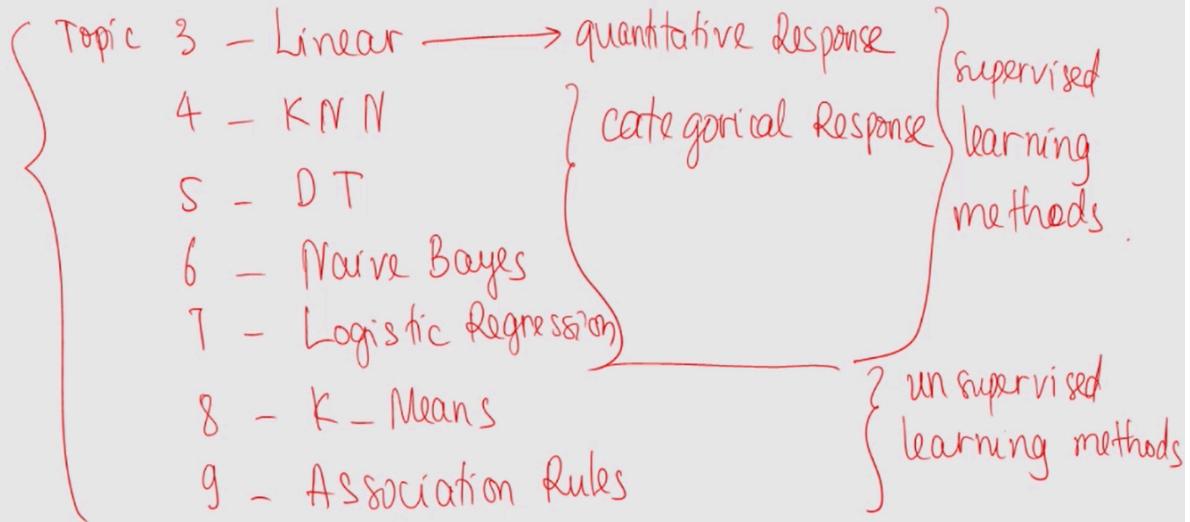


Topic 1 - R

Topic 2 - basic prob & stats →



Histogram:

#for left skewed histograms, the left tail is longer than the right tail

#ie theres more values on the left edge than the right edge, some vid said the left side looks more squished than the right side

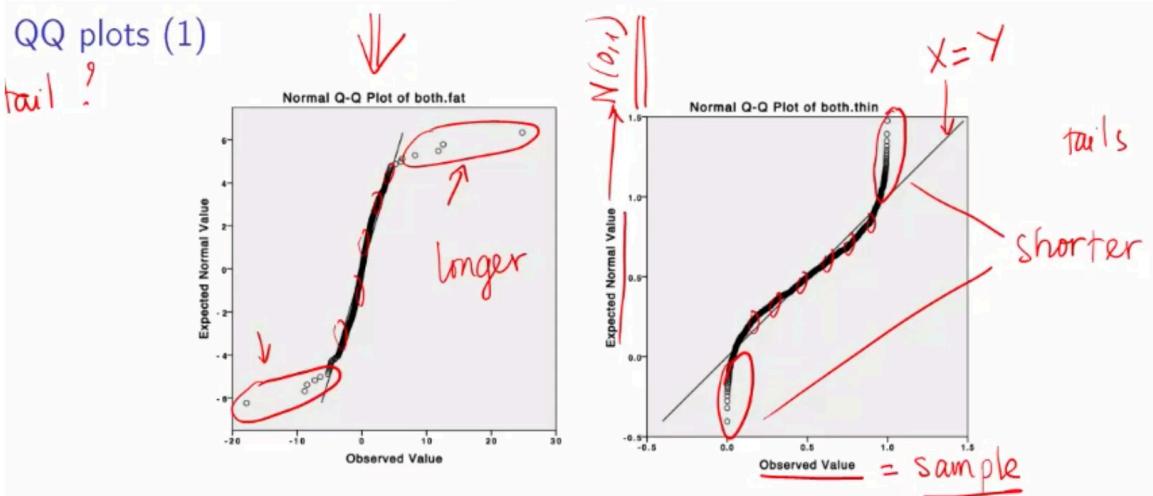
#vice versa for right skew

$$\text{outlier} = \begin{cases} > Q_3 + 1.5 \text{ IQR} \\ < Q_1 - 1.5 \text{ IQR} \end{cases}$$

$$\text{IQR} = Q_3 - Q_1$$

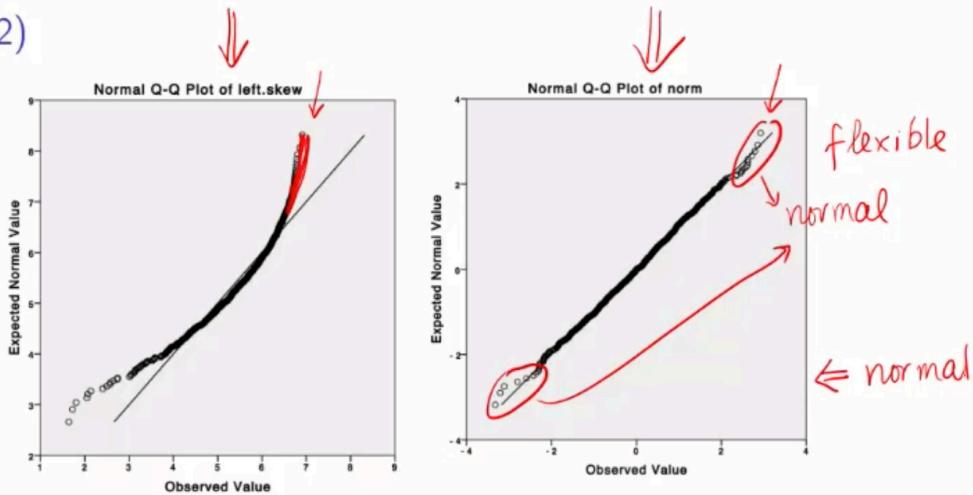
### QQ Plots:

- Stands for "Quantile-Quantile" Plot
- Used to see if the sample follows (approx) a normal distribution or not
  - So the plot matches the sample quantiles against the theoretical quantiles of a  $N(0,1)$  distribution



- Figure on the left is a data with both longer tails than normal.
- Figure on the right is for data with both tails shorter than normal

## QQ plots (2)

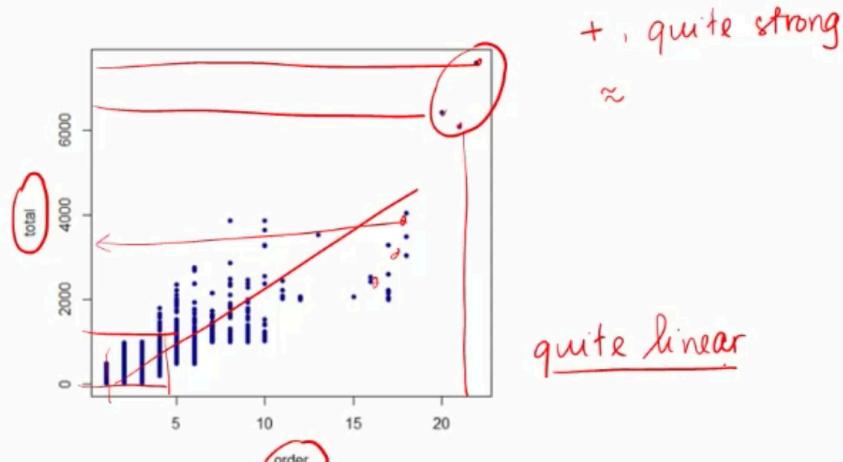


- Figure on the left is a data with left tail longer than normal but right tail is shorter than normal.
- For figure on the right, notice there are some plots not \*on\* the line, but does not show a strong trend that deviates from the straight line (ie not suddenly curving up or smth), so be flexible on it lol

## SCATTERPLOT

### Scatterplots in R

```
> plot(order, total, pch = 20, col = "darkblue")
```



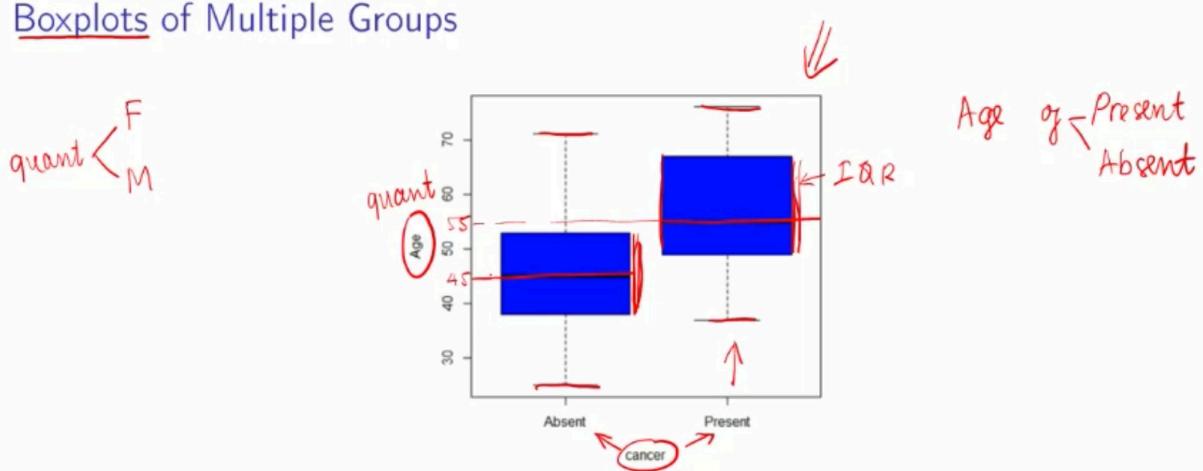
- X is the number of orders and Y is the number of sales
- Points are not random; seems like forming a trend => 2 variables are associated

- Association is positive:  
when order is small => total sales is small,  
when order is large => total sales is large
- Points do not form a straight line nicely, but approx it is quite straight => linear
- Does not form any pattern of non-linear (eg quadratic/sin curve)
- Outliers: Some obs that deviate from the rest ie the circled region, extremely large number of orders & large number of sales
- **TLDR the association is approximately linear, with a few extremely large values**

- "plot of gender by align" => y by x => gender is y and align is x

## BOXPLOT

Boxplots of Multiple Groups



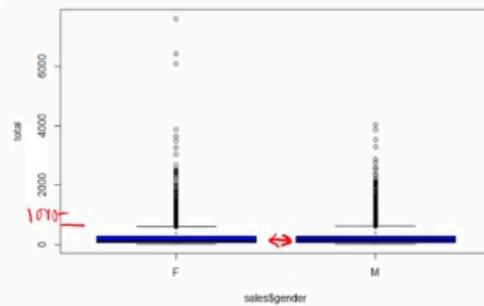
Categorical variable "cancer" has two categories: male and female. Variable "Age" is quantitative. One would check if any relationship between these two variables.

- Compare: are the medians similar, and are the boxes overlapping or not (ie is the IQR within similar ranges)
- Both groups no outliers
- If the distribution of each group is unimodal, they both could be quite normal
- Range is quite similar
- Boxes are overlapping but not much
  - IQR of the group Present is larger than the group Absent  
I.e spread in age in the center part of Present is larger
- Median age of Present > median age of Absent
  - Indicates that age could affect cancer status

## Boxplots of Multiple Groups in R

gender < F  
M

```
> attach(sales)
> boxplot(total ~ gender)
```



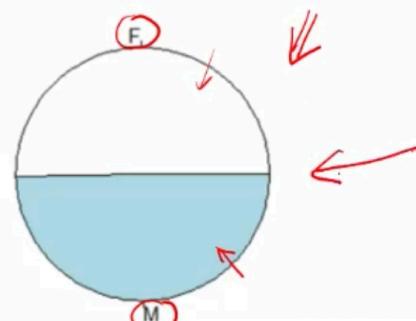
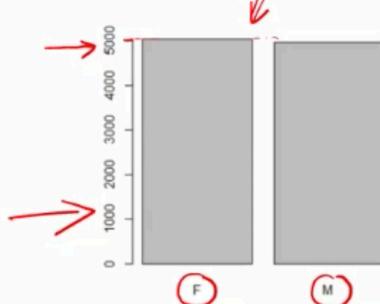
There is no obvious difference in the total sales of the customer's gender. The median of two groups are similar, and the IRQ are about the same.

