**SCALING INPUTS**

Q: When do we have to scale the input variables? Do we do this for all the classifiers like KNN, Decision Tree, Naive Bayes, and Logistic Regression? Do we not scale the input variables unless told in the question?

Ans: Firstly, only the methods that use distance such as KNN or K means (for un-supervised learning) then we should standardize the inputs. Other than that, we normally keep the inputs as original.

For the practice paper, if you read the questions carefully, you can realize that only in Q19, the question explicitly requires us to standardize the inputs: "Use all the observations and **standardized features** to form the best k-NN classifier". Similarly, in our midterm test, it was the same.

Hence, for our test and exam, only when question specify, you should do it.  
  
However, if you have dataset where the **inputs are all of similar magnitude**, then scaling or not is **not a matter**.

**SPLITING DATA INTO TRAIN AND TEST**

Q: When do we split the date into test and train datasets? Do we only split it to train and test when the questions specifically ask for it?  
Ans: If you are asking for a test or an exam, then Yes, when the question asks for it.  
If you are asking for the general, then people normally do the split (ratio 80:20) when they try to find a good classifier.

**ROC AND AUC**

Q: When the question asked us to derive the AUC value of the ROC curve of the classifier, do we use the entire data set to fit the model and then use it again to predict the predicted outcome and compare it with the expected outcome? Or do we use the test dataset that we want to predict the outcome to determine the AUC value for e.g. in the practice paper for final exam, the dataset of Life\_expectancy = 83, Adult\_mortality = 57, infant\_deaths = 2, and Alcohol = 3? 

Ans: The point given with Life\_expectancy = 83, Adult\_mortality = 57, infant\_deaths = 2, and Alcohol = 3

is a new data point, not a test point. A test point is the point with the inputs and response BOTH ARE GIVEN. The point given above doesn't have response.

Back to your question that if we split the original dataset into train set and test set, then should we use test set or train set to get the ROC and AUC?   
The answer is we normally use the test set.

Q: How do we plot the AUC and ROC curve for KNN? Do we have to use as.numeric()?  
Ans: The output of knn() is normally the class label, 0 and 1. If you want to use this to plot ROC curve then you must use as.numeric() function. However, if you want to get the probability from knn() rather than the class label, then you need:

*pred.knn= knn(train.X, test.X, train.y, k = 1, prob=TRUE)*

*pred.prob <- attr(pred.knn, "prob")*  # this is to get the probabilities of winning class from KNN, then

*prediction(pred.prob, actual.class) to get ROC curve*

If you don’t add “prob = TRUE”, then the output of knn() is the class label (with 0 and 1), and you need to use *as.numeric()* function to transform "pred.nn" from factor with 0 and 1 class labels to 0 and 1 numeric by:

*pred.knn = knn(train.X, test.X, train.y, k = 1)*

*pred = as.numeric(paste(pred.knn))*

then

*prediction(pred, actual.class) #to get ROC curve,*

*#actual.class is the column of the response in the test data with 1 and 0*

Let's try to understand how ROC curve is plotted, then we can know the difference:

(1) the ROC curve uses the probability of response = yes, Pr (Y = 1) from model.

(2) for each value of threshold delta, like delta = 0.1, compare Pr(Y = 1) vs delta, if a data point has Pr(Y= 1) > delta then that point is predicted as "Yes" or "1"; otherwise that point will be classified as "No" or "0".

(3) With the classification as in (2) be applied to all the data points for each value of delta, the value of TPR and FPR are calculated for each delta value.

(4) With a series of delta from 0 to 1 (where R will generate this series on its own), the values of TPR and FPR will be calculates for each delta in this series.

(5) ROC curve is plotted.

That's how the ROC curve is generated.

Now, back to KNN: when we use function knn() and do not specify "prob = TRUE", the output we get are the labels ("0" and "1" ) for each point. When we use as.numeric() to transform the labels "0" and "1" into numeric, we get 0 and 1. After that, we use those 0s and 1s to plot the ROC curve. Then, at the step 1 above, R will treat 0 and 1 as the probability Pr(Y = 1) for data points. That means, for all the data points given, each of them has probability Pr(Y = 1) is either 0 or 1, no other values. That's why the curve has only 1 bend for this case.

However, if you run knn() with "prob = TRUE", you will be able to get the probability of response be the WINNING CATEGORY.

When we run knn() with "prob = TRUE", we will be able to get the probability of response variable equal to the WINNING category.

For example, Y has two category: Yes and No, then the output of

pred.knn = knn(... , prob = TRUE)

winning.prob = attr(pred.knn, "prob")

is a series of winning probabilities which are: the probability that Y = "Yes" if the point is classified as Yes and probability that Y = "No" is the point is classified as No.

To get the probability of Yes for every data point in this case, you might want to run (here, I'm assuming the response has "yes" and "no"):

pred.knn = knn(train.x, test.x, cl = train.y, k = 1) # "prob = TRUE" is NOT added --> to get the class labels

pred.knn.prob= knn(train.x, text.x, cl = train.y, k = 1, prob = TRUE) # --> to get the probabilities of winning labels

winning.prob = attr(pred.knn.prob, "prob") # to extract the winning probabilities prob = numeric(0, n) # n is the length of "test.y" used in knn() above

for (i in 1:n) { prob[i] = ifelse(pred.knn[i] == "yes", winning.prob[i], 1-winning.prob[i]) }

Then, "prob" will be the probabilities of response be "Yes" for every point in the test set. With this vector "prob", you can use to plot the ROC curve and calculate its AUC values.

