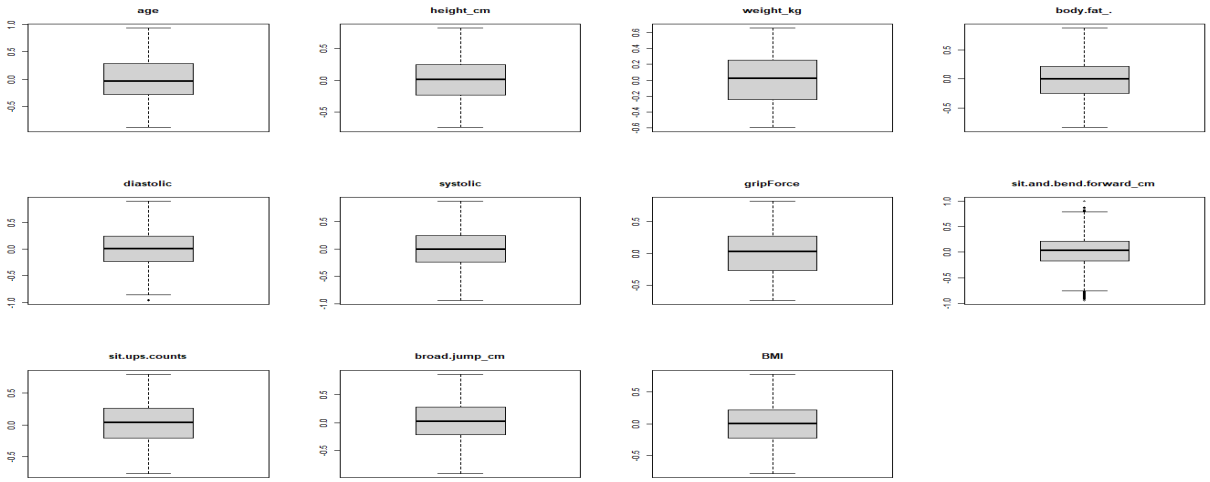


**Figure 9:** Boxplots of after transformations

## Correlations

We generated a correlation plot to delve into the interrelationships among our predictors. In the plot, blue indicates positive correlations between predictors, while red s**ig**nifies negative correlations. Following this, we systematically eliminated predictors with correlations exceeding 0.75. The sole predictor removed was 'weight,' as its correlation with 'BMI' led us to conclude that retaining 'BMI' was sufficient for our analysis. Despite some predictors exhibiting correlations beyond 0.75, we decided to retain all of them, deeming them crucial for constructing our model. This choice was driven by the recognition of their importance, and we opted not to pursue Principal Component Analysis (PCA) at this stage.

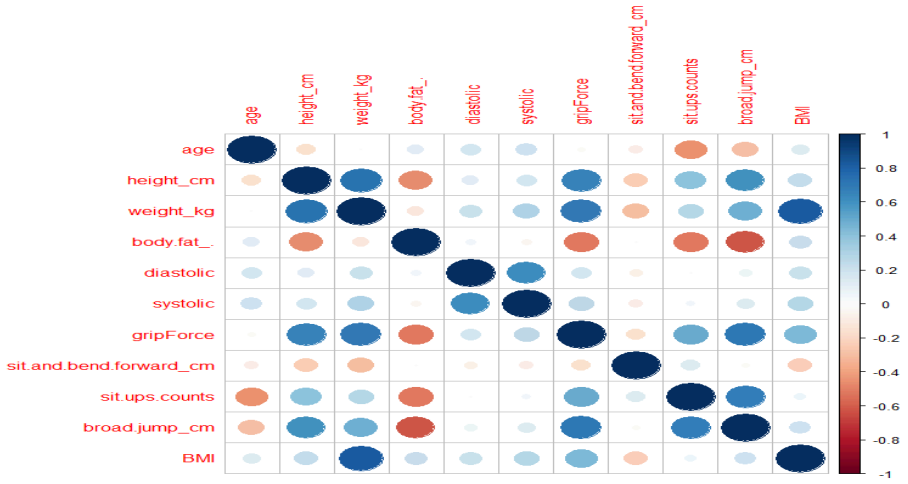


**Figure 10:** Correlation plot before removing highly correlated predictors

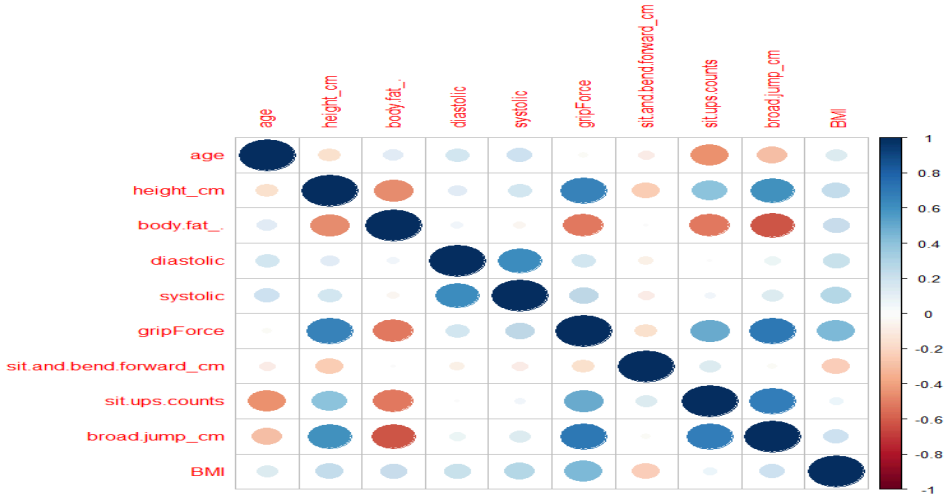


**Figure 11:** Correlation plot after removing highly correlated predictors

# Data Splitting and Resampling

Before partitioning the data, an examination of the distribution of the response variable was conducted, revealing a well-balanced representation across all classes.