- Gender is categorical, sales is quantitative
- Both have majority of datapoints below 1,000
- Both have many outliers
- Both boxes are overlapping and similar in size
- Median values are similar
- HENCE no obvious difference

## Barplot and Pie Chart

Sales → gender < F  
M

```
> count = table(gender) ←
> count # frequency table
gender
F   M
5035 4965
```



- Females are the MODAL VALUE because they're the most frequent categorical value under gender

## Two Categorical Variables

- Categorizing the number of orders into two categories: small and large size.

```
> order.size = ifelse(order<=5, "small", "large")
> table(order.size)
order.size
large small
324 9676
```

- Contingency table of frequency

*response = col*

	order.size	
gender	large	small
F	142	4893
M	182	4783

Raw

- Response variable is the dependent variable lol, cus it changes or 'responds' to changes in the independent variable
- Response variable is usually put as the 2nd parameter in table() so it becomes the column. Its like table(row, column)

## CONTINGENCY TABLE

- Contingency table of joint proportion

	order.size	
gender	large	small
F	0.0142	0.4893
M	0.0182	0.4783

*total = 1*

- Total of all the rows and columns is 1
- prop.table( frequency table created using table() ) so 1 parameter only

## Contingency Tables

large for  $\rightarrow M$  ) larger  $\Rightarrow$   
 $F$

- Contingency table of proportion by gender  $\times 100$

```
> tab = prop.table(table, "gender") # proportion by gender  
> tab
```

gender	large	small	total
F	0.02820258	0.97179742	1
M	0.03665660	0.96334340	1

Among orders by females, 2.82% are large orders while 3.67% of orders by males are large.

There is difference, but the difference is not large.

```
> tab
      order.size
gender    large      small
  F 0.02820258 0.97179742
  M 0.03665660 0.96334340
>

> 0.028/0.972
[1] 0.02880658
> 0.037/0.963
[1] 0.0384216
> tab[1]
[1] 0.02820258
> tab[2]
[1] 0.0366566
> tab[3]
[1] 0.9717974
> tab[4]
[1] 0.9633434
> |
```

## Odds Ratio

- Odds ratio is the ratio of two odds of success: odds of larger orders in the female group (0.029), and odds of larger orders in the male group (0.038).

$$OR = \frac{0.029}{0.038} = 0.76 \rightarrow OR = \frac{1}{0.76} \approx 1.3$$

odds of large in F

What does this value mean?

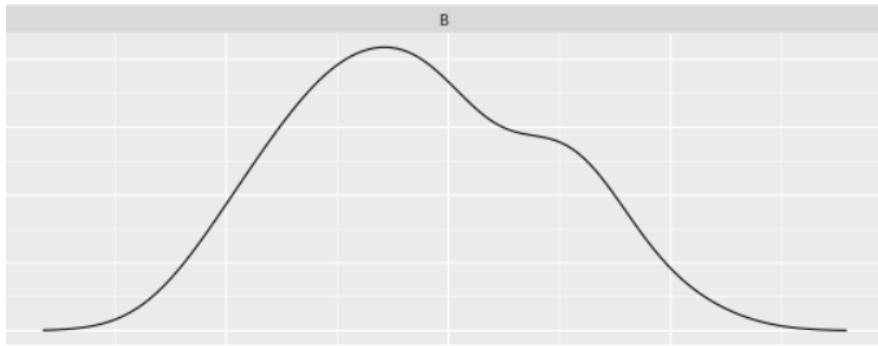
- 0.76: the odds of having large order amongst female is 0.76 times the odds of having large orders among males
- If you flip the numerator and denominator, the odds of having large order amongst males is 1.3 times the odds of having large orders amongst females (reciprocal of 0.76)  $\Rightarrow$  meaning is still the same
- Note that odds of success =  $p / (1-p)$

- For example, in simple linear regression with only one predictor, we assume a model of the form

$$y \approx f(x) = \beta_0 + \beta_1 x.$$

$f(x) = \beta_0 + \beta_1 \cdot x^2$

linear



This is considered unimodal symmetric

*Add on pls. OG is from NM2204*

distinct(): find the unique senders		
Print all lines/specified number of lines	%>% print( <b>n = length(playlist_data\$DJ_Name)</b> )	Filter (%>%) is required
Conjugate strings within a print statement  paste0()	<pre>paste0("hello", 4) Prints hello4</pre> <p>Note that usage of "," does not automatically give a space</p> <pre>paste0("the equation is: price(hat) = ", M2\$coeff[1], " + ", M2\$coeff[2], "price + ",M2\$coeff[3], "I(NW = 0)")</pre>	
Append vector	<pre>col = c(og_vector, new_values) Eg col = c(col, "light")</pre> <p>OR</p> <pre>append(og_vector, new_values)</pre> <p>OR for an INDEXED LIST (typically with for loops)</p> <pre>skibbidi = c()  for (i in 1:15) {   kout = kmeans(standardised.X, centers = i)   <b>skibbidi[i] = kout\$withinss</b> }</pre>	
ggplot garbage	<pre>ggplot(data = playlist_data, mapping = aes(x=Age,y=Rating)) +   geom_point(shape=23, fill = "purple", size=3) +   labs(x = "Age", y = "Rating", title = "Scatter graph of Age against Ratings", caption = "My insight is that since there are both high and low ratings for both younger and older DJs, ratings are not dependent on age")</pre>	Refer to documentation: <a href="https://ggplot2.tidyverse.org/reference/#plot-basics">https://ggplot2.tidyverse.org/reference/#plot-basics</a> Refer to Challenge 2
Initialize list	<pre>student_info &lt;- <b>list</b>(student_names=c("Alice","Bob","Catherine"), scores=c(85L, 92L, 78L), passed=c(TRUE,</pre>	Refer to Challenge 3, Q10

