Final assignment group 4

Deepika Dilip, Tora Mullings, Daniel Sullivan, Deepa Sharma, Bikram Barua, Newman Okereafor

2022-11-24

Contents

Abstract:

Introduction: Maternal mortality is a leading public health issue in Bangladesh, with 173 deaths per 100k births. Yet with improvements in public health surveillance, a preventative responsive could be better informed with biomarker data and accurate risk predictions.

Methods: For this project, we utilize multinomial models to quantify the contribution of biomarkers in predicting mortality risk. We start with multinomial regression, followed by ordinal regression and an xgboost model.

Results: The gradient-boosting model performed the best in predicting maternal mortality risk at both mid and high levels

Discussion: The use of supervised learning to predict outcomes using biomarkers can be a clinically-relevant tool to reduce mortality burden.

Key words:

maternal health, clinical outcomes

Introduction:

Maternal mortality is a leading public health issue in Bangladesh. Advances in public health outreach and medical pipelines have reduced maternal mortality rates, but there remains a glaring gap, especially when considering additional factors, such as socioeconomic status. One of the WHO Sustainable Development Goals was to reduce the global mortality ratio to less than 70 deaths per 100k births

Here, we further explore mortality risk as a product of standard clinical indicators. We obtained this dataset from the UCI repository. Data was aggregated from different sites, including rural and urban health centers.

According to the WHO approximately 810 women die daily due to pregnancy complications (1). With such a high rate of death associated with childbirth it is important to maximize early interventions in high-risk pregnancies in order to monitor and start early intervention to save both the lives of the mother and child. Because of this need, pregnancy has been the focus of many data scientists' research in developing many predictive algorithms to try and aid in identifying at risk pregnancies, best emergency interventions and various other aspects to help both mothers and doctors. For this reason, we want to look at identifying low-, mid-, and high-risk pregnancies through regression and machine learning methods in order to aid in identifying individuals who could be helped through early intervention.

Literature review:

At-risk pregnancies remain a vital research topic despite technological advances and a shrinking pregnancy/childbirth mortality rate. Predictive modeling has been implemented in several ways to reduce pregnancy risks. There are three major groups of studies that have been performed. The largest group predicted risks and complications involved with the pregnancy in specific scenarios (3) as we are trying to assess with our data set. Many papers also covered predicting delivery methods and successful vaginal delivery (2). The last significant area of study is predicting in-vitro fertilization success rates.

Our analysis concerns at-risk pregnancies using specific clinical indicators to predict risk. Most studies that predict complications do it in a much more specific scope. For example, some studies only predict preterm birth or vaginal birth complications while our approach focuses on high-risk births and uses basic vitals.

Additionally, our analysis works through generalized linear models and progresses into simple machine learning models, whereas further studies implement more sophisticated models. However, there was not a consensus of how these studies approached this type of analysis. For example, some took a similar approach to us using gradient boosting however building on it by adding a random forest and Adaboost ensemble(5). Others took a range of approaches some examples include univariate and multivariate approaches(6), as well as K-means clusters, naive Bayes, logistic regression, and multiple types of neural networks.(4)

Methodology:

Exploratory Data Analysis:

We started by creating exploratory plots to describe positive and negative correlations between predictors and our outcome (risk). We also wanted to make note of the nature of the relationship (e.g. linear or non-linear). Lastly, we used an unsupervised approach to classify patients and gain a better understanding of risk heterogeneity.

Data Attributes

- `Age`: Any ages in years when a women during pregnant.
- `SystolicBP`: Upper value of Blood Pressure in mmHg, another significant attribute during pregnancy.
- `DiastolicBP`: Lower value of Blood Pressure in mmHg, another significant attribute during pregnancy.
- `BS`: Blood glucose levels is in terms of a molar concentration, mmol/L.
- `HeartRate`: A normal resting heart rate in beats per minute.
- `Risk Level`: Predicted Risk Intensity Level during pregnancy considering the previous attribute.

Models Used:

Multinomial Regression

First we partitioned the dataset using a 70-30 split. We initially fit a full model with all included variables as predictors. Next, we fit a series of multinomial models, starting with a full model. We then implemented feature selection based on statistical significance to improve accuracy.

XGBoost

We also decided to try using a model that combined previous models with new ones, subsequently increasing accuracy. Therefore, we decided to fit the eXtreme Gradient Boosting algorthim from the `caret` package. In this case, however, we have to split our outcome: one model will predict high-risk while the other will predict medium-risk.