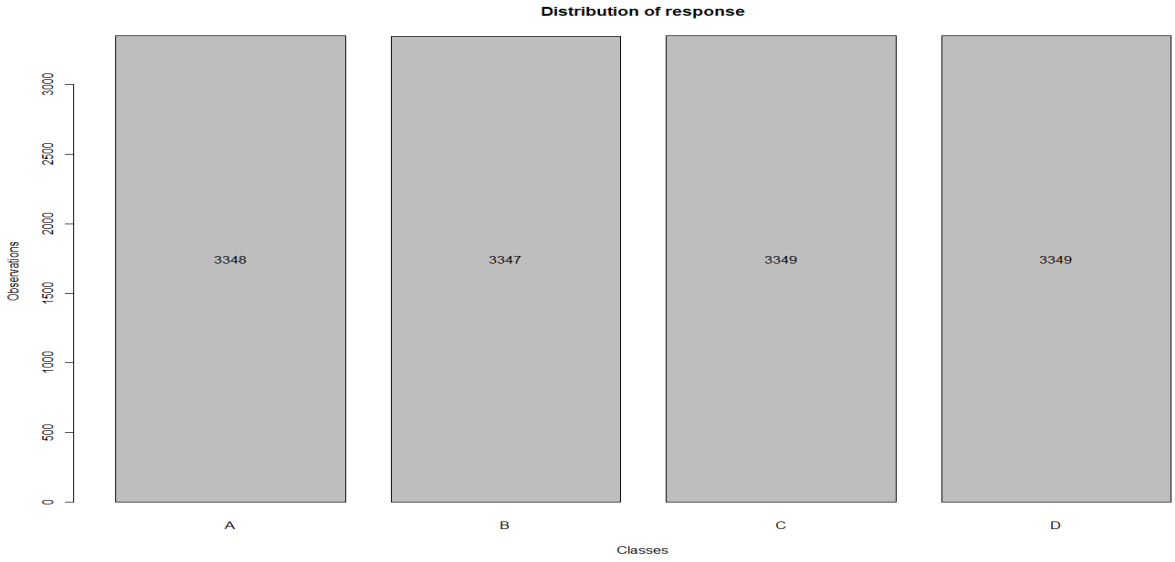


**Figure 12:** Distribution of classes of response variable

The decision was made to employ stratified random sampling to divide the dataset, allocating 80% for training and reserving the remaining 20% for testing purposes. In addition, a stratified K-fold cross-validation approach with K=5 was adopted to ensure

representative sampling and robust model evaluation through resampling the data in a stratified manner. After preprocessing, we have 13,393 records featuring 12 variables: 11 predictors (weight_kg has been removed and BMI has been added) and 1 response variable (performance grades)

# Model Building

We trained the data on several linear and non-linear classification models and the classification statistic we used is Kappa. We chose Kappa as it takes into account the accuracy that would be generated simply by chance.

## Linear Classification Models

We used models like logistic regression, linear discriminant analysis, partial least square discriminant analysis, penalized models, and nearest shrunken centroids. Below is the table summarizing the model's performance on the training and testing dataset along with the best tuning parameters.

| Classification model | Training Kappa | Training Accuracy | Testing Kappa | Testing Accuracy | Best tuning Parameter |
|---|---|---|---|---|---|
| Logistic regression | 0.4594607 | 0.5945930 | 0.4425 | 0.5818 | Decay = 1e-04 |
| Linear Discriminant analysis | 0.4581794 | 0.5936361 | 0.4429 | 0.5822 | NA |
| Partial Least Square Discriminant Analysis | 0.3582046 | 0.5188414 | 0.3508 | 0.5131 | ncomp = 7 |
| Penalized models | 0.4513643 | 0.5884637 | 0.4375 | 0.5781 | alpha = 0.1, lambda = 0.01 |
| Nearest shrunken centroids | 0.3642376 | 0.5228237 | 0.3592 | 0.5194 | threshold = 3 |

**Table 2:** Summary table of Linear models

From the table it is clear that the top two linear classification models are logistic regression and linear discriminant analysis.

## Non-Linear Classification Models

We used models like nonlinear discriminant analysis, neural networks, Flexible discriminant analysis, support vector machines, K-Nearest neighbors, and Naive Bayes. Below is the table summarizing the model's performance on the training and testing dataset along with the best tuning parameters.

| Classification model | Training Kappa | Training Accuracy | Testing Kappa | Testing Accuracy | Best tuning Parameter |
|---|---|---|---|---|---|
| Mixture Discriminant analysis | 0.4609041 | 0.5956870 | 0.4649 | 0.5987 | Subclasses = 5 |
| Regularized Discriminant analysis | 0.339721545 | 0.5046417 | 0.3463 | 0.5097 | Gamma = 1, lambda = 0.0 |
| Quadratic discriminant analysis | 0.5119333 | 0.633948 | 0.5147 | 0.636 | NA |
| Neural Networks | 0.5738997 | 0.6804294 | 0.5884 | 0.6913 | Size = 8, decay = 0.1 |
| Flexible Discriminant analysis | 0.5068240 | 0.6301160 | 0.5037 | 0.6278 | Degree = 2, nprune = 20 |
| Support vector machine | 0.5586943 | 0.6690590 | 0.563 | 0.6723 | Sigma = 0.02795102, c = 64 |
| K-Nearest Neighbours | 0.4570161 | 0.5928277 | 0.4614 | 0.596 | K = 68 |
| Naive Bayes | 0.4001948 | 0.550101 | 0.4136 | 0.5602 | Tuning parameter ‘fl’ was held constant at a value of 2 |

**Table 3:** Summary table for Non-linear models

From the table, it is clear that the top two nonlinear classification models are neural networks and support vector machine.
On comparing both linear and nonlinear classification models we found that nonlinear models performed better for our dataset and had the highest accuracy and Kappa values, hence we chose neural networks and support vector machine models as the top 2 models for our dataset.

