Loss estimation homework

Marko Ivanovski

07 May 2022

Libraries

```r
library(ggplot2)
library(dplyr)
set.seed(123)
```

Data generating process

First we define our DGP and loss function.

```r
toy_data <- function(n, seed = NULL) {
  set.seed(seed)
  x <- matrix(rnorm(8 * n), ncol = 8)
  z <- 0.4 * x[,1] - 0.5 * x[,2] + 1.75 * x[,3] - 0.2 * x[,4] + x[,5]
  y <- runif(n) > 1 / (1 + exp(-z))
  return (data.frame(x = x, y = y))
}
log_loss <- function(y, p) {
  -(y * log(p) + (1 - y) * log(1 - p))
}
```

Proxy for the true risk

Next is defining our proxy for the true risk. We know that this risk will be at most different at the 3rd decimal by using the VC theorem. The VC dimensionallity for Logistic Regression is one more that the number of dimensions. If we set n=100000, dim_vc=100001 and some small delta (so that with great probability this holds) we get changes on the third decimal.

```r
ds_dgp <- toy_data(100000, 123)
```

Model loss estimator variability due to test data variability

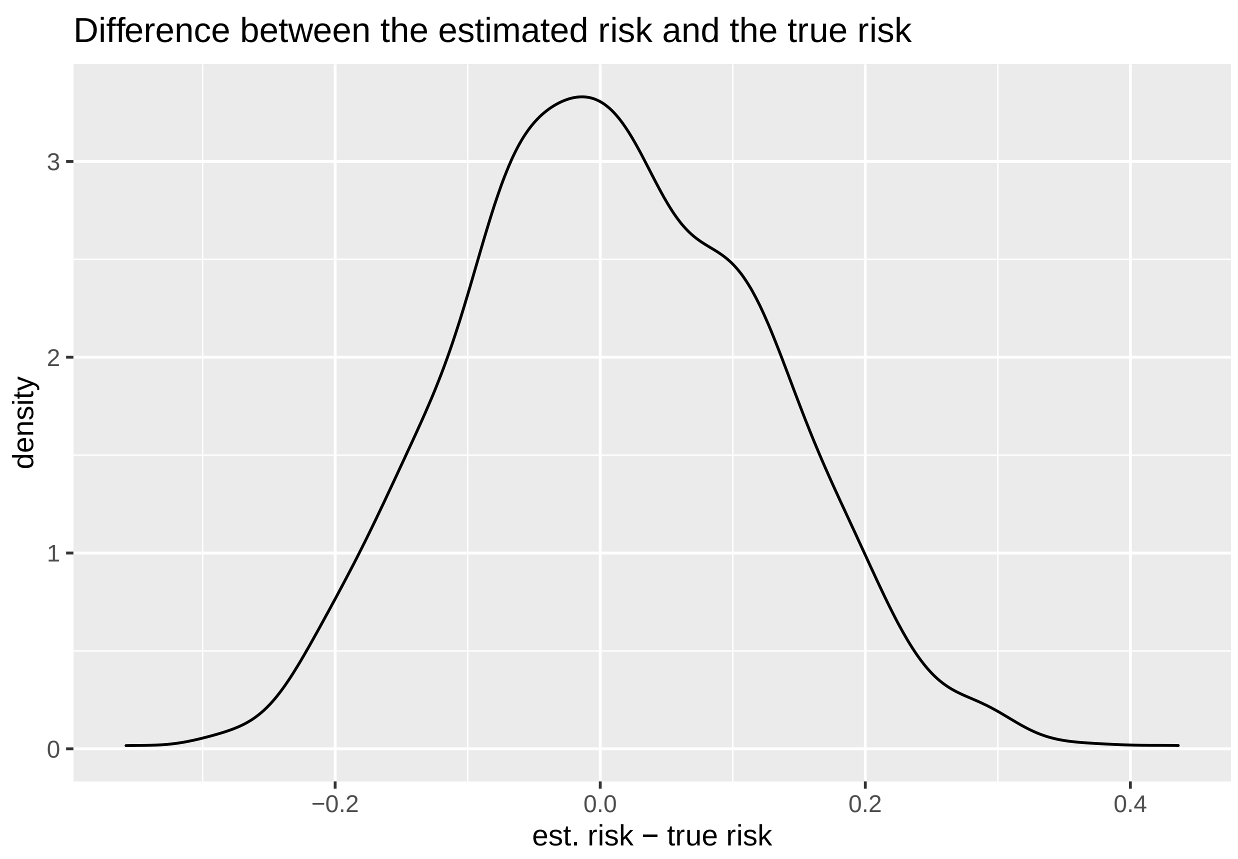
Q: What do we see and what are the implications for practical use? Without experimentation, try to answer: How would these results change if the training set was larger/smaller? How would these results change if the test set was larger/smaller?

