####reference

# <https://www.kaggle.com/sanchitakarmakar/mushroom-classification-99-75-ac-rf-pca>

# <https://www.kaggle.com/andreshg/mushroom-s-h2o-automl-and-clasic-models-auc-1-0>

# <https://www.kaggle.com/tosinabase/mushroom-classification-tree-methods-comparison>

# <http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software>

# <https://www.displayr.com/how-to-create-a-correlation-matrix-in-r/>

# <https://stackoverflow.com/questions/27528907/how-to-convert-data-frame-column-from-factor-to-numeric/27528953>

# <https://towardsdatascience.com/how-to-create-a-correlation-matrix-with-too-many-variables-309cc0c0a57>

#####

# <https://www.machinelearningplus.com/machine-learning/caret-package/>

# <http://topepo.github.io/caret/available-models.html>

# <http://topepo.github.io/caret/train-models-by-tag.html>

#mehtods glm, naive bayes -> qda (one specific of naive bayes) -> lda (simplification of qda)

#predict(train\_qda, test\_set, type = "prob") %>% head()

#predict(train\_qda, test\_set) %>% head()

# rpart to create regression tree, can show descision flow chart

# train\_rpart <- train(margin ~ ., method = "rpart", tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)), data = polls\_2008)

# can tune CP here

# Classification trees, or decision trees, are used in prediction problems where the outcome is categorical.

# Decision trees form predictions by calculating which class is the most common among the training set observations

# within the partition, rather than taking the average in each partition.

# Then can introduce random forest

# Random Forest

# fit <- train(y ~ ., method = "rRborist", tuneGrid = data.frame, data = )

# use mtry to select the features used in prediction by random

# use variable importance to see which variable is important

# carat package reference

# http://topepo.github.io/caret/available-models.html

# http://topepo.github.io/caret/train-models-by-tag.html

# y\_hat\_glm <- predict(train\_glm, mnist\_27$test, type = "raw")

# y\_hat\_knn <- predict(train\_knn, mnist\_27$test, type = "raw")

# ggplot(train\_knn\_cv, highlight = TRUE)

# train can do the parameter maximization

# ggplot(train\_knn, highlight = TRUE)

# train\_knn$bestTune

# train\_knn$finalModel

# can use change control

#control <- trainControl(method = "cv", number = 10, p = .9)

#train\_knn\_cv <- train(y ~ ., method = "knn",

# data = mnist\_27$train,

# tuneGrid = data.frame(k = seq(9, 71, 2)),

#ggplot(train\_knn\_cv, highlight = TRUE)

# train\_knn$results %>%

# ggplot(aes(x = k, y = Accuracy)) +

# geom\_line() +

# geom\_point() +

# geom\_errorbar(aes(x = k,

# ymin = Accuracy - AccuracySD,

# ymax = Accuracy + AccuracySD)) trControl = control)

# knn then can use gamLoess

# pre-processing to transform predictr and remove predictor not useful

# are highly correlated with others, have very few non-unique values, or have close to zero variation

# caret package : nzv <- nearZeroVar(x)

# caret need to have column name

# variable importance importance()

# ensemble different model to one