| Classification model | Testing Kappa | Testing Accuracy | Best tuning Parameter |
|---|---|---|---|
| Neural Networks | 0.5884 | 0.6913 | Size = 8, decay = 0.1 |
| Support vector machine | 0.563 | 0.6723 | Sigma = 0.02795102, c = 64 |

**Table 4:** Summary table for best models

The table below displays the confusion matrix for both neural networks and support vector machines when making predictions on the test set. Additionally, sensitivity and specificity values have been computed for the top two models.

| **Confusion Matrix for Neural Networks** | | | | |
|---|---|---|---|---|
| | **Reference** | | | |
| **Prediction** | **A** | **B** | **C** | **D** |
| **A** | 551 | 167 | 62 | 3 |
| **B** | 114 | 367 | 118 | 38 |
| **C** | 4 | 118 | 422 | 118 |
| **D** | 0 | 17 | 67 | 510 |

**Table 5:** Table for confusion matrix of the best model (Neural Networks)

| Statistics by Class - Neural Networks | | | | |
|---|---|---|---|---|
| | **A** | **B** | **C** | **D** |
| **Sensitivity** | 0.8236 | 0.5486 | 0.6308 | 0.7623 |
| **Specificity** | 0.8844 | 0.8655 | 0.8804 | 0.9581 |

**Table 6:** Table for sensitivity and specificity of the best model (Neural Networks)

- Class A has the highest sensitivity indicating a relatively good ability to identify true positive instances for this class.
- Class D has the highest specificity, indicating a strong ability to correctly identify instances not belonging to class D.

| Confusion Matrix for Support Vector machines | | | | |
|---|---|---|---|---|
| | **Reference** | | | |
| **Prediction** | **A** | **B** | **C** | **D** |
| **A** | 537 | 169 | 56 | 12 |
| **B** | 129 | 345 | 130 | 40 |
| **C** | 3 | 136 | 423 | 123 |
| **D** | 0 | 19 | 60 | 494 |

**Table 7:** Table for confusion matrix of the second best model (SVM)

| Statistics by Class - Support Vector Machine | | | | |
|---|---|---|---|---|
| | **A** | **B** | **C** | **D** |
| **Sensitivity** | 0.8027 | 0.5157 | 0.6323 | 0.7384 |
| **Specificity** | 0.8819 | 0.8510 | 0.8695 | 0.9606 |

**Table 8:** Table for sensitivity and specificity of the second best model (SVM)

- Class A has the highest sensitivity indicating a relatively good ability to identify true positive instances for this class.
- Class D has the highest specificity, indicating a strong ability to correctly identify instances not belonging to class D.

# Summary

Our analysis leads us to affirm that the standout models are the Neural Networks and Support Vector Machines. These models demonstrate superior accuracy and effectiveness in predicting or classifying performance grades based on the provided features, surpassing the performance of all other models considered. The robust predictive capabilities of these two models make them the top choices for accurately assessing and categorizing performance grades.