A: We have trained a GLM model on a small dataset of 50 instances. Then we have evaluated the 'true risk' using a huge dataset. After this we have evaluated the estimated risk of this model using other 50 instances datasets. First, on average the estimated risk on 50 instances is a good approximation on the true risk since the average difference is really small although still a bit biased. But the variance (standard error) of these estimates is big. We see that these estimates vary quite a lot based on the testing data but by computing the 95% confidence intervals we can capture the true risk approximately 93% of the time.

If we have more training data the true risk proxy will be smaller. I also think the variance of the estimates will slowly start to decrease mainly because we have more training data so it is more probable that we generate similar instances in the test data.

By increasing the test data the variance of the estimates will decrease as well, due to the larger test dataset and increasing the chanse of having more similar test datasets.

```r
toy_dataset = toy_data(50, 0)
model  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=toy_dataset)
y_pred <- predict(model, ds_dgp[,1:8], type="response") #predict on the proxy
true_risk <- 1/length(y_pred)*sum(log_loss(ds_dgp[,9], y_pred))

y_pred <- rep(0.5, nrow(ds_dgp))
true_risk_5050 <- 1/nrow(ds_dgp)*sum(log_loss(ds_dgp[,9], y_pred))


est_risk_minus_true_risk <- c()
standard_errors <- c()
ci_contains_true_risk <- c()
for (i in 1:1000) {
  temp_toy_dataset = toy_data(50, i)
  y_pred <- predict(model, temp_toy_dataset[,1:8], type="response")
  #loss <- 1/length(y_pred)*sum(log_loss(temp_toy_dataset[,9], y_pred))
  losses <- log_loss(temp_toy_dataset[,9], y_pred)

  var_loss = var(losses)
  var_mean_loss = var_loss/length(losses)
  ste = sqrt(var_mean_loss)


  ci_upper_limit =  mean(losses) + (ste*1.96)
  ci_lower_limit =  mean(losses) - (ste*1.96)

  if (true_risk<=ci_upper_limit & ci_lower_limit<=true_risk) {
    ci_contains_true_risk<-append(ci_contains_true_risk, TRUE)
  }
  else{
    ci_contains_true_risk<-append(ci_contains_true_risk, FALSE)
  }

  est_risk_minus_true_risk <- append(est_risk_minus_true_risk, mean(losses)-true_risk)
  standard_errors <- append(standard_errors, ste)

}

ggplot(data = data.frame(x=est_risk_minus_true_risk), aes(x = x)) +
  geom_density() +
  ggtitle('Difference between the estimated risk and the true risk')+
  xlab("est. risk - true risk")+
  ylab("density")
```



```
print(paste("True risk proxy:", round(true_risk, digits = 4)))
```

```
## [1] "True risk proxy: 0.5701"
```

```
print(paste("Mean difference between the estimated risk with 50 observations and with 100000:", round(me
```

```
## [1] "Mean difference between the estimated risk with 50 observations and with 100000: 0.0095"
```

```
print(paste("0.5-0.5 baseline true risk:", round(true_risk_5050,4)))
```

```
## [1] "0.5-0.5 baseline true risk: 0.6931"
```

```
print(paste("Median standard error:", round(median(standard_errors),4)))
```

```
## [1] "Median standard error: 0.1124"
```

```
print(paste("Percentage of CI that contains true risk:", sum(ci_contains_true_risk)/length(ci_contains_t
```

```
## [1] "Percentage of CI that contains true risk: 92.6"
```