	TRUE, FALSE))	
Explicit coercion (conversion)	prices <- <b>as.numeric</b> (prices)	Refer to Challenge 3, Q12
Implicit coercion (conversion)	<pre>nums &lt;- c(5, 10, 15) noms &lt;- c("apple", "banana", "cherry") <b>numnoms</b> &lt;- c(nums, noms)</pre> <p>Combining the numeric and character vectors will implicitly coerce the numeric vector to character. The resulting vector will be character type. R coerces data to a common type when vectors are combined. <b>R coerces data to a common type when vectors are combined</b></p>	Refer to Challenge 3, Q13
What does unlist() do?	It converts a list to a vector.	
Filter: tl;dr	<pre>comm_data %&gt;% <b>filter</b>(date == "2/8/2023" &amp; channel == "Twitter") %&gt;% select(date, channel, message)  common_cyl &lt;- cars %&gt;% filter(ncyl %in% c(4, 6, 8))</pre>	The AND(&), OR( ) all apply here <a href="https://www.tutorialspoint.com/r/r_operators.htm">https://www.tutorialspoint.com/r/r_operators.htm</a> Datacamp
arrange()	Will arrange rows in ASCENDING ORDER. To reverse order, put desc()	
create a new data frame with summarised data  summarise()  Summarising based on other variable: eg summarise ave sentiment FOR EACH sender (summarise variable A categorised by variable B)	<pre>%&gt;% summarise(total_char = sum(nchar(message)))  %&gt;% summarise(avg_sentiment=mean(sentiment))  %&gt;% group_by(sender) %&gt;% summarise(avg_score=mean(sentiment))</pre>	<a href="https://dplyr.tidyverse.org/reference/summarise.html">https://dplyr.tidyverse.org/reference/summarise.html</a> Challenge 4, Q8 & 12 Challenge 4, Q7
Create dataframe / data frame	df <- data.frame(first_column, second_column)	
Access ONE specific column only OR Access/limit to certain columns	data_frame\$column_name  For MULTIPLE columns: use %>% select()	

Access vector within list, given vector name	<pre>mixed_data &lt;- list( numeric_vector = c(10, 20, 30), character_vector = c("red", "green", "blue"), logical_vector = c(TRUE, FALSE, TRUE) )  mean_numeric &lt;- mean(mixed_data\$numeric_vector)</pre>	Challenge 3, Q16
Access a vector inside of a list, given the name of the vector  Eg uk bob = "tire", you want to find "tire" and uk bob	<pre>score_bob &lt;- student_info\$scores[student_info\$names == "Bob"]</pre>	Challenge 3, Q10
Find and <b>print the indices of all occurrences of the word</b> "apple."	<pre>words &lt;- c("apple", "banana", "cherry", "apple") indices &lt;- which(words == "apple") print(indices)</pre>	Challenge 3, Q19
Create frequency table	<b>table(row, column)</b>  <b>For all categorical variables data</b>	Topic 2 Video 8
Create contingency table  Create a table where all the rows OR columns add to 1	<pre>prop.table(name_of_freq_table) prop.table(name_of_freq_table, "name_of_row/column")  Eg tab = prop.table(table, "gender")#proportion by gender  Remember if using a single variable, don't use the variable directly ie must do: prop.table(table(Churned))  Instead of prop.table(Churned)  <b>For all categorical variables data</b>  NOTE: prop.table(name_of_freq_table)with NO ADDITIONAL PARAMETERS will make the WHOLE TABLE TO ADD TO 1. i.e. [,1] [,2] [,3] [1,]    1    3    5 [2,]    2    4    6  will become</pre>	Topic 2 Video 8

	<pre>[,1]      [,2]      [,3] [1,] 0.04761905 0.1428571 0.2380952 [2,] 0.09523810 0.1904762 0.2857143  Where total = 1+2+3+4+5+6 = 21 [1,1] = 1/21 = 0.04761905 [2,1] = 2/21 = 0.09523810  To get proportions by row, try q = table(Churned) q = q/rowSums(q)  Same goes for colSums() if columns need to be added.  [OR do q = prop.table(q, "gender")#proportion by gender but i dont trust you to do this]</pre>	
Create bar plot  No library needed	<pre>barplot()  barplot(max.temp, main = "Maximum Temperatures in a Week", xlab = "Degree Celsius", ylab = "Day", names.arg = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"), col = "darkred", horiz = TRUE)</pre>	
Create a Boxplot  Find outlier values of boxplot (without ggplot)	<pre>boxplot(total, xlab = "variable", col = "blue")  Outlier = boxplot(column_variable)\$out  Eg outtie = boxplot(fev_var, xlab = "FEV", col = "blue")\$out ; outtie</pre>	
Comment on boxplot outliers		
Find the position of the outliers, ie the index of outliers	<pre>index = which(FEV %in% c(out))  Note FEV is the name of the variable/column, not the entire db  FEV %in% c(out) gives you a logical list of TRUEs and FALSEs ⇒ which() will isolate the TRUEs and return the indexes of the outliers</pre>	

Get number of rows/columns  dim()	Returns no. rows (obs) and columns (variables) Stands for dimensions  dim(db)	
Directly copy dataset into Rstudio, by implicitly defining the variables based on column names  attach() Note spelling!	<b>attach(db_name)</b>  Now the name of the dataset is fev Can reference the variables/column directly eg.  <b>Forced_Expiratory_Volume &lt;- read.csv("~/GitHub/DSAI1101 Slayers/datasets/FEV.csv")</b>  <b>attach(Forced_Expiratory_Volume)</b>  <b>hist(FEV, col = 10, freq = FALSE)</b> #where FEV is the name of the variable from the Forced_Expiratory_Volume db	Tutorial 2
Is the sample of (variable) normally distributed?	USE QQPLOT. IF QN DOESNT ASK, INITIATE IT YOURSELF  <b>qqnorm(fev_var, pch = 10) qqline(fev_var, col = "red")</b>	Tutorial 2
predict()	<b>predict(model_fit, newdata = , type = "prob")</b>  type = "class" will just give u the most likely category.  Note that if categorical value, keep the factor = c("3") or factor = c("1") yada yada  Note if new data has multiple predictors/catagories, put the, in a data frame eg new_data = data.frame(width = 5, length = "yes")  <b>NEW DATA IS LIKE ALL THE PREDICTORS, DO NOT PUT UR RESPONSE VARIABLE AS UR NEW DATA</b>  <b>Someone tell me why the FUCK new.data and newdata are 2 DIFFERENT BUT COMPLETELY VALID INPUT VARIABLES</b>  <b>USE newdata</b>	