**The best and final model chosen for the body performance dataset is the Neural Network.**

## Appendix 1: Supplement material for linear classification models

A. Logistic regression: The optimal model has decay = 1e-04

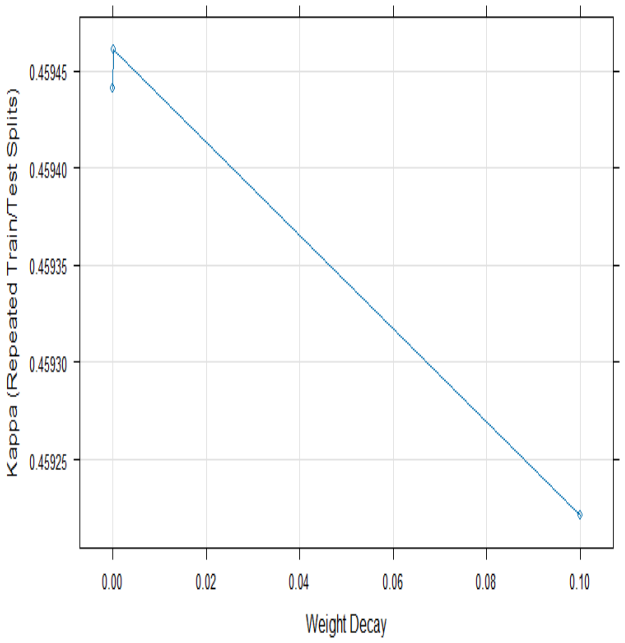


**Figure 13:** Tuning plot for Logistic Regression

B. Linear Discriminant analysis

No plot as there are no tuning parameters for this model

C. Partial least square discriminant analysis: The optimal number of components is 7

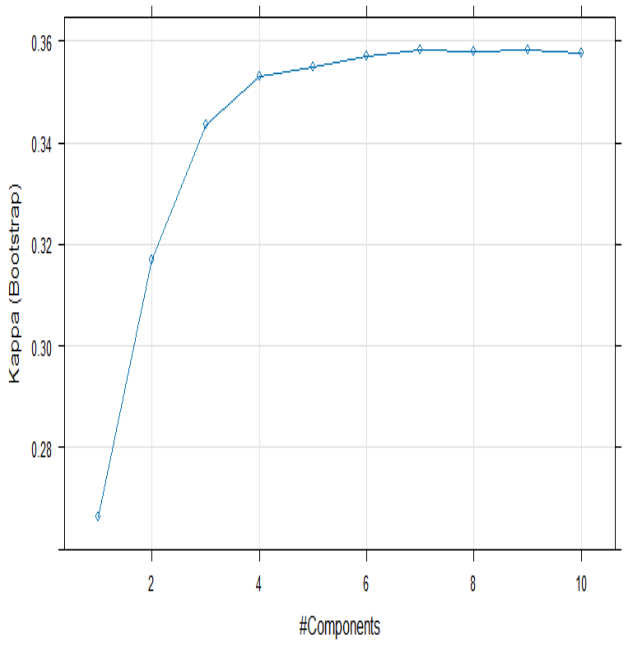


**Figure 14:** Tuning plot for Partial Least Squares Discriminant Analysis (PLSDA)

D. Penalized models: The optimal model uses alpha 0.1 and lambda 0.01

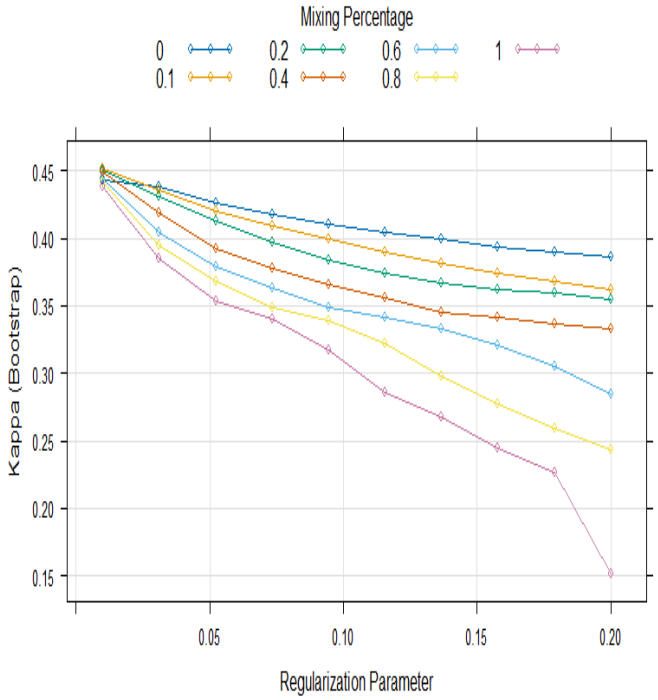


**Figure 15:** Tuning plot for glmnet

E. Nearest Shrunken centroids: The optimal model has threshold 3

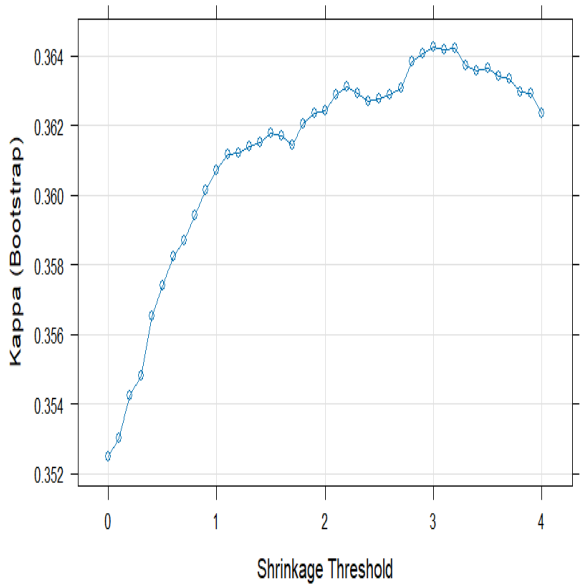


**Figure 16:** Tuning plot for Nearest Shrunken Centroids

# Appendix 2: Supplement material for non-linear classification models

A. Mixture Discriminant analysis: The optimal model has subclasses 5

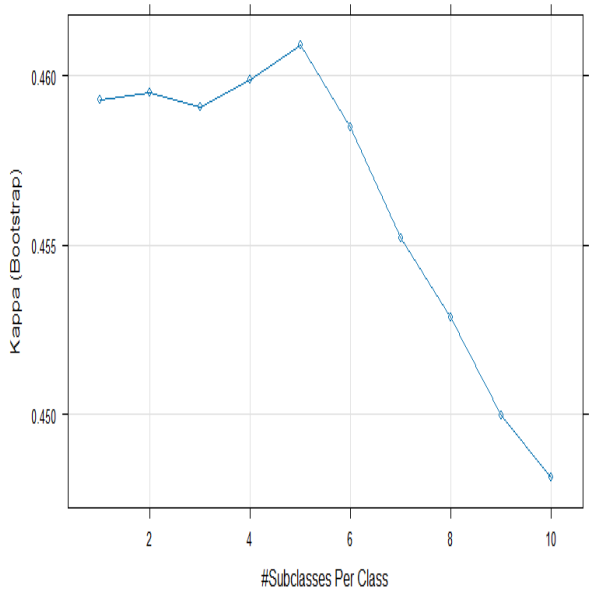


**Figure 17:** Tuning plot for Mixture Discriminant Analysis (MDA)

B. Regularized Discriminant analysis: The optimal model has gamma 1 and lambda 0

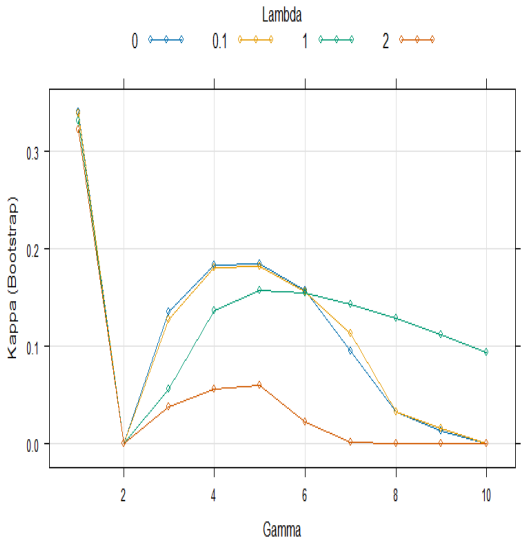


**Figure 18:** Tuning plot for Regularized Discriminant Analysis (RDA)

C. Quadratic discriminant analysis

No Plot, as there are no tuning parameters for this model.