Overestimation of the deployed model's risk

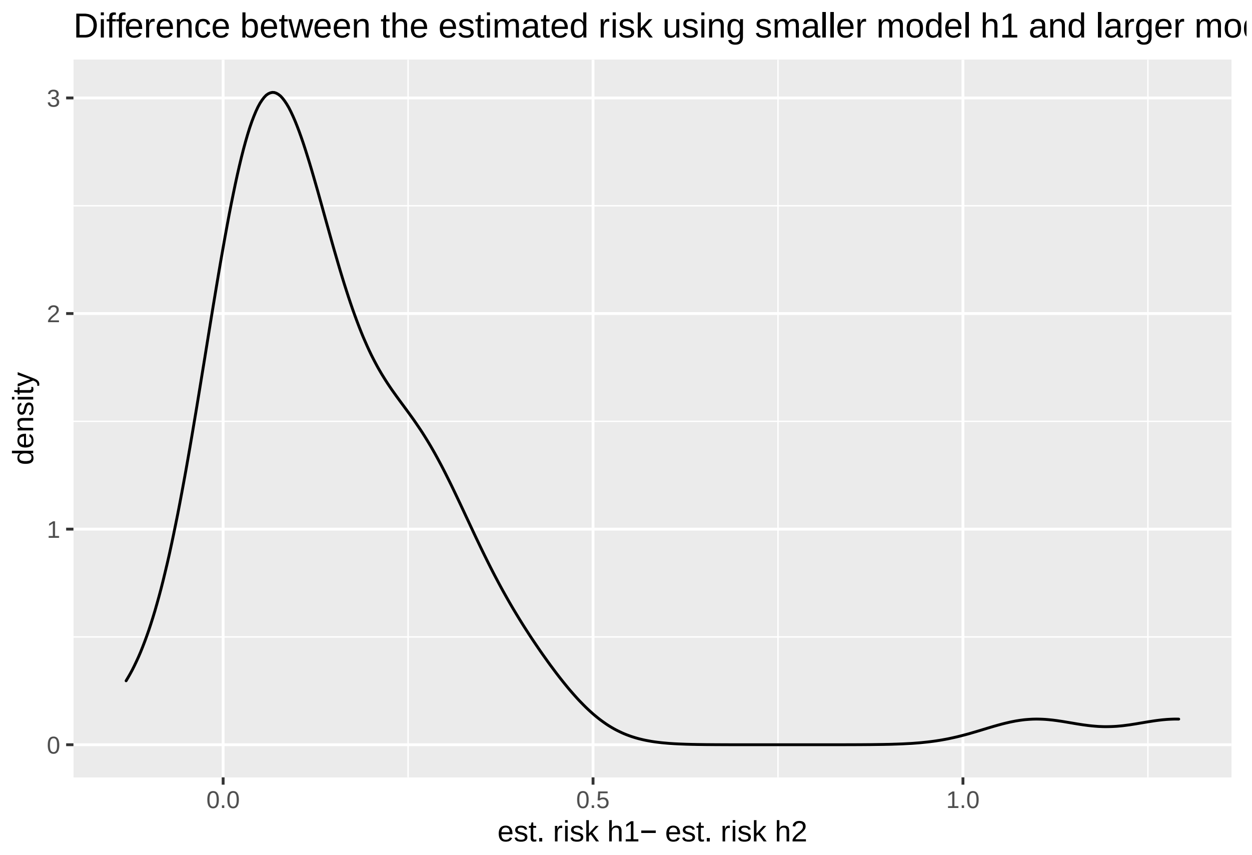
Q: What do we see and what are the implications for practical use? Without experimentation, try to answer: How would these results change if the data sets were larger/smaller?

A: Surely the estimated empirical risk decreases by increasing the training data (we reduce the bias),this experiment proves that. If we would to increase more data the difference between the estimated risks between h1 and h2 would be smaller since the model h1 will see enough data to learn the DGP and even though h2 has more data they both will have learned the DGP well, hence the smaller difference.

```
risk_h1_minus_h2<-c()
for (i in 1:50){
  toy_dataset1 = toy_data(50, i+1000)
  toy_dataset2 = toy_data(50, i+50)

  h1  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=toy_dataset1)
  h2  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=rbind(toy_dataset1, toy_datas

  y_pred <- predict(h1, ds_dgp[,1:8], type="response") #predict on the proxy
  true_risk_model_h1 <- 1/length(y_pred)*sum(log_loss(ds_dgp[,9], y_pred))

  y_pred <- predict(h2, ds_dgp[,1:8], type="response") #predict on the proxy
  true_risk_model_h2 <- 1/length(y_pred)*sum(log_loss(ds_dgp[,9], y_pred))

  risk_h1_minus_h2 <- append(risk_h1_minus_h2, true_risk_model_h1-true_risk_model_h2)
}

print(summary(risk_h1_minus_h2))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.13117  0.03456  0.09899  0.17416  0.25668  1.29169
```

```
ggplot(data = data.frame(x=risk_h1_minus_h2), aes(x = x)) +
  geom_density() +
  ggtitle('Difference between the estimated risk using smaller model h1 and larger model h2')+
  xlab("est. risk h1- est. risk h2")+
  ylab("density")
```