Ordinal Models

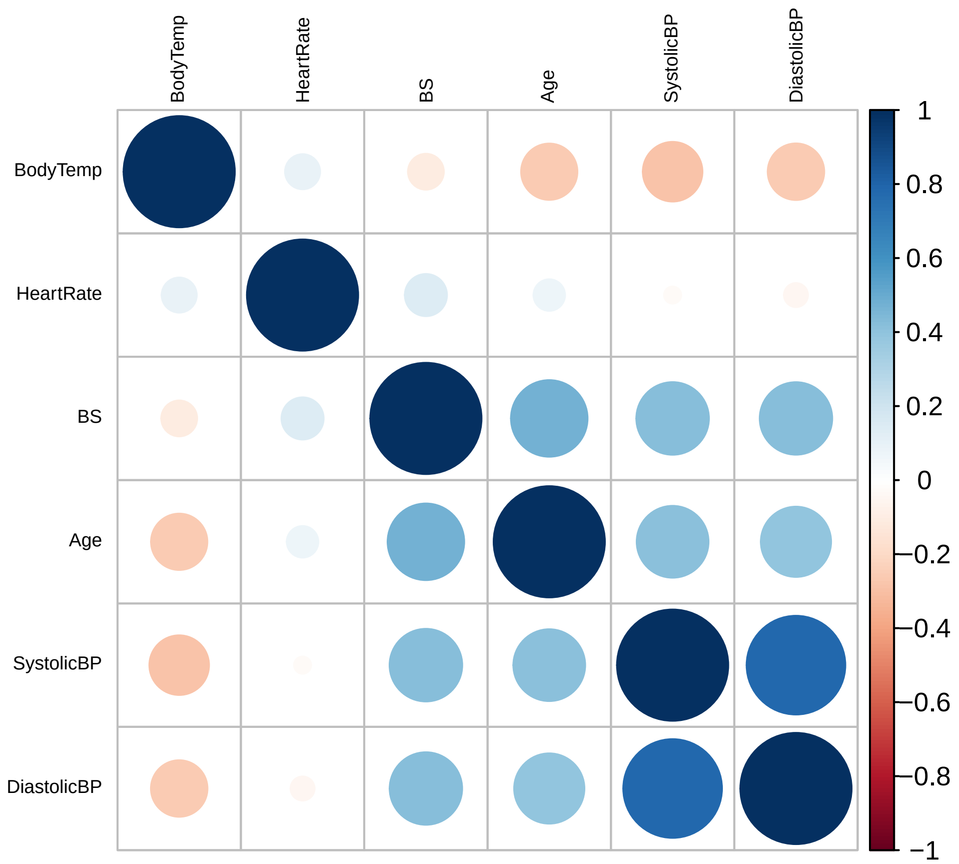
Lastly, we used ordinal models as an alternative to multnomial regression.

Experimentation and Results:

Exploratory Data Analysis

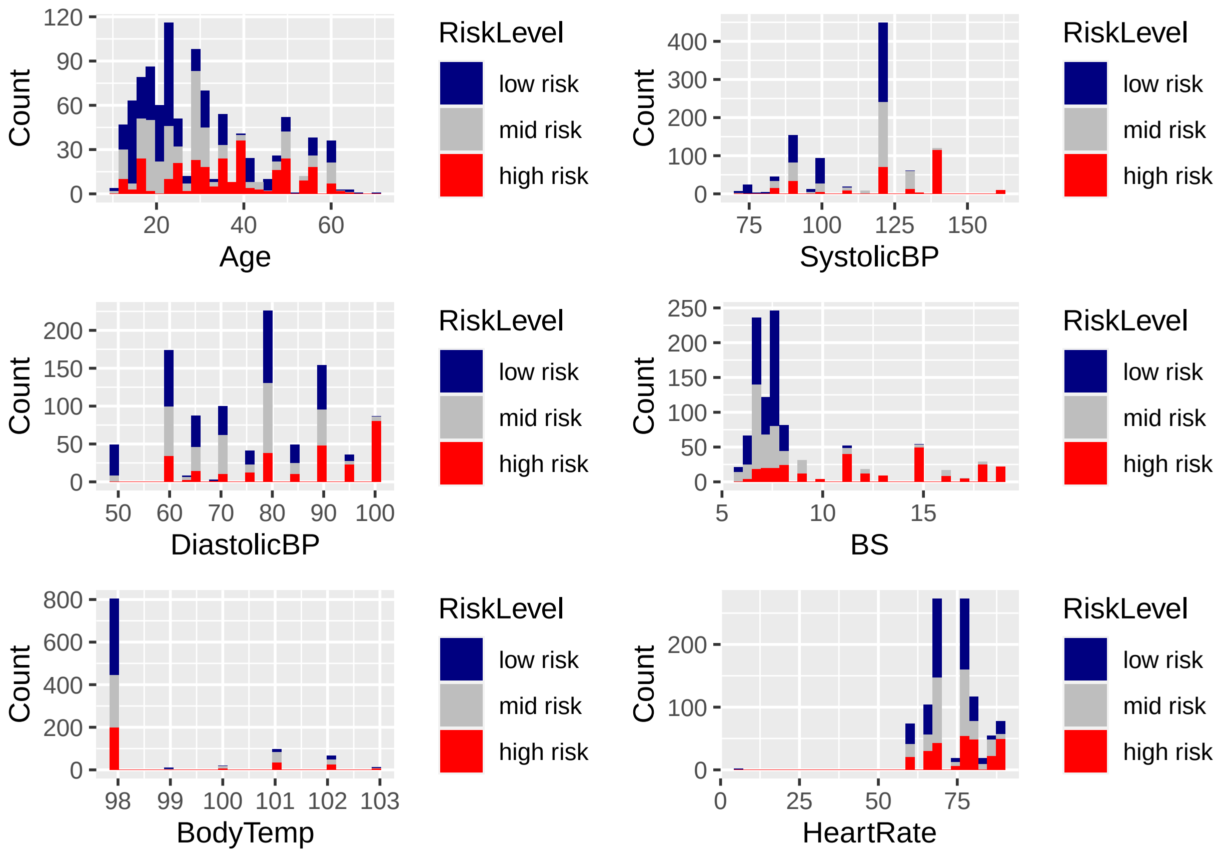
Correlation Plot:

We can start by making a correlation plot to compare continuous values. Age is positively correlated with systolic and diastolic blood pressure.



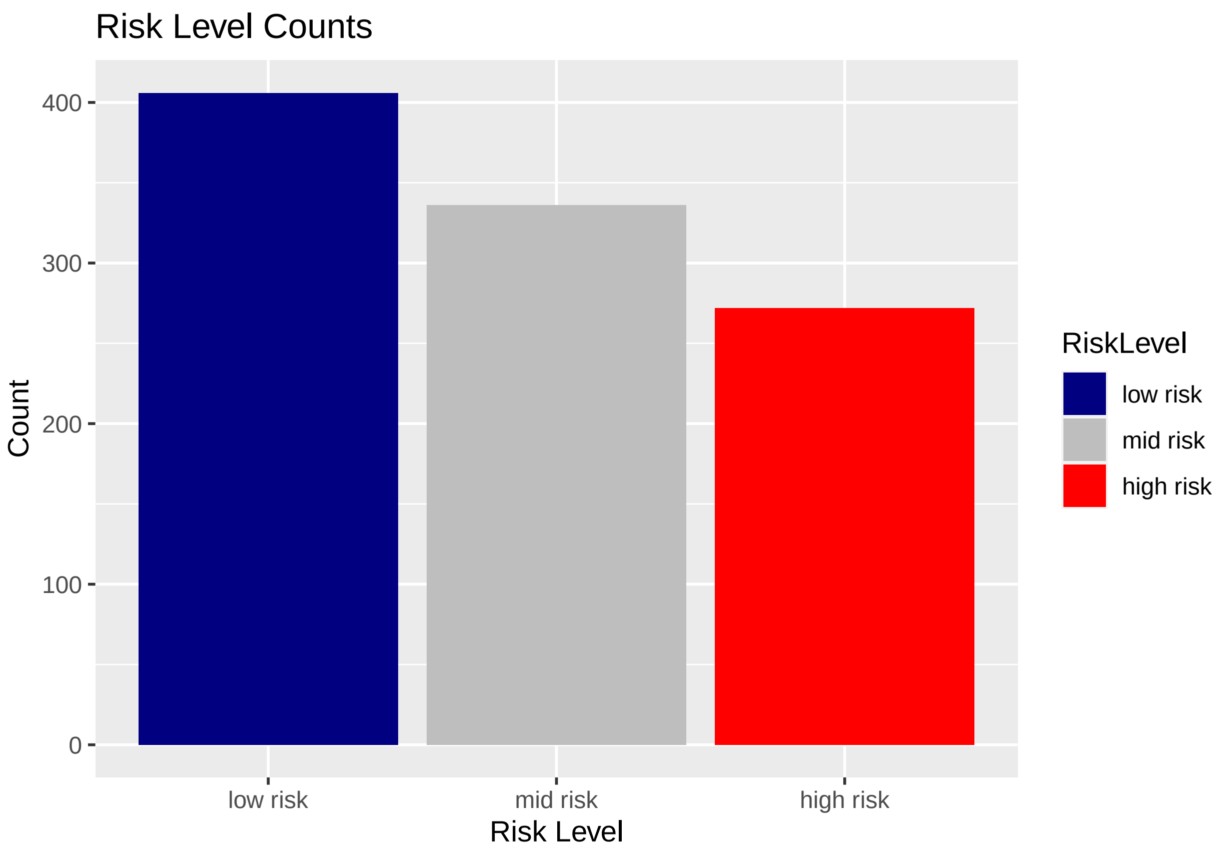
Histograms:

The first step is visualizing data distribution by risk. Here we can see age is skewed, and the extremities of blood pressure and glucose levels are flagged as high risk.



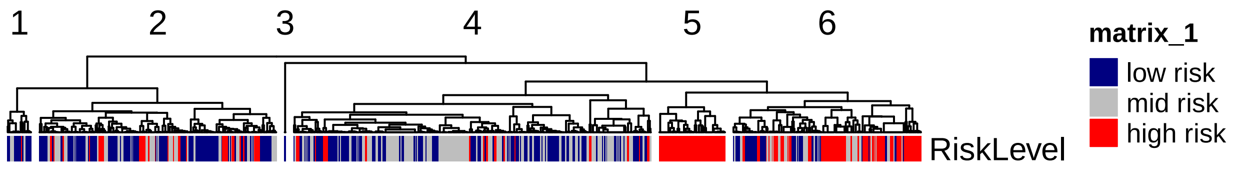
Count
Age
SystolicBP
DiastolicBP
BS
BodyTemp
HeartRate
RiskLevel
low risk
mid risk
high risk

Risk Distribution:



Unsupervised Learning:

One approach we can take is implementing unsupervised clustering since many of the biomarkers are continuous. We can do this by forming a matrix of indicators and seeing if risk levels clusters. From this analysis, we see 6 distinct subgroups: 2 high-risk, 3 low-risk, and 1 mid- risk (with some heterogeneity).



Model Results

Multinomial: Full Model:

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.6356589	0.4062500	0.7125000
Pos Pred Value	0.6307692	0.4020619	0.7307692

Multinomial: Age and Systolic BP as Predictors:

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.6589147	0.3854167	0.6125000
Pos Pred Value	0.5666667	0.3936170	0.8032787

Multinomial: Blood Sugar and Systolic BP as Predictors:

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.7054264	0.2187500	0.6
Pos Pred Value	0.5290698	0.2876712	0.8

Multinomial: Blood Sugar as Predictor:

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.9689922	0.0520833	0.6500
Pos Pred Value	0.5506608	0.3571429	0.8125

XGBoost Model

Predicting High Risk:

	x
Sensitivity	0.9733333
Specificity	0.9875000
Pos Pred Value	0.9954545
Neg Pred Value	0.9294118
Precision	0.9954545
Recall	0.9733333
F1	0.9842697
Prevalence	0.7377049
Detection Rate	0.7180328
Detection Prevalence	0.7213115
Balanced Accuracy	0.9804167

`xBoost`: Predicting Medium Risk

	x
Sensitivity	0.9808612
Specificity	0.9166667
Pos Pred Value	0.9624413
Neg Pred Value	0.9565217
Precision	0.9624413
Recall	0.9808612
F1	0.9715640
Prevalence	0.6852459
Detection Rate	0.6721311
Detection Prevalence	0.6983607
Balanced Accuracy	0.9487640

Ordinal: Full Model

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.620155	0.3958333	0.6125000
Pos Pred Value	0.640000	0.3486239	0.6901408

Ordinal: Blood Sugar, Systolic Blood Pressure, Age

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.6356589	0.4375000	0.5875000
Pos Pred Value	0.5774648	0.4038462	0.7966102

Ordinal: Blood Sugar, Systolic Blood Pressure

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.6201550	0.3958333	0.5875000
Pos Pred Value	0.5594406	0.3689320	0.7966102