D. Neural Networks: The optimal number of hidden units found was 8 with a decay parameter of 0.1
**(Note: It's worth noting that attempts to tune the model beyond 10 hidden units further resulted in errors.)**

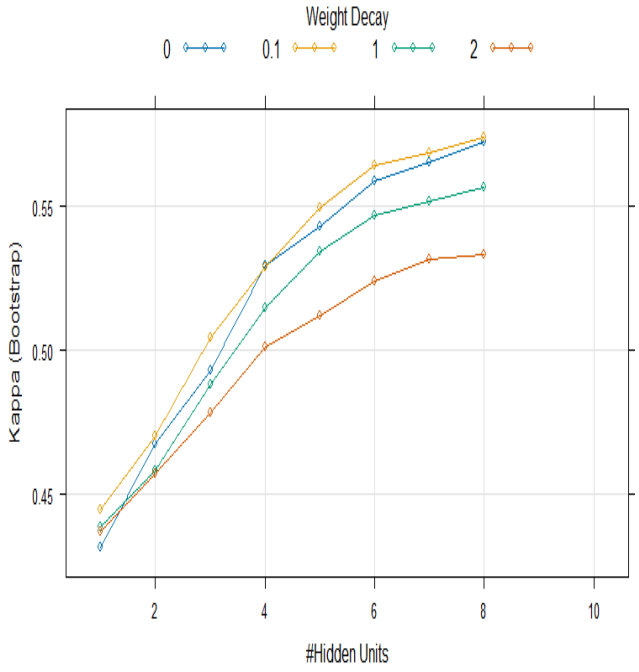


**Figure 19:** Tuning plot for Neural Networks (Nnet)

E. Flexible Discriminant analysis: The optimal model tuned selected 20 terms of degree 2

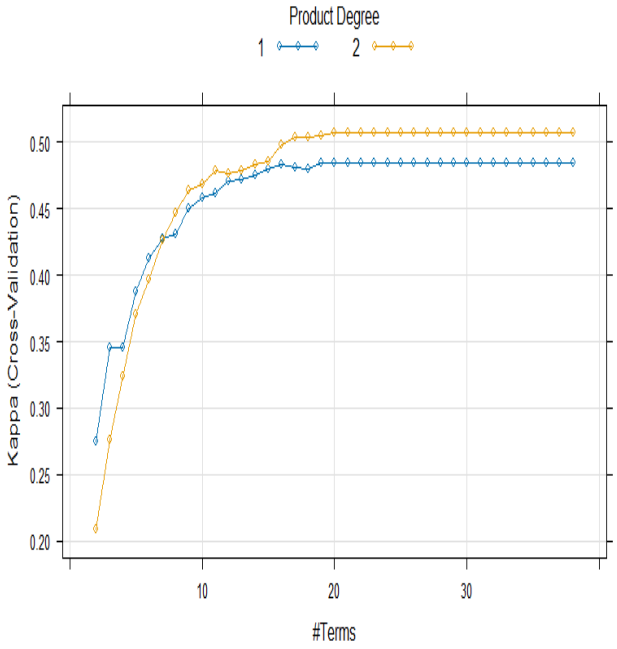


**Figure 20:** Tuning plot for Flexible Discriminant Analysis (FDA)

F. Support vector machine: The sigma parameter was precalculated to be 0.02795102 and the optimum cost was found to be 64

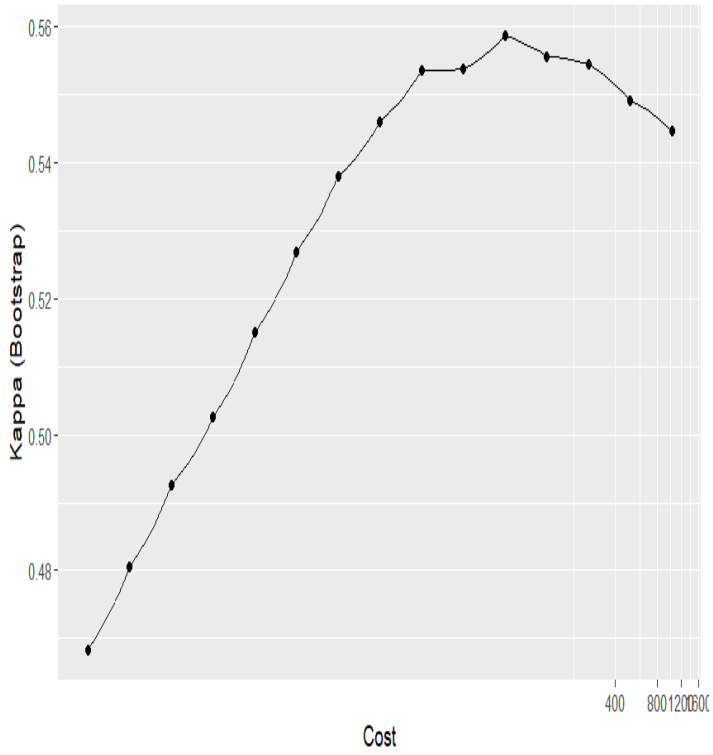


**Figure 21:** Tuning plot for Support Vector Machine (SVM)

G. K-Nearest Neighbors: The optimum number of neighbors was found to be 68

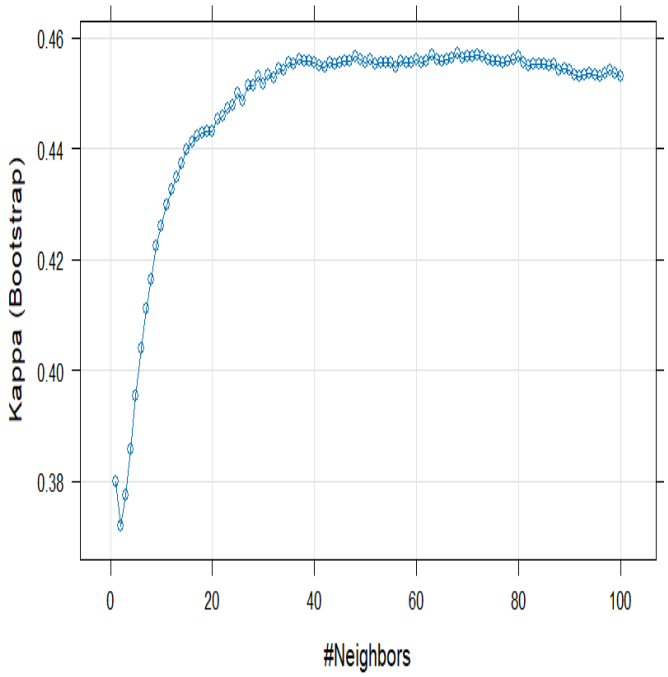


**Figure 22:** Tuning plot for K-Nearst Neighbors (KNN)

H. Naive Bayes

No Plot, as there are no tuning parameters with more than 1 value.