Loss estimator variability due to split variability

Q: What do we see and what are the implications for practical use? Without experimentation, try to answer: How would these results change if the data set was larger/smaller? How would these results change if the proportion of training data was larger/smaller?

A: We have trained a model on all 100 data points and calculated its 'true' risk by evaluating it on the huge dataset. Then we used holdout estimation (multiple times) **on the same dataset** and trained on 50%, then tested on 50% and said this will be the error on of the model when trained on all 100% data. Now from the experiment we saw that this estimate is biased since the loss will be in reality SMALLER. The median standard error is also big due to the randomness of the train-test split. The standard error also suggests that we won't always overestimate the risk, sometimes we will even underestimate it.

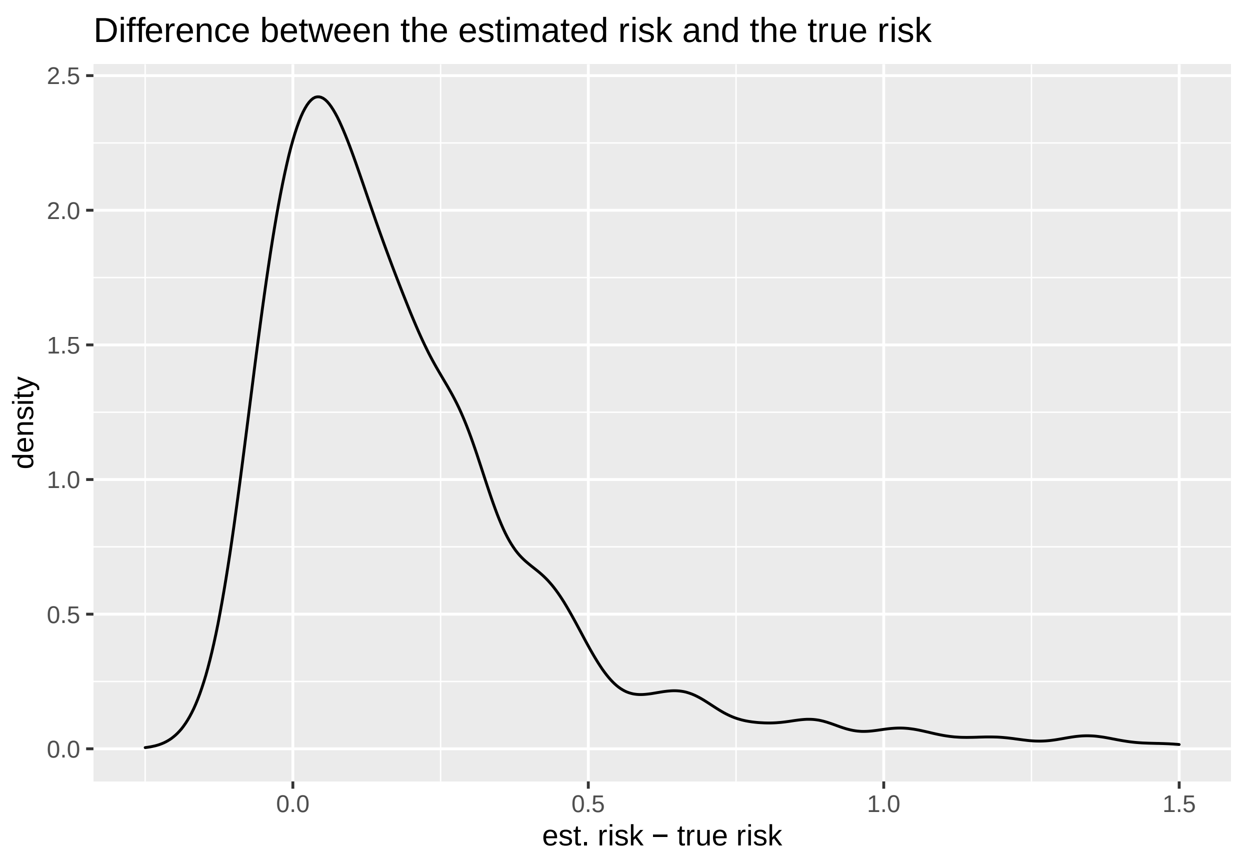
By having a larger dataset the distribution will become thinner and the difference between estimated and true will get smaller. This is due to the fact that the model will be able to learn the DGP better and better so it will get closer to the true risk. The variance will also decrease because yet again the model will be better so the train test split will not influence the estimation of the risk by a lot.