lm(price ~ area)\$coeff	<b>Linear regression</b> M1 = lm() summary(M1)													
geom_bar usage  How to get bar chart to go of proportions (like percentage, in a piechart, but in bar from)	<pre>email %&gt;%   mutate(has_image = image&gt;0) %&gt;%   ggplot(aes(x = has_image, fill = spam)) +   geom_bar(position = "fill")</pre> <p>Count of emails by has_image and spam status.</p> <table border="1"> <thead> <tr> <th>has_image</th> <th>spam</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>FALSE</td> <td>not-spam</td> <td>~0.95</td> </tr> <tr> <td>FALSE</td> <td>spam</td> <td>~0.05</td> </tr> <tr> <td>TRUE</td> <td>spam</td> <td>1.00</td> </tr> </tbody> </table>	has_image	spam	Count	FALSE	not-spam	~0.95	FALSE	spam	~0.05	TRUE	spam	1.00	Datacamp
has_image	spam	Count												
FALSE	not-spam	~0.95												
FALSE	spam	~0.05												
TRUE	spam	1.00												
correlation  cor(heighty, fev_var)	cor(1st_var, 2st_var)  #does NOT matter which variable goes first #idt it works for categorical variables													
Create a decision tree  Libraries: - rpart - rpart.plot  Relevant functions: - rpart() - rpart.control()	<pre>fit &lt;- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + poutcome,               method = "class",               data = bankdata,               control = rpart.control(minsplit = 1),               parms = list(split = "information"))  method = "class", #tells R to return the category of response (yes or no) [DSA1101 will only use method = "class", no "anova" or "poision"]  METHOD IS A STRING VARIABLE HOR ie method = "class"</pre>	Topic 5 Pre-lect vid 3												

	<pre> control = rpart.control(minsplit = 1), #tells R how to split the node; minsplit = 1 means a branch is created when there is at least 1 observation in that branch =&gt; <b>tells R how big we want the tree to be</b> (how many branches we want)  Can also split by complexity parameter: control = rpart.control(cp = 0.001)  parms = list(split = "information") # what criteria to use to choose the root node &amp; internal nodes; either info gained or gini index; default val is gini  <b><u>TO PRINT THE TREE:</u></b>  rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE) </pre>	
standardise all the predictors (the x es)	Lag1 = scale(Lag1)	
Get the inverse of a matrix	solve(matrix_name)	
Transpose the matrix	t(matrix_name)	
Multiply matrix by matrix	matrix_name %*% matrix_name	
Create a matrix	<pre> cbind(1st_desired_col, 2nd_desired_col) [bind the columns together]  Eg cbind(1, SAT) #SAT is variable name returns       SAT [1,] 1 1315 [2,] 1 1310 [3,] 1 1336 [4,] 1 1300 [5,] 1 1250 [6,] 1 1320 [7,] 1 1290  Or can do cbind(c(1,2), c(3,4)) [,1] [,2] [1,]    1     3 [2,]    2     4 </pre>	

	<pre>ALTERNATIVELY there is rbind() rbind(1st_desired_row, 2nd_desired_row) [bind the rows together]</pre>																																	
Create scatterplot (Without ggplot)	<pre>plot(x, y, pch = 20)  #notice its x then y please #pch is just the shape of the points, can ignore  For this type (type = "b"), plot(1:K, wss, col = "blue", type="b", xlab="Number of Clusters", ylab="Within Sum of Squares")</pre> <table border="1"> <caption>Data points estimated from the scatterplot</caption> <thead> <tr> <th>Number of Clusters</th> <th>Within Sum of Squares</th> </tr> </thead> <tbody> <tr><td>1</td><td>10000</td></tr> <tr><td>2</td><td>5000</td></tr> <tr><td>3</td><td>3000</td></tr> <tr><td>4</td><td>2500</td></tr> <tr><td>5</td><td>2200</td></tr> <tr><td>6</td><td>2000</td></tr> <tr><td>7</td><td>1800</td></tr> <tr><td>8</td><td>1600</td></tr> <tr><td>9</td><td>1500</td></tr> <tr><td>10</td><td>1400</td></tr> <tr><td>11</td><td>1300</td></tr> <tr><td>12</td><td>1200</td></tr> <tr><td>13</td><td>1100</td></tr> <tr><td>14</td><td>1000</td></tr> <tr><td>15</td><td>1000</td></tr> </tbody> </table>	Number of Clusters	Within Sum of Squares	1	10000	2	5000	3	3000	4	2500	5	2200	6	2000	7	1800	8	1600	9	1500	10	1400	11	1300	12	1200	13	1100	14	1000	15	1000	
Number of Clusters	Within Sum of Squares																																	
1	10000																																	
2	5000																																	
3	3000																																	
4	2500																																	
5	2200																																	
6	2000																																	
7	1800																																	
8	1600																																	
9	1500																																	
10	1400																																	
11	1300																																	
12	1200																																	
13	1100																																	
14	1000																																	
15	1000																																	
Get R square value / R^2 / R2 Get adjusted R square	<pre>M1 = lm(price ~ size) summary(M1) # take from here  OR summary(hdb.model)\$r.squared  adjusted R square: summary(M1)\$adj.r.squared</pre>																																	
Form equation of Linear Model M2\$coeff	<pre>M2\$coeff</pre> <table style="margin-left: 40px;"> <tr> <td>#</td> <td>(Intercept)</td> <td>size</td> <td>NW1</td> </tr> </table>	#	(Intercept)	size	NW1																													
#	(Intercept)	size	NW1																															