# Appendix 3: Important Variables

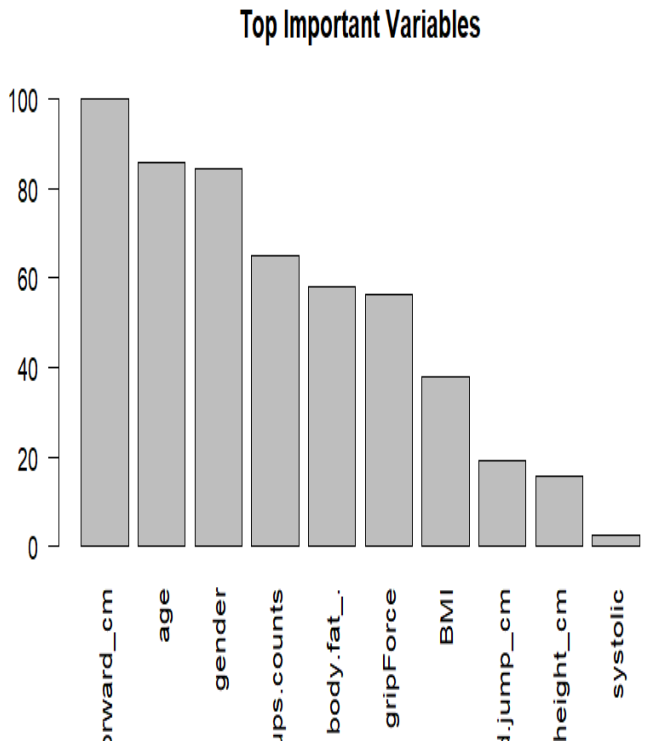


**Figure 23:** Important Variables graph for Neural Networks (First best model)

| Important variables - Nnet | A | B | C | D |
|---|---|---|---|---|
| sit.and.bend.forward_cm | 100.00 | 100.00 | 100.00 | 100.00 |
| age | 85.599 | 85.599 | 85.599 | 85.599 |
| gender | 84.230 | 84.230 | 84.230 | 84.230 |
| sit.ups.down | 65.053 | 65.053 | 65.053 | 65.053 |
| body.fat_ | 58.141 | 58.141 | 58.141 | 58.141 |

**Table 9:** Important variables for Neural Networks (First best model)

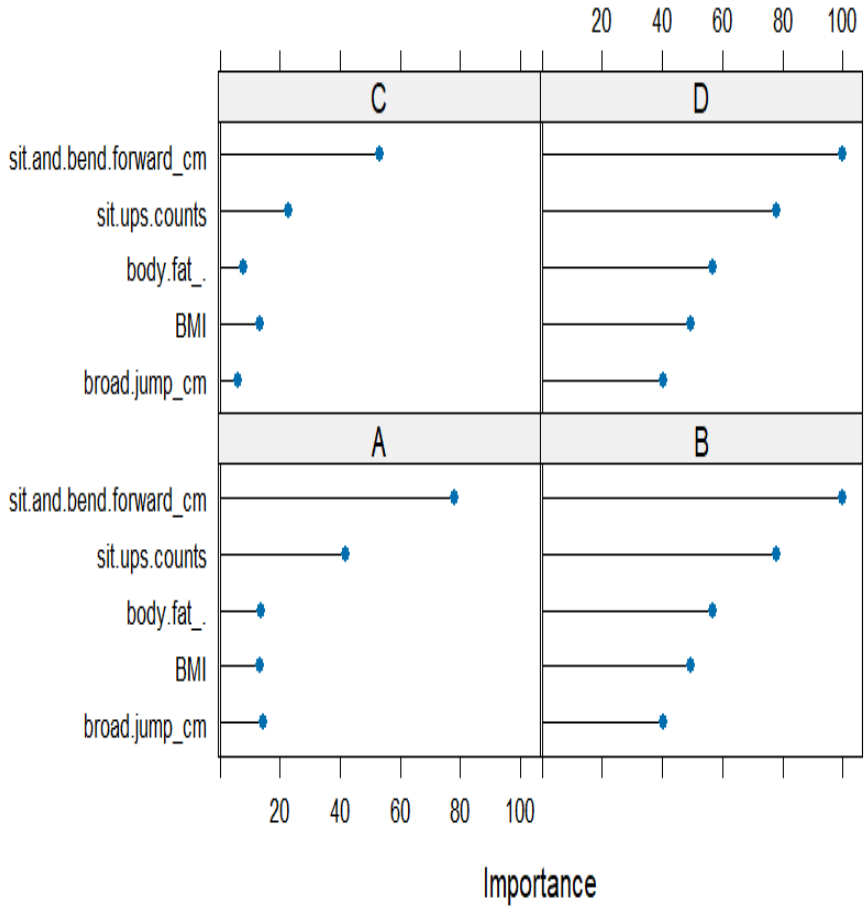


**Figure 24:** Important variables graph for Support Vector Machine (Second best model)

| Important variables - SVM | A | B | C | D |
|---|---|---|---|---|
| sit.and.bend.forward_cm | 77.855 | 100.00 | 53.210 | 100.00 |
| sit.ups.counts | 41.721 | 77.957 | 22.950 | 77.957 |
| body.fat_ | 13.758 | 56.848 | 8.008 | 56.848 |
| BMI | 13.259 | 49.565 | 13.220 | 49.565 |
| broad.jump_cm | 14.570 | 40.371 | 6.009 | 40.371 |