By decreasing the training proportion our model will have very few training instances, consequently the risk estimates will be bigger and the variance will be bigger as well.

By increasing the training proportion our model will fit the data better, the risk estimates will get closer to the true risk (smaller bias). But because we have little test data the variance will increase because it will hugely depend on the split.

```
toy_dataset0 = toy_data(100, 456)
```

```r
h0  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=toy_dataset0)
y_pred <- predict(h0, ds_dgp[,1:8], type="response") #predict on the proxy
true_risk_model_h0<- 1/length(y_pred)*sum(log_loss(ds_dgp[,9], y_pred))

est_risk_minus_true_risk<-c()
ci_contains_true_risk<-c()
standard_errors<-c()
#50,50 holdout split
for (i in 1:1000){
  split_dummy <- sample(c(rep(0, 0.5 * nrow(toy_dataset0)),  # Create dummy for splitting
                        rep(1, 0.5 * nrow(toy_dataset0))))
  toy_dataset0_train <- toy_dataset0[split_dummy == 0, ]
  toy_dataset0_test <- toy_dataset0[split_dummy == 1, ]

  h <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=toy_dataset0_train)
  y_pred <- predict(h, toy_dataset0_test, type="response") #predict on the proxy
  losses<- log_loss(toy_dataset0_test[,9], y_pred)

  var_loss = var(losses)
  var_mean_loss = var_loss/length(losses)
  ste = sqrt(var_mean_loss)

  ci_upper_limit =  mean(losses) + (ste*1.96)
  ci_lower_limit =  mean(losses) - (ste*1.96)

  if (true_risk_model_h0<=ci_upper_limit & ci_lower_limit<=true_risk_model_h0) {
    ci_contains_true_risk<-append(ci_contains_true_risk, TRUE)
  }
  else{
    ci_contains_true_risk<-append(ci_contains_true_risk, FALSE)
  }

  est_risk_minus_true_risk <- append(est_risk_minus_true_risk, mean(losses)-true_risk_model_h0)
  standard_errors <- append(standard_errors, ste)
}

ggplot(data = data.frame(x=est_risk_minus_true_risk), aes(x = x)) +
  geom_density() +
  ggtitle('Difference between the estimated risk and the true risk')+
  xlab("est. risk - true risk")+
  ylab("density")+
  xlim(-0.25, 1.5)
```



```r
print(paste("True risk proxy:", true_risk))
```

```
## [1] "True risk proxy: 0.570068521765031"
```

```r
print(paste("Mean difference:", mean(est_risk_minus_true_risk)))
```

```
## [1] "Mean difference: 0.347896813547576"
```

```r
print(paste("Median standard error:", median(standard_errors)))
```

```
## [1] "Median standard error: 0.13711464005125"
```

```r
print(paste("Percentage of CI that contains true risk:", sum(ci_contains_true_risk)/length(ci_contains_t
```

```
## [1] "Percentage of CI that contains true risk: 84.8"
```