	<pre> ## -15257.51385      77.98513   30569.08729  M2\$coeff[1]: β0 M2\$coeff[2]: β1 M2\$coeff[2]: β2 </pre>	
Cheatcode for equation of general linear model	<pre> paste0("log[phat/(1-phat)] = ", M2\$coeff[1]," + ", M2\$coeff[2], "Life_expectancy + ", M2\$coeff[3], "Adult_mortality + ", M2\$coeff[4], "infant_deaths + ", M2\$coeff[5], "Alcohol")  paste0("log[phat/(1-phat)] = ", M3\$coeff[1]," + ", M3\$coeff[2], "*I(Age = Child) + ", M3\$coeff[3], "*I(Sex = Male) + ", M3\$coeff[4], "*I(Class = 2nd) + ", M3\$coeff[5], "*I(Class = 3rd) + ", M3\$coeff[6], "*I(Class = Crew)")  log[phat/(1-phat)] = 2.04383742228878 + 1.06154237626018*I(Age = Child) + -2.42006034 580599*I(Sex = Male) + -1.01809495158843*I(Class = 2nd) + -1.77776221774166*I(Class = 3rd) + -0.857676155374125*I(Class = Crew) </pre>	
knn()	<pre> Requires library(class)  knn.pred = knn(train.x, test.x, train.y, k = 1) </pre>	Documented in Topic 4 my own code

Create histogram (without ggplot)	<pre>hist(Temperature, main = "Maximum daily temperature at La Guardia Airport", xlab = "Temperature in degrees Fahrenheit", xlim = c(50,100), col = "darkmagenta", freq = FALSE)</pre>	
Overlay density plot line over histogram	<pre>lines(density(data\$x), col = "red")</pre>	
Sample data randomly  <code>sample(values, size_of_subsample)</code>	<pre>sample(my_vec, size = 3)  sample(1:10, size = 3) Will randomly give 3 numbers from 1 to 10  folds_j &lt;- sample(rep(1:n_folds, length.out = n))  Eg &gt; sample(rep(1:4, length.out = 15)) [1] 1 3 4 4 3 2 2 2 3 1 3 1 2 1 4 &gt; sample(rep(1:4, length.out = 15)) [1] 3 4 1 4 3 1 2 1 3 1 2 3 4 2 2</pre>	<b>Rep basically duplicates the vector however many times</b>  <pre>#define vector x &lt;- c(1, 10, 50)  #replicate the vector three times rep(x, times=3)  [1] 1 10 50 1 10 50 1 10 50</pre>
standardised features  <code>scale(all_relavent_predictors)</code>	<pre>standardized.X = scale(caravan[,-86]) # scaling all the data set, except the last column</pre>	

<p>Forming Naive Bayes Classifier</p> <p>Requires library(e1071)</p>	<pre>model &lt;- naiveBayes(Enrolls ~ Age+Income+JobSatisfaction+Desire, traindata) #, laplace=0)  model &lt;- naiveBayes(response_var ~ predictor1 + predictor2, traindata)  Then use: results &lt;- predict(model,testdata,"raw"); results  results[2]/results[1]</pre>	
<p>Create ROC / Find AUC</p> <p>Requires library(ROCR)</p>	<p>Drop unnecessary columns first.</p> <p>Idea: use testing data, see how well the model does. So split the test set into predictors and response.</p> <pre>#prelude (still using predict) nb_prediction = predict(nb_model,   newdata = banktest[,-ncol(banktest)],   type = 'raw') NOTE its banktest[,-ncol(banktest)] WITH THE NEGATIVE SIGN SO ITS EVERYTHING <b>BUT</b> THE LAST COLUMN =&gt; <b>REMOTES THE RESPONSE VARIABLE</b>  type = 'raw' will give raw probabilities If type = "response" doesnt work than idk just change to "class" or "raw" instead  library(ROCR)  score &lt;- nb_prediction[,c("yes")]</pre>	<p>Tutorial 8 and 9</p>

	<pre> #this will return all the probabilities that each outcome will be positive (or 'yes').  #so if the prediction of 'yes' is 0.9, it's very likely to be yes,, if the prediction of 'yes' is 0.001 then its very likely to be no  actual_class &lt;-   banktest\$subscribed == 'yes' #this converts all to TRUE and FALSE which will be treated as 1 or 1 (due to how the method works)  pred &lt;- prediction(score , actual_class) #this formats the input so 'score' and 'actual_class' match with each other  perf &lt;- performance(pred , "tpr", "fpr")  #now we can plot the figure  plot(perf, lwd = 2) #lwd specifies how thick the curve is  abline(a = 0,       b = 1,       col = "blue",       lty = -3) #create diagonal curve that slices square into half by its diagonal </pre>	
Predict for ROC / AUC Ie using ROCR to predict	#TO FORMAT THE INPUT	

This is the main bits, repeated from previous row.

`prediction()` followed by `performance()`, then `plot()`

```
pred <- prediction(score , actual_class)
perf <- performance(pred , "tpr", "fpr")
plot(perf, lwd =2)
abline(a=0, b=1, col ="blue", lty =3)
```

Determine AUC value	<p>AUC value determines what is the best threshold for a negative to become a positive (usually delta)</p> <pre> threshold &lt;- round(as.numeric(unlist(perf@alpha.values))) )  fpr &lt;- round(as.numeric(unlist(<u>perf@x.values</u>)), 4)  tpr &lt;- round(as.numeric(unlist(<u>perf@y.values</u>)), 4)  #adjust margins and plot TPR and FPR par(mar = c(5,5,2,5))  #the plotting stuff: just copy paste LMAO plot(alpha ,tpr , xlab ="Threshold", xlim =c(0 ,1) ,       ylab = "True positive rate ", type ="l", col = "blue") par( new ="True") plot(alpha ,fpr , xlab ="", ylab ="", axes =F, xlim =c(0 ,1) , type ="l", col = "red" ) axis( side =4) # to create an axis at the 4th side mtext(side =4, line =3, "False positive rate") text(0.18 ,0.18 , "FPR") text(0.58 ,0.58 , "TPR") </pre>	<a href="https://sets.netlify.app/module/65d9a044a1d9248bb0c34963/topic/65d9a076a1d9248bb0c34b55/6619fd77da66393bf2be69e8">https://sets.netlify.app/module/65d9a044a1d9248bb0c34963/topic/65d9a076a1d9248bb0c34b55/6619fd77da66393bf2be69e8</a> not relavent
---------------------	---	--