Q: With regards to the ROC curve, I note that with respect to naive Bayes; when the 2nd argument for the **prediction** function is being passed; it is usually in the form of a vector of Booleans derived from df$response == class1.

*E.g.*

*response is the column in the df dataframe we are interested in.*

*response has 2 variables - "yes" and "no"*

*...*

*predict\_prob <- predict(naive\_bayes\_model,test\_X,'raw')*

***prediction( predict\_prob, df$response == "yes")***

*...*

However, for other machine learning models like logistic regression and decision trees, the 2nd argument isn't a vector of True/False values (at least not explicitly). Can I check if this is something only unique to Naive Bayes and ROC or if its use of other models does not severely affect the outcome of the ROC.  
Ans:

Function *prediction(y-hat, df$y)* requires  **y** in 0 and 1 format.

Hence, if the response *df$y* has categories with characters like "yes" and "no", then using *df$y == "yes"*will help to transform the response into 0 and 1 accordingly.

However, if the response *y* itself is already having 0 and 1, then you just need to have it as

*prediction (y-hat, df$y)*

This is not only for Naive Bayes but for any other classifier.

**LINEAR REGRESSION**

**Q:** From the summary of a linear regression model, should we use multiple R-squared or adjusted R-squared to derive the goodness of fit of the model? Or both are good?  
Ans: With only the model, then we use R^2 to check the goodness of fit. However, if you have few models to compare with each other, to see which one is better, then we use adjusted R^2.

**Q:** Regarding the assumptions on response for a linear model, we were told that if the histogram plotted is right skewed, using log would be better. Does this mean that we should always check the histogram before using lm() function?

Ans: Yes. The response variable should be symmetric so that the linear model would be good. Hence, checking the distribution of the response before lm() is better.  
If the distribution of y-the response is highly right skewed, then taking transformation like log(y) be the response is recommended. However, there are few points:

1. taking log() is applicable if y is positive.
2. Taking log(y) will not 100% surely make the linear model be good. Just recommended to try with log(y) only.

**Q: what is predicted values and fitted values? are they the same?**

Ans: fitted values are the predicted values for the points in the train data (used to form the model.

When you have a new point, like test point, you use predict() function to get the predicted value of the point.

**ASSOCIATION BETWEEN VARIABLES**

**Q:** How to check the association between 2 categorical variables where one has more than 2 categories? Can odds ration be still used for this case?

Ans: odds ratio is used for 2x2 table, not for larger table. Let’s assume X has 3 categories and Y – response has 2 categories.

You then can consider finding the conditional probabilities of Y given level of X. For example, in slide 45/47 of Topic 2, if X = income level with 3 levels ordered (low, med and high), Y = order size – response with 2 categories (small, large), then you might consider finding the probability of large order size given the level of X – income:

Pr(Y = large, | X = low)  
Pr(Y = large | X = med)

Pr(Y = large | X = high)

If there is a trend where the probabilities above increase when the level of income X increases (from low to high) then it suggests an association between X and Y.

In case if X is 3 categories but nominal, we also can find the similar probabilities as above to investigate the association.

**KNN pros and cons**

https://www.datacamp.com/tutorial/k-nearest-neighbors-knn-classification-with-r-tutorial

**AS.FACTOR()**

**Q:** What is the purpose of converting a vector into a factor? E.g. in the practice paper for midterm, why do wee need the following code: data$col = as.factor(col)?

Ans: For this paper, “col” is a newly created vector (from Q4), and it has “light” and “dark” as two categories.

There is no need to factorize it since R will automatically recognizes it as a categorical variable because it has the 2 categories labelled by words (light, dark).

Only variable “color”, the original column in the dataset which has 4 categories denoted as 2,3,4,5, needs to be categorized using as.factor(). This is because its categories are 2,3,4,5 which makes R considers it as NUMERIC vector. If we wants R to recognize it as a categorical variable, then we have to factorize it by: data$color = as.factor(data$color).  
(i) If you don’t factorize it, then when you fit a linear model (like model M1 in that paper), there is only one coefficient for variable “color”.

(ii) If you factorize it, then after factorizing, variable “color” has THREE coefficients in model M1.

Q: For the mock finals, my linear model has an error when my code has

data$Status =as.factor(data$Status) ### (\*)

lm(Status ~ ., data = data)

But, my linear model works when my code doesn’t have the line in (\*).

I thought that as.factor was to change it into a categorical variable. My question is: When should I use as.factor?

Ans: For the part of using as.factor(X):

If X has categories by words (like Yes/No or Child/Adult) then there is NO need to use as.factor(X) since R will automatically know that X is categorical.

If X is categorical but its categories are numeric like 1/2/3, then you should use as.factor(X) to declare with R that X is a categorical variable.

Next, about linear model: we only can use function lm() for a response variable that is NUMERIC, not categorical variable with labels.

For the case of Status in your question, if you declare with R at the beginning: data$Status = as.factor(data$Status),

Then you cannot fit a linear model with Status as response.

Hence, if you plan to fit a linear model for Status, that means we are considering Status as a numeric variable with values are 0 and 1, then you DO NOT use as.factor() for Status.

**PREDICT() FUNCTION IN R**

Different types of models requires different input for the argument “type” in predict() function.

For Naïve Bayes, you have type = “raw” to get the probabilities of “1” and “0”; if you specify type = “class” then you get the predicted class label, either “1” or “0” only.

For DT, you can specify type = “class”, or type = “prob”.

For logistic regression, you should specify type = “response” to get the probability of success.

For KNN, when running knn(). If you specify prob = TRUE, you can get the probability of the winning class. You might check the black parts above for more details on this.

That’s how R defines for function predict() for different type of models.