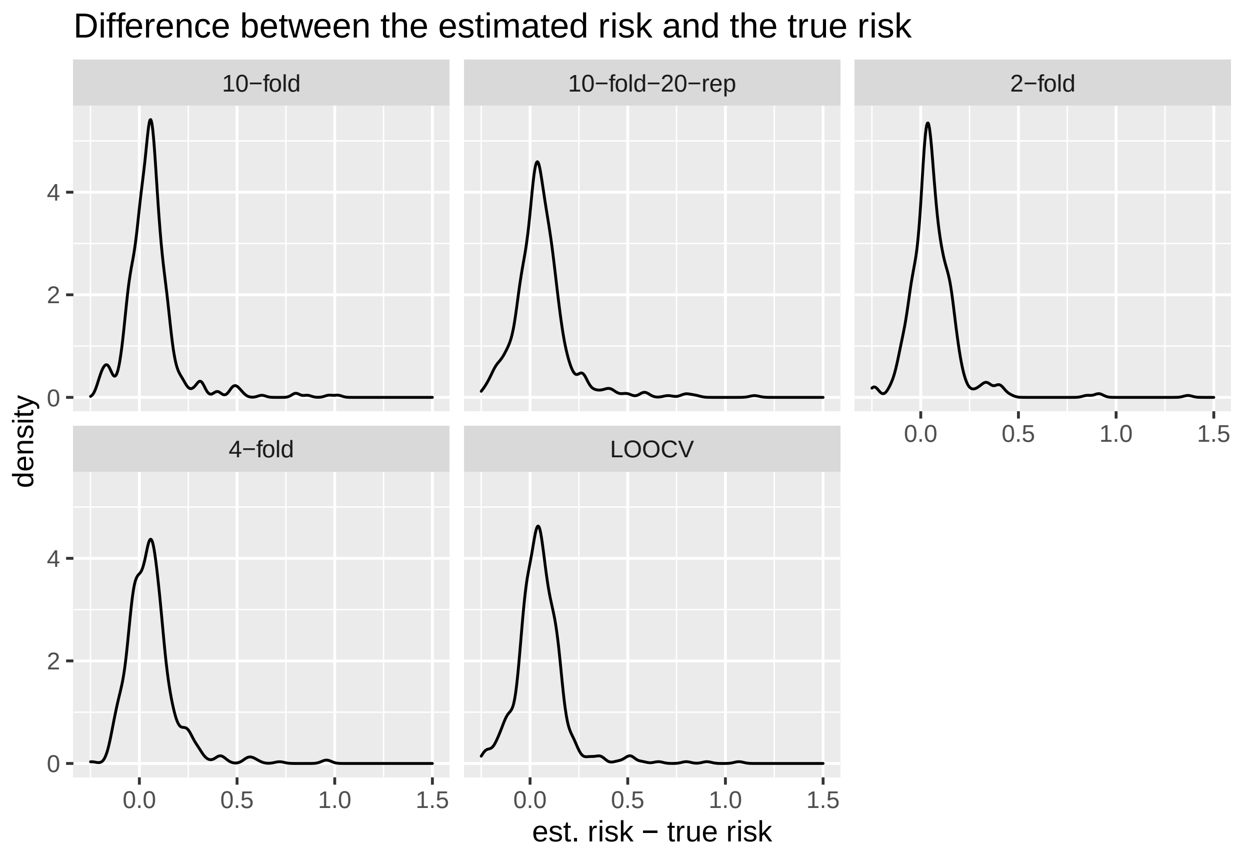
Cross-validation

Q: What do we see and what are the implications for practical use?