AUC Simplified	<pre>aucLM = performance(pred, measure = "auc") aucLM@y.values[[1]] #AUC value = 0.8895238</pre>	
Final code for ROC / AUC for <b>KNN</b>	<pre>pred.knn = knn(train.X, test.X, cl = train.Y, k = 30) # "prob = TRUE" is NOT added --&gt; to get the class labels  pred.knn.prob= knn(train.X, test.X, cl = train.Y, k = 30, prob = TRUE) # --&gt; to get the probabilities of winning labels  winning.prob = attr(pred.knn.prob, "prob") # to extract the winning probabilities  n = length(test.Y)  prob = numeric(length = 0) # 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]) }  helppain = prediction(prob, test.Y)  rocObjKNN = performance(helppain, measure = "tpr", x.measure = "fpr")  plot(rocObjKNN, lwd =2)  abline(a=0, b=1, col ="blue", lty =3)</pre>	Refer to statistical assignment (and below this document)

Attain accuracy (trust me bro)	<pre>M3 &lt;- naiveBayes(Status ~ Life_expectancy + Adult_mortality + infant_deaths + Alcohol, data)  results &lt;- predict(M3, data[,2:5], "class"); results  confusion.matrix = table(Status, results) confusion.matrix  diag(confusion.matrix)  <b>accuracy</b> = <b>sum(diag(confusion.matrix))/sum(confusion.ma</b> <b>trix)</b> <b>accuracy</b></pre>	Refer to Statistical Report
Logistic regression Generalised Linear Regression (GLM)	<pre>M1 &lt;- glm(Churned ~ . , # . is all columns except the response col       data = churn,       family = binomial(link = "logit")) #logit is default tho  Note: Churned must be equal to 0 and 1, it can be as.factor(), but the category names MUST be 0 or 1. Ironically it doesn't matter if you converted to factor, for glm.  REMEMBER ur LHS will be log(p/(1-p)), the log odds of having disease or wtvr. So the next step is finding p itself.</pre>	

Interpret coefficient of logistic regression / generalised linear model	<p>The coefficient of Alcohol is 0.148746770871717.</p> <p># When the alcohol consumption (per capita (15+)) consumption, in liters of pure alcohol) increases by 1 unit, then the log odds of Status changing from developing to developed increases by 0.149, keeping all other variables the same.</p> <p># When alcohol consumption increases by 1 unit, then the odds of being a developed country changes by <math>e^{0.148}</math> times, keeping all other variables the same</p> <p>When the number of contacts increases by 1 unit, then the log odds of churning increases by [coefficient of contacts], keeping all other variables the same. =&gt; When the number of contacts increases by 1 unit, then the odds of success changes by <math>e^{0.38}</math> times (or <math>e^{(\text{coefficient of contacts})}</math> times), keeping all other variables the same</p> <p>How to interpret married: Comparing a married person (<code>Married = 1</code>) vs a non-married person, when other variables are the same, then the log odds of churning will be larger by 0.07. =&gt; Comparing a married person (<code>Married = 1</code>) vs a non-married person, then the odds of churning will change by <math>e^{0.07}</math> times.</p>	Refer to past year paper working OR Topic-7-MY OWN code Logistic Regression
---	---	---

<pre>k-means clusters No external library needed</pre>	<pre>kout = kmeans(hdb[,c("floor_area_sqm", "amenities")],               centers = 2) #centers = nummber of clusters #note kmeans data only accepts dataframes  <b>Plot the clusters</b> plot(floor_area_sqm,       hdb\$amenities,       col = kout\$cluster) #col is colours, so colours differentiate the 2 clusters  <b>Centroids of the clusters</b> kout\$centers #a matrix of cluster centers  #   floor_area_sqm amenities # 1      69.46017 7.448739 # 2      65.40782 37.279330 # so these are the coordinates of the 2 centroids (of the 2 clusters)  <b>Size of each cluster</b> kout\$size #the number of pts in each cluster # [1] 595 179  <b>Within Sum of Squares</b> kout\$withinss #sum of squares of all the different clusters added together  kout\$tot.withinss</pre>	<p>STANDARDISE THE INPUTS before starting K-means</p> <ul style="list-style-type: none"> <li>- Use scale()</li> </ul> <p>If doing manually, ie given all the indiv plot points, USE plot() TO DETERMINE WHERE ALL THE PLOTS ARE</p> <p>REMEMBER K-MEANS IS VERY SCATTERPLOT BASED</p>
--	--	---

Association Rules	<pre>library(arules) library(arulesViz)  Groceries\$itemInfo[1:10,] #all columns Groceries@data[,100:110] #all rows #note the 2nd one will give u the sparse matrix (unreadable dot and slash)</pre>	
Return the actual itemsets (instead of the general data / summary)	<pre>inspect() inspect(sort(itemset.1, by = "support"))</pre>	
Attain itemsets that fit a certain support level	<pre>itemset.1 = apriori(Groceries, parameter = list(minlen = 1, maxlen = 1, support = 0.02, target = "frequent itemsets")) #so here itemsets that hit the minimum support of 0.02 (2%) are considered frequent itemsets #bc minlen = 1 and maxlen = 1, all returned itemsets only have 1 item (each itemset has a length of 1)  summary(itemset.1)</pre>	As support increases, number of returned itemsets decreases
Attain and plot all the rules	<pre>rules = apriori(Groceries, parameter = list(support = 0.001, confidence = 0.6, target = "rules"))  plot(rules)  #limit to only 100 dots / rules with the highest lift plot(rules, measure = c("support", "confidence"), shading = "lift", col = "black", limit = 100)  <b>Attain the top few rules</b> inspect(head(sort(rules, by = "lift"), 3))</pre>	

