You will compare results when you include variables that could induce confounding as features entered into the algorithm.

Goal: identify prenatal social and environmental risk factors for childhood overweight/obesity among preterm infants

Women and children were then followed up periodically during infancy and childhood.

You are provided with a dataset containing a number of features, in addition to a binary outcome indicating childhood overweight or obesity vs normal weight

Ow\_obesity is the outcome

**Upsampling vs. downsampling**

<https://towardsdatascience.com/how-to-handle-imbalance-data-and-small-training-sets-in-ml-989f8053531d>

<https://dsp.stackexchange.com/questions/45276/what-is-better-up-or-downsampling>

When the outcome is small and your overall sample is small you upsample, and if the outcome is high you downsample. Watch the class video (last class before spring break) and write out her explanation of up/down sampling

**Use of cross validation in feature selection**

<https://gijopeter.medium.com/cross-validation-for-ml-feature-selection-ffdb71e5a68f>

<https://stats.stackexchange.com/questions/546396/must-we-do-feature-selection-in-cross-validation>

Cross-validation is a means of estimating the performance of a method for fitting a model, rather than of the model itself, so all steps in fitting the model (including feature selection and optimising the hyper-parameters) need to be performed independently in each fold of the cross-validation procedure. If you don't do this, then you will end up with an optimistically biased performance estimate.

**Understanding ROC curves**

<https://towardsdatascience.com/demystifying-roc-curves-df809474529a>

bc = read.csv("/Users/judyfordjuoh/Desktop/Machine Learning/birthcohort\_data.csv") %>%

janitor::clean\_names() %>%

select(-starts\_with("pregnancy")) %>%

mutate(ow\_obesity = as.factor(ow\_obesity)) %>%

mutate(ow\_obesity = recode(ow\_obesity,

"0" = "Typical Weight",

"1" = "Overweight/Obese"))

bc = read.csv("/Users/judyfordjuoh/Desktop/Machine Learning/birthcohort\_data.csv") %>%

janitor::clean\_names() %>%

select(hh\_income, race\_ethnicity,smoking\_preg, mother\_education,child\_human\_biological\_sex, ow\_obesity) %>%

mutate(ow\_obesity = as.factor(ow\_obesity)) %>%

mutate(ow\_obesity = recode(ow\_obesity,

"0" = "Typical Weight",

"1" = "Overweight/Obese"))

### STEP FOUR: Test your final model in the testing dataset

Use the implementation of your model in the testing set to obtain final performance metrics and perform the inference needed to address your research question.

las$finalModel

las\_final <- predict(las, test\_data)

#Get evaluation metrics from test set

confusionMatrix(las\_final, test\_data$ow\_obesity, positive = "Overweight/Obese") #Accuracy #Sensitivity: #Specificity:

#Create ROC Curve for Analysis

pred.prob <- predict(las, test\_data, type = "prob")

analysis <- roc(response = test\_data$ow\_obesity, predictor = pred.prob[,2])

plot(1 - analysis$specificities,analysis$sensitivities,type = "l",

ylab = "Sensitivity",xlab = "1-Specificity",col = "black",lwd = 2,

main = "ROC Curve for Obesity Classification")

abline(a = 0,b = 1)

#Using the test data to make predictions

las2 <- las %>% predict(test\_data)

confusionMatrix(las2,test\_data$ow\_obesity, positive = "Overweight/Obese")

#Obtain predicted probabilities

test.outcome.probs<-predict(las, test\_data, type="prob")

testProbs.rmodel <- data.frame(obs = test\_data$ow\_obesity,

pred.las=test.outcome.probs[,2])

#Create calibration plot

obesity\_PlotData.rmodel<-calibration(obs ~ pred.las, data = testProbs.rmodel, class="Overweight/Obese", cuts=5)

xyplot(obesity\_PlotData.rmodel, auto.key = list(columns = 2))

plot(test.outcome.probs[,2])

# WRONG I THINK

```{r LASSO Calibration plot}

#LASSO Calibration

fitted\_results\_model <- las %>% predict(train\_data)

error\_model <- mean(fitted\_results\_model !=test\_data$ow\_obesity, na.rm = T)

print(paste('Accuracy [las]', 1-error\_model))

testProb <- data.frame(obs = test\_data$ow\_obesity,

pred.logit = error\_model)

calPlotData\_model<- calibration(obs~as.numeric(pred.logit), data = testProb)

xyplot(calPlotData\_model, auto.key = list(columns = 2))

```