A: Compared to the normal train-test split we observe how the bias and variability decreases. LOOCV has the lowest bias and low median standard error which is surprising. I expected that for the cost of lowering the bias we will increase the variance since we only have one test example so the test sets vary quite a lot. Next is 10-CV which has second lowest bias and finally the 4-CV with the biggest median standard error. In this experiment we see how 1-fold cross validation is the best estimator (gets closest to the real value and has the smallest variability).

```r
cross_val <- function(dataset, k){
  step=nrow(dataset)/k
  previous_start=1

  all_losses <- c()
  for (i in 1:k){
    train_fold = dataset[-(previous_start:(step*i)),]
    test_fold = dataset[previous_start:(step*i),]

    h0  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=train_fold)
    #print(nrow(test_fold))

    y_pred <- predict(h0, test_fold[,1:8], type="response") #predict on the proxy
    split_losses<- log_loss(test_fold[,9], y_pred)
    all_losses <- append(all_losses, split_losses)
    #all_losses <- split_losses

    previous_start = (i*step)+1
  }


  var_loss = var(all_losses)
  var_mean_loss = var_loss/length(all_losses)
  ste = sqrt(var_mean_loss)

  ci_upper_limit =  mean(all_losses) + (ste*1.96)
  ci_lower_limit =  mean(all_losses) - (ste*1.96)


  return (as.numeric(c(mean(all_losses), ste, ci_lower_limit, ci_upper_limit)))
}

repeated_cross_val <- function(dataset, k, n){
  step=nrow(dataset)/k

  all_losses <- rep(0, nrow(dataset))
  for (j in 1:n){
    shuffle <- sample(1:nrow(dataset))
    shuffled_data <- dataset[shuffle, ]
    previous_start <- 1
    all_losses_in_rep <- c()

    for (i in 1:k){
      train_fold = shuffled_data[-(previous_start:(step*i)),]
      test_fold = shuffled_data[previous_start:(step*i),]

      h0  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=train_fold)
      #print(nrow(test_fold))
      y_pred <- predict(h0, test_fold[,1:8], type="response") #predict on the proxy
      split_losses<- log_loss(test_fold[,9], y_pred)


      all_losses_in_rep <- append(all_losses_in_rep, split_losses)
      previous_start = (i*step)+1

    }


    for (shuff_index in 1:length(shuffle)){
      all_losses[shuff_index] <- all_losses[shuff_index] + all_losses_in_rep[shuffle==shuff_index]
      #print(shuffle[shuff_index])
    }

  }


  all_losses <- all_losses/n
  var_loss = var(all_losses)
  var_mean_loss = var_loss/length(all_losses)
  ste = sqrt(var_mean_loss)

  ci_upper_limit =  mean(all_losses) + (ste*1.96)
  ci_lower_limit =  mean(all_losses) - (ste*1.96)

  return (as.numeric(c(mean(all_losses), ste, ci_lower_limit, ci_upper_limit)))
}

est1 <- as.data.frame(matrix(ncol=3, nrow=0))
names(est1) <- c("est_minus_real","ste", "in_ci")
est2 <- as.data.frame(matrix(ncol=3, nrow=0))
names(est2) <- c("est_minus_real","ste", "in_ci")
est3 <- as.data.frame(matrix(ncol=3, nrow=0))
names(est3) <- c("est_minus_real","ste", "in_ci")
est4 <- as.data.frame(matrix(ncol=3, nrow=0))
names(est4) <- c("est_minus_real","ste", "in_ci")
est5 <- as.data.frame(matrix(ncol=3, nrow=0))
names(est5) <- c("est_minus_real","ste", "in_ci")

for (j in 1:500){
  toy_dataset0 = toy_data(100, j+1000)
  h0  <- glm(y ~ x.1+x.2+x.3+x.4+x.5+x.6+x.7+x.8, family="binomial", data=toy_dataset0)
  y_pred <- predict(h0, ds_dgp[,1:8], type="response") #predict on the proxy
  true_risk_model_h0<- 1/length(y_pred)*sum(log_loss(ds_dgp[,9], y_pred))

  #shuffled_data= data[sample(1:nrow(data)), ]

  #2-fold cross-validation
  results <- cross_val(toy_dataset0, 2)
  if (results[3]<=true_risk_model_h0 & true_risk_model_h0<=results[4]){
    est1[j,] <- c(results[1]-true_risk_model_h0, results[2], TRUE)
  }
  else{
    est1[j,] <- c(results[1]-true_risk_model_h0, results[2], FALSE)
  }

  #LOOCV
  results <- cross_val(toy_dataset0, 100)
  if (results[3]<=true_risk_model_h0 & true_risk_model_h0<=results[4]){
    est2[j,] <- c(results[1]-true_risk_model_h0, results[2], TRUE)
  }
  else{
    est2[j,] <- c(results[1]-true_risk_model_h0, results[2], FALSE)
  }

  #4-fold cross-validation
  results <- cross_val(toy_dataset0, 4)
  if (results[3]<=true_risk_model_h0 & true_risk_model_h0<=results[4]){
    est3[j,] <- c(results[1]-true_risk_model_h0, results[2], TRUE)
  }
  else{
    est3[j,] <- c(results[1]-true_risk_model_h0, results[2], FALSE)
  }