Ordinal: Blood Sugar

metric	Class: low risk	Class: mid risk	Class: high risk
Sensitivity	0.8914729	0.15625	0.6375000
Pos Pred Value	0.5693069	0.37500	0.8095238

Summary of Results:

model	accuracy	AIC
mm_model1	0.5836066	1109.227
mm_model2	0.5606557	1208.700
mm_model3	0.5245902	1214.164
mm_model4	0.5967213	1261.509
ordinal1	0.5475410	NA
ordinal2	0.5606557	NA
ordinal3	0.5409836	NA
ordinal4	0.5934426	NA

GLM model construction:

We first tried to predict the level of risk involved with pregnancy using multinomial and ordinal regression modeling. Starting with all variables and then paring down based on correlation to risk level and co-linearity in predictors. We selected first age, systolic blood pressure, and blood sugar level as our first set of restricted predictors, followed by blood sugar and systolic blood pressure for our second.

Although we expected to improve the prediction rate by only keeping the most correlated predictors, both model 2(systolicBP, Age, blood sugar) and model 3(systolicBP, blood sugar) in both multinomial and ordinal regressions showed a decrease in predictive accuracy when compared to models with all predictors. The only model in which we saw the best improvement in our predictions was when we only included blood sugar as a predictor. Because assessing pregnancy risk is a critical topic and could lead to life-saving interventions, we want to strengthen this model and suggest a more in-depth analysis of this data via unsupervised learning methods.

Discussion and Conclusions:

Model Interpretation:

First, we look into the multinomial model assessing the accuracy of all predictors and pairing things down to try and improve the model. Starting with every variable, the multinomial model gives an error rate of 41%, which could be better. From here, we eliminated redundancy and low correlation values to the class level.

The first pared-down model consisting of Age, Systolic blood pressure, and Blood sugar showed an even worse error rate of around 44%. A pared-down model with blood sugar and systolic blood pressure had a missclassification rate of about 48%. The only improvement was upon making a model solely with blood sugar when the error rate dropped to 40%. This shows that the multinational regression clearly does not reflect or predict the data too well.

After the predictions of the supervised models proved unreliable we moved on to more complex methods, implementing gradient boosting. We had to break this into two models but found that the accuracy of prediction by doing this far exceeded those of any of our supervised models. Each model achieving accuracy of ~98% as well as specificity greater then 90% drastically improving on all other results. Combining these two models would result in a much better classification model.

References:

1 Trends in maternal mortality 2000 to 2017: estimates by who, unicef, unfpa, world bank group and the united nations population division. https://www.unfpa.org/featured-publication/trends-maternal-mortality-2000-2017. Accessed 10 Jan 2021.

2 Birara M, Gebrehiwot Y. Factors associated with success of vaginal birth after one caesarean section (vbac) at three teaching hospitals in addis ababa, ethiopia: a case control study. BMC Pregnancy Childbirth. 2013;13(1):1–6.

3 Gao C, Osmundson S, Edwards DRV, Jackson GP, Malin BA, Chen Y. Deep learning predicts extreme preterm birth from electronic health records. J Biomed Inform. 2019;100:103334.

4 Islam, Muhammed N, Mustafina, Sumaiya N, Mahmud, Tahsin, Khan, Nafiz I Machine learning to predict pregnancy outcomes: a systematic review, synthesizing framework and future research agenda

5 Lipschuetz M, Guedalia J, Rottenstreich A, Persky MN, Cohen SM, Kabiri D, Levin G, Yagel S, Unger R, Sompolinsky Y. Prediction of vaginal birth after cesarean deliveries using machine learning. Am J Obstet Gynecol. 2020;222(6):613–1.

6 Li Y-X, Bai Z, Long D-J, Wang H-B, Wu Y-F, Reilly KH, Huang S-R, Ji Y-J. Predicting the success of vaginal birth after caesarean delivery: a retrospective cohort study in china. BMJ Open. 2019;9(5):027807.