	<pre>#note that 3 is a parameter for head() ie head() controls how many outputs come out  <b>Plot the 5 rules with the highest lift</b> highLiftRules = head(sort(rules, by = "lift"), 5) plot(highLiftRules, method = "graph"  (refer below for a plot that is more easily readable)</pre>	
<i>Interpreting the weirdass association rules plot</i>	<pre>highLiftRules = head(sort(rules, by = "lift"), 5) plot(highLiftRules, method = "graph", engine = "igraph", edgeCol = "blue", alpha = 1)</pre> <p><b>Graph for 5 rules</b></p> <p>size: support (0.001 - 0.002) color: lift (11.279 - 18.996)</p> <p>As size increases, support increases As colour darkens, lift increases</p>	

	<p>Those who buy instant food products AND soda, also tend to buy hamburger meat</p>	
Inspect / List the first 6 transactions	<pre>inspect(head(data)) #gives first 6 itemsets inspect(head(data, 10))</pre>	requires arules library
Add text to a plot Manually add a point to a scatter plot	<pre>text(xcord, ycord, "text to insert") points(xcord, ycord, pch = 20, col = "green")</pre>	points() WITH AN S ITS PLURAL!!!
Within Sum of Squares (WSS)	<pre>K = 15 wss &lt;- numeric(K) #this creates an empty vector of 15 0s  for (k in 1:K) {   wss[k] &lt;-   sum(kmeans(scale(data[,c("floor_area_sqm",   "resale_price")]), centers=k)\$withinss) }  plot(1:K, wss, col = "blue", type="b", xlab="Number of Clusters", ylab="Within Sum of Squares")  My own attempt is this:</pre>	By default just plot everything first to get a sense of the data  Remember to scale the data before doing kmeans() !

	<pre> kdata = scale(sendhelp[,    c("floor_area_sqm","resale_price")])  wss = numeric(15)    for (i in 1:15) {    wss[i] = sum(kmeans(kdata, centers =    i)\$withinss) }    plot(1:15, wss, type = "b") </pre>	
Convert sparse matrix into readable itemset	<pre> apply(Groceries@data[,100:105], 2,         function(r)  paste(Groceries@itemInfo[r, "labels"],  collapse = ", ")) </pre>	

**NOTES FROM MIDTERM:**

- 1) Remember to `as.factor()` EVERYTHING. Not sure? CHECK IF ANY CATEGORICAL VAR NEEDS TO BE CONVERTED FROM INT TO CAT. `AS.FACTOR()` EVERYTHING
  - a) If attaching dataframe,
    - i) `breastdata$survival.status = as.factor(breastdata$survival.status)`  
use the dollar signs first!!!
    - ii) THEN attach, after converting to factor
    - iii) CHECK AGAIN using `glimpse` or `head`
    - iv) `breastdata$survival.status = as.factor(breastdata$survival.status)`  
`attach(breastdata)`  
`glimpse(breastdata)`
- 2) Removing specific columns: DO NOT
 

```
breastdata = breastdata[,1] + breastdata[,3] + breastdata[,4]
```

  - a) INSTEAD: `breastdata = breastdata[,c(1,3,4)]`
  - b) OR to remove ID column / first column:  
`breastdata = breastdata[,-1]`
- 3) Type 1 error: False Positive Rate (FPR)  
Type 2 Error: False Negative Rate (FNR)
- 4) 3 sf please!!

**INCLUDING PICTURES IN R MARKDOWN:**

```

```{r,eval=TRUE, echo=FALSE}
knitr::include_graphics("~/GitHub/DSA1101 Slayers/Pictures/Confusion Matrix Pic.jpg")
knitr::include_graphics("~/GitHub/DSA1101 Slayers/Pictures/CM spam emails.jpg")
```

```

Hi Bryan, this is Dawn from DSA1101 T3, I wanted to ask about a mistake from assignment 2 (statistical report). I don't understand this comment: "Wrong KNN ROC code (should use 1-attr)"

Hi,

Sorry to reply your question late. I'm stuffed with full schedule these days. :(((

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  $\Pr(Y = 1) > \text{delta}$  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 calculated 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 0 and 1 to plot the ROC curve. Then, at the step 1 above, R will treat 0 and 1 as the probabilities  $\Pr(Y = 1)$  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.

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 = 30) # "prob = TRUE" is NOT added -->
to get the class labels
pred.knn.prob= knn(train.X, test.X, cl = train.Y, k = 30, prob = TRUE) # --> to get the
probabilities of winning labels
winning.prob = attr(pred.knn.prob, "prob") # to extract the winning probabilities
```

```
n = length(test.Y)
prob = numeric(length = 0) # 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.

```
helppain = prediction(prob, test.Y)
rocObjKNN = performance(helppain, measure = "tpr", x.measure = "fpr")
plot(rocObjKNN, lwd =2)
abline(a=0, b=1, col ="blue", lty =3)
```

How do I plot ROC for Naive Bayes and Logistic regression with class label, instead of probability?

With class label, like "0" and "1", you will need to change them to be numeric, as 0 and 1. For example:

```
x = as.factor(c(0, 1, 1, 0)) # x is a vector of labels "0" and "1"
```

```
y = as.numeric(paste(x)) # this will change vector x into y = a numeric vector
```

different types of models requires different ways to input for “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 my 2 answers for Q83 above.

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

We should choose a lower threshold (than 0.5) when the success criteria is very rare ie a very low percentage of the population will be positive

#how tpr and fpr changes when threshold changes

```
```{r getting the precise best alpha value, eval = TRUE, echo=TRUE}
cbind(alpha, tpr, fpr)
````
```

for churn data, delta is around 0.24

this lines up with the prop.table(table(Churned)) value where 0.217875 ppl churned (proportion of 'yes' in the data), so delta should be AROUND there (need not be exact)

The exact delta used is up to you, can depend on the users needs (if they want to prioritise TPR over FPR, etc)

Each value of delta will correspond to a value of TPR and FPR.

With 0.5, you can find TPR and FPR to see if both of them are good. If both of them are good then suitable; if any of them is not good then not suitable. :)