  #10-fold cross-validation
  results <- cross_val(toy_dataset0, 10)
  if (results[3]<=true_risk_model_h0 & true_risk_model_h0<=results[4]){
    est4[j,] <- c(results[1]-true_risk_model_h0, results[2], TRUE)
  }
  else{
    est4[j,] <- c(results[1]-true_risk_model_h0, results[2], FALSE)
  }

  #20 reps of 10-fold cross-validation
  #estimator5_difference <- 0
  #for (i in 1:20){
  #  toy_dataset0 = toy_dataset0[sample(1:nrow(toy_dataset0)), ]
  #  estimator5_difference <- estimator5_difference + (cross_val(toy_dataset0, 10, h0) - true_risk_mode
  #}
  #estimator5_difference <- estimator5_difference/20
  results <- repeated_cross_val(toy_dataset0, 10, 20)
  if (results[3]<=true_risk_model_h0 & true_risk_model_h0<=results[4]){
    est5[j,] <- c(results[1]-true_risk_model_h0, results[2], TRUE)
  }
  else{
    est5[j,] <- c(results[1]-true_risk_model_h0, results[2], FALSE)
  }
}

df1 <- data.frame(x = c( c(est1$est_minus_real),
                         c(est2$est_minus_real),
                         c(est3$est_minus_real),
                         c(est4$est_minus_real),
                         c(est5$est_minus_real)),
                  type = c(rep("2-fold", 100),
```

```r
                           rep("LOOCV", 100),
                           rep("4-fold", 100),
                           rep("10-fold", 100),
                           rep("10-fold-20-rep", 100)
                           ))


ggplot(data = df1, aes(x = x)) +
  geom_density() +
  facet_wrap(~type) +
  ggtitle('Difference between the estimated risk and the true risk')+
  xlab("est. risk - true risk")+
  ylab("density")+
  xlim(-0.25, 1.5)
```



```r
#print("2-fold cross-validation")
#print(paste("Mean difference:", mean(est1$est_minus_real)))
#print(paste("Median standard error:", median(est1$ste)))
#print(paste("Percentage of CI that contains true risk:", #sum(est1$in_ci)/length(est1$in_ci)*100 ))

print("LOOCV")
```

```
## [1] "LOOCV"
```

```r
print(paste("Mean difference:", mean(est2$est_minus_real)))
```

```
## [1] "Mean difference: 0.00814001128220487"
```

```r
print(paste("Median standard error:", median(est2$ste)))
```

```
## [1] "Median standard error: 0.0747237806561526"
```

```r
print(paste("Percentage of CI that contains true risk:", sum(est2$in_ci)/length(est2$in_ci)*100 ))
```

```
## [1] "Percentage of CI that contains true risk: 91.2"
```

```r
print("4-fold cross-validation")
```

```
## [1] "4-fold cross-validation"
```

```r
print(paste("Mean difference:", mean(est3$est_minus_real)))
```

```
## [1] "Mean difference: 0.0492774295021027"
```

```r
print(paste("Median standard error:", median(est3$ste)))
```

```
## [1] "Median standard error: 0.0834285257797925"
```

```r
print(paste("Percentage of CI that contains true risk:", sum(est3$in_ci)/length(est3$in_ci)*100 ))
```

```
## [1] "Percentage of CI that contains true risk: 90"
```

```r
print("10-fold cross-validation")
```

```
## [1] "10-fold cross-validation"
```

```r
print(paste("Mean difference:", mean(est4$est_minus_real)))
```

```
## [1] "Mean difference: 0.0181851973345907"
```

```r
print(paste("Median standard error:", median(est4$ste)))
```

```
## [1] "Median standard error: 0.0764157258088809"
```

```r
print(paste("Percentage of CI that contains true risk:", sum(est4$in_ci)/length(est4$in_ci)*100 ))
```

```
## [1] "Percentage of CI that contains true risk: 91.6"
```

```r
print("20 reps 10-fold cross-validation")
```

```
## [1] "20 reps 10-fold cross-validation"
```

```r
print(paste("Mean difference:", mean(est5$est_minus_real)))
```

```
## [1] "Mean difference: 0.0190812781591558"
```

```r
print(paste("Median standard error:", median(est5$ste)))
```

```
## [1] "Median standard error: 0.0760046376980785"
```

```r
print(paste("Percentage of CI that contains true risk:", sum(est5$in_ci)/length(est5$in_ci)*100 ))
```

```
## [1] "Percentage of CI that contains true risk: 91.6"
```

A different scenario

I think that by increasing k in cross validation we will always get lower bias on the cost of increasing the variance, although the variance might not be bigger as seen in the last experiment. This also comes at expense of computation since LOOCV takes the most time. Consequently lower k will always have bigger bias but lower variance. This does not mean that we should always use the largest k. In my opinion if we have a lot of data using LOOCV is not necessary.