Appendices:

R statistical programming code.

```
## ----setup, include=FALSE----------------------------------------------------------------------
knitr::opts_chunk$set(echo = F, warning = F, message = F)


## -------------------------------------------------------------------------------------------------
library(tidyverse)
library(ggplot2)
library(ggthemes)
library(corrplot)
library(reshape2)
library(knitr)
library(broom)
library(caret)
library(leaps)
library(MASS)
library(magrittr)
library(betareg)
library(pscl)
library(gtsummary)
library(nnet)
library(readr)
library(fastDummies)
library(ComplexHeatmap)
library(kableExtra)
library(xgboost)


## -------------------------------------------------------------------------------------------------
MHRD<-read.csv("https://raw.githubusercontent.com/TheSaltyCrab/DATA621_Final/main/Maternal%20Health%20R


## -------------------------------------------------------------------------------------------------
# MHRD<-MHRD%>% mutate(Risk_num = case_when(
#     str_detect(.$RiskLevel, "low risk") ~ "0",
#     str_detect(.$RiskLevel, "mid risk") ~ "1",
#     str_detect(.$RiskLevel, "high risk") ~ "2",
#     TRUE ~ as.character(.$RiskLevel)))
MHRD = MHRD %>% mutate(RiskLevel = factor(RiskLevel, levels = c("low risk", "mid risk", "high risk")))



## -------------------------------------------------------------------------------------------------
set.seed(100)
trainingRows <- sample(1:nrow(MHRD), 0.7*nrow(MHRD))
```

```
training <- MHRD[trainingRows, ]
test <- MHRD[-trainingRows, ]


## ------------------------------------------------------------------------------------------------
corrplot(cor( select_if(MHRD, is.numeric), use = "complete.obs"), tl.col="black", tl.cex=0.6, order='AO


## ------------------------------------------------------------------------------------------------
lst.histogram = list()
for (i in names(MHRD)[1:6]) {
  MHRD.sub = MHRD %>% select(i, "RiskLevel")
  colnames(MHRD.sub) = c("value", "RiskLevel")
  lst.histogram[[i]] = ggplot(aes(value, fill = RiskLevel), data = MHRD.sub) + geom_histogram() + labs(
}
ggpubr::ggarrange(plotlist = lst.histogram, ncol = 2, nrow = 3)


## ------------------------------------------------------------------------------------------------
ggplot(aes(RiskLevel, fill = RiskLevel), data = MHRD) + geom_bar(stat = "count") + labs(x = "Risk Level


## ------------------------------------------------------------------------------------------------
mat.MHRD = MHRD %>% select(-c("RiskLevel")) %>% as.matrix()
rownames(mat.MHRD) = rownames(MHRD)
mat.MHRD = t(mat.MHRD)
tree = hclust(dist(t(mat.MHRD), method = "euclidean"))
tree.groups = cutree(hclust(dist(t(mat.MHRD), method = "euclidean")), k = 6)

mat.risk = MHRD %>% select(c("RiskLevel")) %>% as.matrix()

Heatmap(t(mat.risk), cluster_columns  = tree, cluster_rows = F, col =c("low risk" = "navyblue", "mid ri


## ------------------------------------------------------------------------------------------------
lst.model.results = list()
mn_model<-multinom(RiskLevel ~ ., data=training, trace = F)
# data.frame(summary(mn_model)$coefficients/summary(mn_model)$standard.errors) %>% kable() %>% kableExt

predicted_scores <- predict (mn_model, test, "probs")
predicted_class <- predict (mn_model, test)

lst.model.results[['mm_model']][['accuracy']]  = mean(as.character(predicted_class) == as.character(tes
lst.model.results[['mm_model']][['AIC']] = mn_model$AIC

# print(paste0("accuracy=",mean(as.character(predicted_class) == as.character(test$RiskLevel))))
# print(paste0("AIC=",mn_model$AIC))

as.tibble(t(confusionMatrix(data = predicted_class, reference = test$RiskLevel)$byClass), rownames = "m


## ------------------------------------------------------------------------------------------------
### Model Coefficients
```

```
mn_model2<-multinom(RiskLevel ~ Age + SystolicBP + BS, data=training, trace = F)
# summary(mn_model2)
# data.frame(summary(mn_model2)$coefficients/summary(mn_model2)$standard.errors) %>% kable()



## -------------------------------------------------------------------------------------------
predicted_scores2 <- predict (mn_model2, test, "probs")