**Table 10:** Important variables for Support Vector Machine (Second best model)

# R Code

```
library(fastDummies)
library(caret)
library(corrplot)
library(e1071)
library(pls)
library(klaR)
library(kernlab)
library(ggplot2)
library(doParallel)

ncores<-4
cl<-makePSOCKcluster(ncores)
registerDoParallel(cl)
pre_data <- read.csv("C:/Users/vinay/OneDrive/Desktop/Predective
Modeling/bodyPerformance.csv")

#Checking for missing values
sapply(pre_data,function(x) sum(is.na(x)))

Response<-as.factor(pre_data$class)

#check BMI column
pre_data$BMI<- round(pre_data$weight_kg / ((pre_data$height_cm/100)^2),2)
pre_data

#dummy variables
pre_data <- dummy_cols(pre_data,select_columns ="gender")
pre_data<-pre_data[,-which(names(pre_data)=="gender")]
pre_data <- pre_data[, c(1,2,3,4,5,6,7,8,9,10,12,11,13,14)]
pre_data

#Gender distribution
male<-round(sum(pre_data$gender_M)/nrow(pre_data),2)*100
female<-round(sum(pre_data$gender_F)/nrow(pre_data),2)*100
gender<-c(male,female)
gender
```

```
par(mfrow= c(1,1))
gender_dist<-barplot(table(pre_data$gender_F), main="Gender Distribution",xlab =
"Gender(M=0, F=1)",ylab="Count")
# Add labels with percentages inside the bars
text(gender_dist,gender/2,labels=gender,pos=3)
# Identify near-zero variance predictors for categorical variables
categorical<-sapply(pre_data,is.factor)
categorical
nzv_result <- nearZeroVar(pre_data[,categorical], saveMetrics = TRUE)
print(nzv_result)
#Class distribution

pre_data
## Missing values
sapply(pre_data, function(x) sum(is.na(x)))

## Frequency distribution - Skewness
par(mfrow= c(3,4))
for(i in 1:11){
  hist(pre_data[,i], main = names(pre_data[i]), xlab=names(pre_data[i]))

}

#Skewness
skewValues <- apply(pre_data[1:11], 2, skewness)
skewValues

## Outliers
par(mfrow= c(3,4))
for(i in 1:11){
  boxplot(pre_data[,i], main = names(pre_data[i]))
}

cor_data <- cor(pre_data[c(1,2,3,4,5,6,7,8,9,10,11)])
cor_data
par(mfrow= c(1,1))
corrplot(cor_data)


## Box-Cox transformation
```

```
xx1 <- preProcess(pre_data[1:11], method = c("BoxCox", "spatialSign","center","scale"))
xx1

# Apply the transformations:

transformed <- predict(xx1, pre_data[1:11])
apply(pre_data[1:11],2,skewness)
apply(transformed,2,skewness)
colnames(transformed)

#hist plots after transformation

par(mfrow= c(3,4))
for(i in 1:11){
  hist(transformed[,i], main = names(transformed[i]), xlab= names(transformed[i]),col =
"grey")
}

#Barplot after transformation
par(mfrow= c(3,4))
for(i in 1:11){
  boxplot(transformed[,i], main = names(transformed[i]))
}

#Removing highly correlated data
highCorr <- findCorrelation(cor_data, cutoff = .75)
length(highCorr)
highCorr
filteredSegData <- transformed[, -highCorr]
filteredSegData

#Correlation plot after removing highly correlated variables
par(mfrow=c(1,1))
corrplot(cor(filteredSegData))

filteredSegData <- cbind(filteredSegData ,pre_data$gender_F)
colnames(filteredSegData)[11] <- "gender"

#Split data into train and test sets
```

```
trainingRows<-createDataPartition(pre_data[,12],p=0.8,list = FALSE)

trainX<-filteredSegData[trainingRows,]
trainY<-as.factor(pre_data[trainingRows,12])
testX<-filteredSegData[-trainingRows,]
testY<-as.factor(pre_data[-trainingRows,12])

###MODELS BUILDING###
##Linear Classification Models##

#Logistic Regression#
set.seed(100)
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = defaultSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

lrFull <- train(trainX,
                y = trainY,
                method = "multinom",
                metric = "Kappa",
                trControl = ctrl)
lrFull
plot(lrFull)
summary(lrFull)
lrPred <- predict(lrFull,newdata = testX)
confusionMatrix(lrPred,testY)

#Linear Discriminant Analysis#
LDAFull <- train(trainX,
                 y = trainY,
                 method = "lda",
                 metric = "Kappa",

                 trControl = ctrl)
LDAFull
plot(LDAFull)
summary(LDAFull)
LDAPred <- predict(LDAFull,newdata = testX)
confusionMatrix(LDAPred,testY)
```

```
#Partial Least Squares Discriminant Analysis#
set.seed(476)
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)

plsFit <- train(x = trainX,
                y = trainY,
                method = "pls",
                tuneGrid = expand.grid(.ncomp = 1:10),
                preProc = c("center","scale"),
                metric = "Kappa",
                trControl = ctrl)
plsFit
plot(plsFit)
#summary(plsFit)
plsPred <- predict(plsFit,newdata = testX)
confusionMatrix(plsPred,testY)

#glmnet#
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(.01, .2, length = 10))
set.seed(476)
glmnTuned <- train(x=trainX,
                   y = trainY,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   preProc = c("center", "scale"),
                   metric = "Kappa",
                   trControl = ctrl)
glmnTuned
plot(glmnTuned)
summary(glmnTuned)
glmnPred <- predict(glmnTuned,newdata = testX)
confusionMatrix(glmnPred,testY)
##varImp(plsFit)
##plot(varImp(plsFit),top=5)

# Nearest Shrunken Centroids model
library(pamr)
nsc_Grid <- data.frame(.threshold=seq(0,4, by=0.1))
```

```
set.seed(951)
nsc_model <- train(x=trainX,y=trainY,
                   method = "pam",
                   tuneGrid = nsc_Grid,
                   metric = "Kappa",
                   trControl = ctrl)
nsc_model

pred_nsc_model <- predict(nsc_model, testX)
pred_nsc_model

plot(nsc_model)
summary(nsc_model)
confusionMatrix(data = pred_nsc_model,reference = testY)

##NON-LINEAR CLASSIFICATION MODELS##

#Mixture Discriminantion Analysis#
set.seed(100)
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)
