> head(max(average\_acc\_history$validation\_acc))

[1] 0.9543677

> # Evaluate on Testset

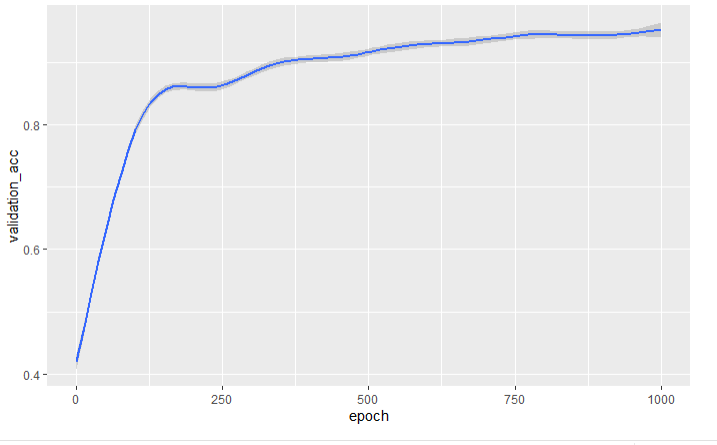
> eval <- evaluate(model, test\_data, test\_targets, verbose = 1)

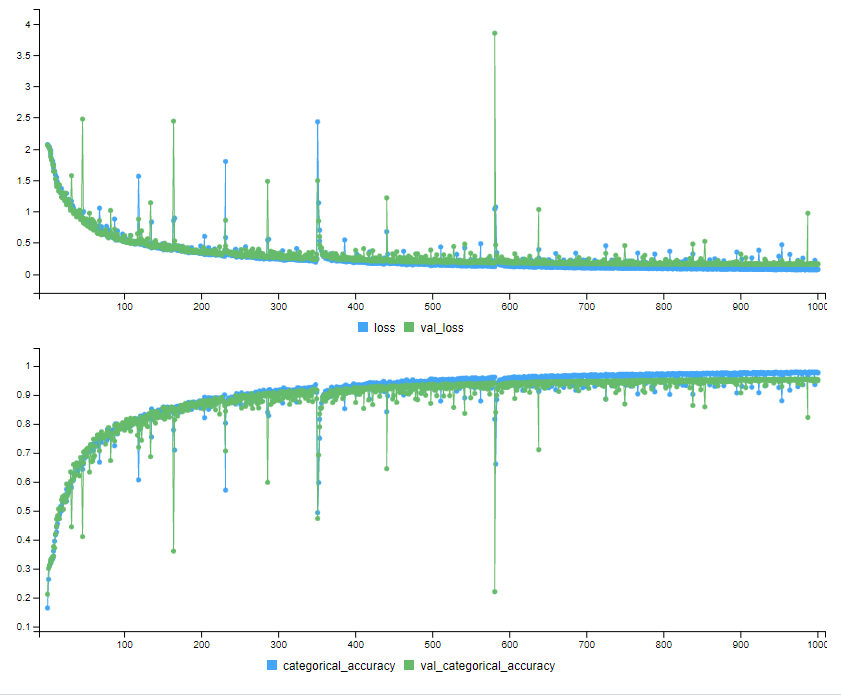
423/423 [==============================] - 2s 4ms/step - loss: 1.3329 - categorical\_accuracy: 0.7621

> head(eval)

loss categorical\_accuracy

1.332938 0.762109





set.seed(5)

preprocRecipe <-

recipe(status ~., data = data) %>%

step\_dummy(all\_nominal(), -status, one\_hot = TRUE) %>%

step\_range(all\_predictors(), -all\_nominal(), min = 0, max = 1)%>%

step\_smote(status, over\_ratio = 1) %>%

# step\_downsample(status, under\_ratio = 1, skip=TRUE) %>%

# step\_smote(status, over\_ratio = 1, skip=TRUE) %>%

# step\_smotenc(status, over\_ratio = 1) %>%

# step\_adasyn(status, over\_ratio = 1) %>%

# step\_nearmiss(status, over\_ratio = 1) %>%

step\_dummy(status, one\_hot = TRUE)# %>%

Model:

# Function to build the model

build\_model <- function() {

model <- keras\_model\_sequential() %>%

#layer\_batch\_normalization(axis = -1L, input\_shape = dim(train\_data)[[2]]) %>%

layer\_dense(units = 128, activation = "relu", input\_shape = dim(train\_data)[[2]]) %>%

layer\_dense(units = 128, activation = "relu") %>%

layer\_dense(units = 128, activation = "relu") %>%

layer\_dense(units = 128, activation = "relu") %>%

layer\_dense(units = 128, activation = "relu") %>%

#layer\_dropout(0.3) %>%

layer\_dense(units = 8, activation = "softmax")

model %>% compile(

#optimizer = optimizer\_sgd(learning\_rate = 0.1),

optimizer = optimizer\_adam(learning\_rate = 0.1),

loss = "categorical\_crossentropy",

metrics = "categorical\_accuracy"

)

}