predicted_class2 <- predict (mn_model2, test)
# table(predicted_class2,test$RiskLevel)


lst.model.results[['mm_model2']][['accuracy']]  = mean(as.character(predicted_class2) == as.character(te
lst.model.results[['mm_model2']][['AIC']] =mn_model2$AIC

as.tibble(t(confusionMatrix(data = predicted_class2, reference = test$RiskLevel)$byClass), rownames = "r



## ---- echo=FALSE-------------------------------------------------------------------------------

mn_model3<-multinom(RiskLevel ~ BS+SystolicBP, data=training, trace = F)


predicted_scores3 <- predict (mn_model3, test, "probs")
predicted_class3 <- predict (mn_model3, test)
# table(predicted_class3,test$RiskLevel)


lst.model.results[['mn_model3']][['accuracy']]  = mean(as.character(predicted_class3) == as.character(te
lst.model.results[['mn_model3']][['AIC']] = mn_model3$AIC

#table(predicted_class3,test$RiskLevel)
as.tibble(t(confusionMatrix(data = predicted_class3, reference = test$RiskLevel)$byClass), rownames = "r


## -------------------------------------------------------------------------------------------
mn_model4<-multinom(RiskLevel ~ BS, data=training, trace = F)
# summary(mn_model4)
# data.frame(summary(mn_model4)$coefficients/summary(mn_model4)$standard.errors) %>% kable()

predicted_scores4 <- predict (mn_model4, test, "probs")
predicted_class4 <- predict (mn_model4, test)

lst.model.results[['mn_model4']][['accuracy']]  = mean(as.character(predicted_class4) == as.character(te
lst.model.results[['mn_model4']][['AIC']] = mn_model4$AIC


#print(paste0("accuracy=",mean(as.character(predicted_class3) == as.character(test$RiskLevel))))
#print(paste0("AIC=",mn_model3$AIC))


# data.frame(summary(mn_model4)$coefficients/summary(mn_model4)$standard.errors) %>% kable()
# t(confusionMatrix(data = predicted_class4, reference = test$RiskLevel)$byClass) %>% knitr::kable()

as.tibble(t(confusionMatrix(data = predicted_class4, reference = test$RiskLevel)$byClass), rownames = "



## -----------------------------------------------------------------------------------------------
xboost.dat = training
xboost.dat = xboost.dat %>% mutate(high_risk = ifelse(RiskLevel == "high risk", T, F))
xboost.dat = xboost.dat %>% mutate(mid_risk = ifelse(RiskLevel == "mid risk", T, F))

xboost.dat.test = test

xboost.dat.test = xboost.dat.test %>% mutate(high_risk = ifelse(RiskLevel == "high risk", T, F))
xboost.dat.test = xboost.dat.test %>% mutate(mid_risk = ifelse(RiskLevel == "mid risk", T, F))


grid_default <- expand.grid(
  nrounds = 100,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_control <- caret::trainControl(
  method = "none",
  verboseIter = FALSE, # no training log
  allowParallel = TRUE # FALSE for reproducible results
)

xgb_train_high <- caret::train( x = select(xboost.dat.test, -c("RiskLevel", "high_risk", "mid_risk")),
  y = as.numeric(xboost.dat.test$high_risk),
  trControl = train_control,
  tuneGrid = grid_default,
  method = "xgbTree",
  verbose = TRUE
)

xgb_train_mid <- caret::train( x = select(xboost.dat.test, -c("RiskLevel", "high_risk", "mid_risk")),
  y = as.numeric(xboost.dat.test$mid_risk),
  trControl = train_control,
  tuneGrid = grid_default,
  method = "xgbTree",
  verbose = TRUE
)

pred.high <- predict(xgb_train_high, select(xboost.dat.test, -c("RiskLevel", "high_risk", "mid_risk")))

vec.pred.high = c(pred.high > 0.5)
```

```
(confusionMatrix(data = factor(vec.pred.high), reference = factor(xboost.dat.test$high_risk))$byClass) %