mdaFit <- train(x = trainX,
                y = trainY,
                method = "mda",
                metric = "Kappa",
                tuneGrid = expand.grid(.subclasses = 1:10),
                trControl = ctrl)
mdaFit
plot(mdaFit)
summary(mdaFit)
mdaPred <- predict(mdaFit,newdata = testX)
confusionMatrix(mdaPred,testY)

mdaFit <- train(x = trainX,
                y = trainY,
                method = "mda",
                metric = "Kappa",
                tuneGrid = expand.grid(.subclasses = 1:4),
                trControl = ctrl)
mdaFit
```

```
plot(mdaFit)
summary(mdaFit)
mdaPred <- predict(mdaFit,newdata = testX)
confusionMatrix(mdaPred,testY)

# Regularized Discriminant Analysis (RDA)
rdaGrid <- expand.grid(.gamma= 1:10, .lambda = c(0, .1, 1, 2))
rdaFit <- train(x = trainX,
                y = trainY,
                method = "rda",
                metric = "Kappa",
                tuneGrid = rdaGrid,
                trControl = ctrl)
rdaFit
plot(rdaFit)
summary(rdaFit)
pred_rda_model <- predict(rdaFit,newdata = testX)
confusionMatrix(pred_rda_model,testY)

# Quadratic Discriminant Analysis (QDA)
set.seed(709)
ctrl<- trainControl(method = "cv", number = 5, returnResamp = "all",
                    classProbs = TRUE,
                    summaryFunction = defaultSummary)
tuneGrid <- expand.grid(parameter = seq(0, 1, by = 0.1))
qdaFit <- train(x = trainX,
                y = trainY,
                method = "qda",
                metric = "Kappa",
                tuneGrid = data.frame(),
                preProcess = c("center","scale"),
                trControl = ctrl)
qdaFit
plot(qdaFit)
summary(qdaFit)
qdaPred <- predict(qdaFit,newdata = testX)
confusionMatrix(qdaPred,testY)

#Neural Networks#
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, 1, 2))
```

```
maxSize <- max(nnetGrid$.size)
numWts <- (maxSize * (11 + 1) + (maxSize+1)*2)
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)
set.seed(456)

nnetFit <- train(x = trainX,
                 y = trainY,
                 method = "nnet",
                 metric = "Kappa",
                 preProc = c("center", "scale", "spatialSign"),
                 tuneGrid = nnetGrid,
                 trace = FALSE,
                 maxit = 2000,
                 MaxNWts = numWts,
                 trControl = ctrl)
nnetFit
plot(nnetFit)
summary(nnetFit)
nnetPred <- predict(nnetFit,newdata = testX)
confusionMatrix(nnetPred,testY)

#Flexible Discriminant Analysis#
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)

fdaTuned <- train(x = trainX,
                  y = trainY,
                  method = "fda",
                  metric = "Kappa",
                  # Explicitly declare the candidate models to test
                  tuneGrid = marsGrid,
                  trControl = trainControl(method = "cv"))

fdaTuned
plot(fdaTuned)
summary(fdaTuned)
fdaPred <- predict(fdaTuned,newdata = testX)
confusionMatrix(fdaPred,testY)

#Support Verctor Machine#
```

```
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)
sigmaRangeReduced <- sigest(as.matrix(trainX))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1],
                               .C = 2^(seq(-4, 10)))
set.seed(476)
svmRModel <- train(x = trainX,
                   y = trainY,
                   method = "svmRadial",
                   metric = "Kappa",
                   preProc = c("center", "scale"),
                   tuneGrid = svmRGridReduced,
                   fit = FALSE,
                   trControl = ctrl)
svmRModel
plot(svmRModel)
ggplot(svmRModel)+coord_trans(x='log2')
summary(svmRModel)
svmPred <- predict(svmRModel,newdata = testX)
confusionMatrix(svmPred,testY)

#K-Nearest Neighbours#
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)
set.seed(476)
knnFit <- train(x = trainingData,
                y = trainResponse,
                method = "knn",
                metric = "Kappa",
                preProc = c("center", "scale"),
                ##tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), ## 21 is
the best
                tuneGrid = data.frame(.k = 1:50),
                trControl = ctrl)
knnFit
plot(knnFit)
summary(knnFit)
knnPred <- predict(knnFit,newdata = testingData)
confusionMatrix(knnPred,testResponse)
```

```
#Bayesian#
set.seed(476)
nbFit <- train( x = trainingData,
                y = trainResponse,
                method = "nb",
                metric = "Kappa",
                ## preProc = c("center", "scale"),
                ##tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), ## 21 is
the best
                tuneGrid = data.frame(.fL = 2,.usekernel = TRUE,.adjust = TRUE),
                trControl = ctrl)

nbFit
plot(nbFit)
summary(nbFit)
nbPred <- predict(nbFit,newdata = testingData)
confusionMatrix(nbPred,testResponse)

#Important variables for best model

# Extract variable importance scores
top_5 <- varImp(nnetFit)$importance

# Subset the matrix to include only rows where "Overall" > 0
top_5_subset <- top_5[top_5[, "Overall"] > 0, ]

# Sort the data frame in descending order based on the "Overall" column
top_5_subset <- top_5_subset[order(-top_5_subset[, "Overall"]), ]

# Plot the top 5 important variables
if (nrow(top_5_subset) > 0) {
  barplot(top_5_subset[, "Overall"], main = "Top Important Variables", col = "grey", las =
2)
} else {
  cat("No variables with positive importance scores.\n")
}
top5 <- varImp(svmRModel)
Top5

plot(top5)
```

# References

1. http://appliedpredictivemodeling.com/
2. https://topepo.github.io/caret/index.html
3. https://www.r-project.org/