#
# dtrain.high <- xgb.DMatrix(data = as.matrix(select(xboost.dat, -c("RiskLevel", "high_risk", "low_risk
#
# dtest.high <- xgb.DMatrix(data = as.matrix(select(xboost.dat.test, -c("RiskLevel", "high_risk", "low_
#
# dtest.low<- xgb.DMatrix(data = as.matrix(select(xboost.dat.test, -c("RiskLevel", "high_risk", "low_ri
#
#
# dtrain.low <- xgb.DMatrix(data = as.matrix(select(xboost.dat, -c("RiskLevel", "high_risk", "low_risk")
#
#
# xboost.high <- xgboost(data = dtrain.high, # the data
#                   nround = 2, # max number of boosting iterations
#                   objective = "binary:logistic")  # the objective function
#
#
# xboost.low <- xgboost(data = dtrain.low, # the data
#                   nround = 2, # max number of boosting iterations
#                   objective = "binary:logistic")  # the objective function
#
# pred.high <- predict(xboost.high, dtest.high)



## ----------------------------------------------------------------------------------------------------

pred.mid <- predict(xgb_train_mid, select(xboost.dat.test, -c("RiskLevel", "high_risk", "mid_risk")))

vec.pred.mid = c(pred.mid > 0.5)

(confusionMatrix(data = factor(vec.pred.mid), reference = factor(xboost.dat.test$mid_risk))$byClass) %>%


## ---- echo=FALSE-----------------------------------------------------------------------------------------
O_model1<-polr(RiskLevel ~ ., data=training, Hess = TRUE)


predicted_scores_ord1 <- predict (O_model1, test, "probs")
predicted_class_ord1 <- predict (O_model1, test)
# table(predicted_class_ord1,test$RiskLevel)

as.tibble(t(confusionMatrix(data = predicted_class_ord1, reference = test$RiskLevel)$byClass), rownames



lst.model.results[['mn_model5']][['accuracy']]  = mean(as.character(predicted_class_ord1) == as.characte
lst.model.results[['mn_model5']][['AIC']] =O_model1$AIC
```

```r
# print(paste0("accuracy=",mean(as.character(predicted_class_ord1) == as.character(test$RiskLevel))))
#print(paste0("AIC=",O_model1$AIC))



## ---- echo=FALSE--------------------------------------------------------------------------
O_model2<-polr(RiskLevel ~ Age + SystolicBP + BS, data=training, Hess = TRUE)


predicted_scores_ord2 <- predict (O_model2, test, "probs")
predicted_class_ord2 <- predict (O_model2, test)
# table(predicted_class_ord2,test$RiskLevel)

as.tibble(t(confusionMatrix(data = predicted_class_ord2, reference = test$RiskLevel)$byClass), rownames


lst.model.results[['ordinal_model2']][['accuracy']]  = mean(as.character(predicted_class_ord2) == as.cha
lst.model.results[['ordinal_model2']][['AIC']] =O_model2$AIC


#print(paste0("accuracy=",mean(as.character(predicted_class_ord2) == as.character(test$RiskLevel))))
#print(paste0("AIC=",O_model2$AIC))



## ---- echo=FALSE--------------------------------------------------------------------------
O_model3<-polr(RiskLevel ~ SystolicBP + BS, data=training, Hess = TRUE)


predicted_scores_ord3 <- predict (O_model3, test, "probs")
predicted_class_ord3 <- predict (O_model3, test)
#table(predicted_class_ord3,test$RiskLevel)

#print(paste0("accuracy=",mean(as.character(predicted_class_ord3) == as.character(test$RiskLevel))))
#print(paste0("AIC=",O_model3$AIC))

as.tibble(t(confusionMatrix(data = predicted_class_ord3, reference = test$RiskLevel)$byClass), rownames

lst.model.results[['ordinal_model3']][['accuracy']]  = mean(as.character(predicted_class_ord3) == as.cha
lst.model.results[['ordinal_model3']][['AIC']] =O_model3$AIC




## ---- echo=FALSE--------------------------------------------------------------------------
O_model4<-polr(RiskLevel ~ BS, data=training, Hess = TRUE)


predicted_scores_ord4 <- predict (O_model4, test, "probs")
predicted_class_ord4 <- predict (O_model4, test)
#table(predicted_class_ord4,test$RiskLevel)
```

```
#print(paste0("accuracy=",mean(as.character(predicted_class_ord4) == as.character(test$RiskLevel))))
#print(paste0("AIC=",O_model4$AIC))

as.tibble(t(confusionMatrix(data = predicted_class_ord4, reference = test$RiskLevel)$byClass), rownames

lst.model.results[['ordinal_model4']][['accuracy']]  = mean(as.character(predicted_class_ord4) == as.cha
lst.model.results[['ordinal_model4']][['AIC']] =O_model4$AIC





## -----------------------------------------------------------------------------------------------------
df.results = as.data.frame(lst.model.results) %>% melt() %>% mutate(var = ifelse(grepl("accuracy", varia
df.results$model = c("mm_model1", "mm_model1", "mm_model2", "mm_model2", "mm_model3", "mm_model3", "mm_m

dcast(data = df.results, formula = model ~ var, value.var = "value") %>% knitr::kable()
```