Statistical Computing with R: Masters in Data Sciences 503 (S24) Third Batch, SMS, TU, 2024

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Data Analysis and Decision Modeling, MBA, Pokhara University, Nepal

Faculty, FAIMER Fellowship in Health Professions Education, India/USA.

Supervised Learning with classification models

- Logistic Regression
  - Model Accuracy
  - Model Prediction
  - Model Validation
- KNN and ANN (done!)
  - Model Accuracy
  - Model Prediction
  - Model Validation
- Naïve Bayes
  - Model Accuracy
  - Model Prediction
  - Model Validation
- Support Vector Machine
  - Model Accuracy
  - Model Prediction
  - Model Validation

Supervised learning classification models:

- When the dependent variable is categorical or factor (instead of continuous) then we must use the classification model
- The most common one is the logistic regression, which is of three types
  - Binary logistic regression
  - Multinomial logistic regression
  - Ordinal logistic regression

- Binary logistic regression
  - Used when dependent variable has only two categories: 0 and 1
- Multinomial logistic regression
  - Used when dependent variable has more than two nominal categories (**e.g. ChatGPT**)
- Ordinal logistic regression
  - Used when dependent variable has more that two ordinal categories

Binary Logistic Regression Model

- Y (dependent variable) denotes, say, the survival status i.e. **1 = Yes and 0 = No.**
- Then, P (Y = 1) = p represents the probability of being survived in the titanic incidence.
- And, P (Y = 0) = 1 – p represents the prob. of not being survived in the titanic incidence.
- The equation for this type of problem is:

$$[p/(1-p)] = e^{(\alpha + \beta x + u)}$$

Linearized Binary Logistic Regression Equation

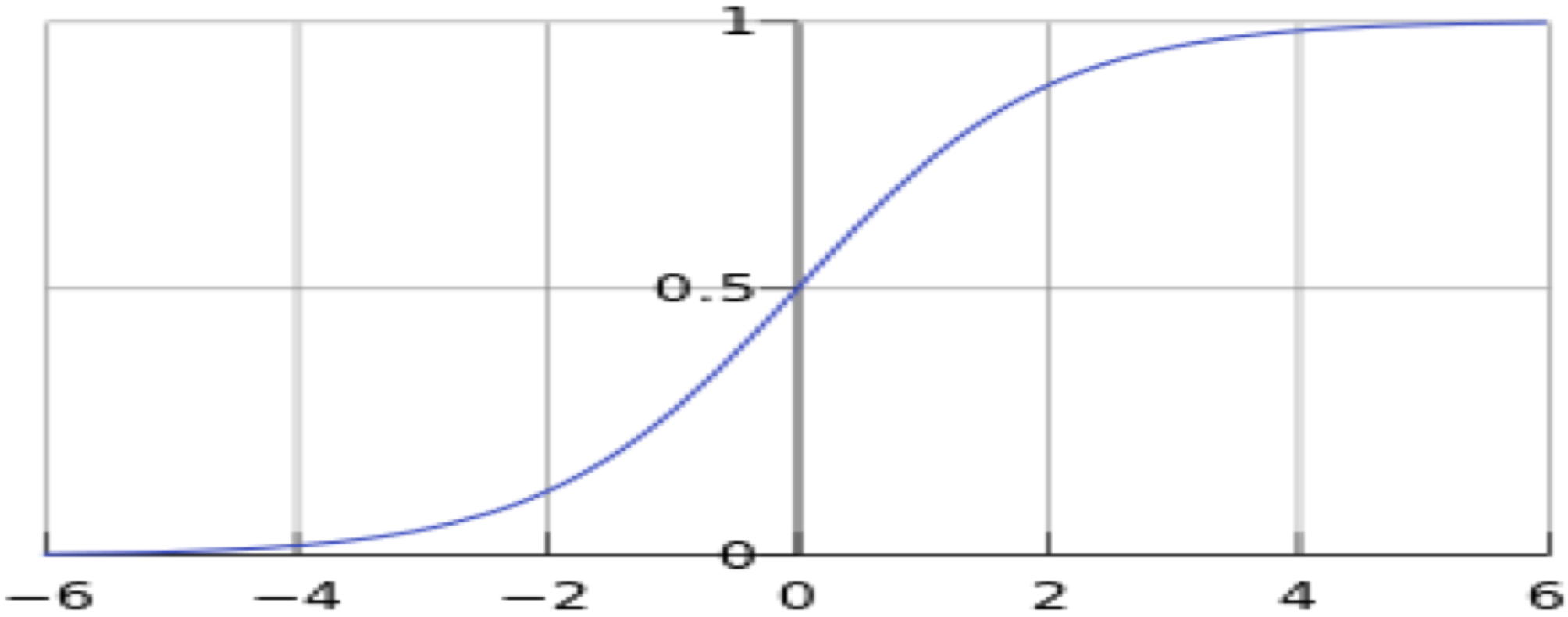
- Then we can re-write the equation by taking **log** on both sides:

$$LN[p/(1-p)] = \alpha + \beta x + u$$

- Where, LN[(p/(1-p)] is known as **logit or log-odds** and gives the probability of being survivor in relation to the non survivor.

- Log transformation of odds i.e. p/1-p enables us to link the model with linear regression

A logistics “S” function curve after transformation: **Sigmoid function used in ANN!**



Odds ratio

- The estimated value of slope (b) of the model is the **logit**

- The **logit** is difficult to interpret so we calculate the odds ratio of b as with back-transformation function as follows:

- Odd ratio = Anti-log of b = Exp(b) i.e. e^b

- If odds ratio is = 1, then equal chance (of surviving the titanic incidence)
- If odds ratio < 1, then less chance (of surviving the titanic incidence)
- If odds ratio > 1, then high chance (of surviving the titanic incidence)

General Notation (Stochastic/Population model):

- *For a single explanatory variable:*

- $g(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x$

Here $\boldsymbol{\pi}$ = Proportion in the Population!

- For multiple explanatory variables:

- $g(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m$

Assumptions of binary logistic regression

- **The dependent variable must be a dichotomous (2 categories i.e. Yes=1 and No=0)**
- The dependent variable categories must be mutually exclusive and exhaustive (in one group only)
- Logistic regression does not assume a linear relationship between the dependent and independent variables
- **Logistic regression assumes a linear relationship between the dependent variable and the log-odds (logit) of independent variables**
- The independent variables need NOT follow normal distribution nor have equal variances, residual analysis is also NOT a concern here
- Larger samples (>100) are needed than linear regression as it uses Maximum Likelihood Estimation method to compute the coefficients

Multicollinearity and Accuracy of the logistic regression model:

- Multivariate/multivariable binary logistic regression model must be free from multicollinearity among the independent variables
- **We must check it using VIF, Independent variables with VIF >2 means presence of multicollinearity for this model**
- The accuracy of the model must be based on the “confusion matrix”
- Confusion matrix is created using observed and predicted categories
- Predicted categories (0 and 1) is created using predicted probabilities
- **Accuracy must be calculated by adding diagonal values by total values**
- The error of the model will be: 1 - accuracy

Let’s fit the binary logistic regression with titanic.csv data sent to you in Google Classroom:

Read the Titanic data from the working directory

- library(readr)

- titanic <- read_csv("titanic.csv")

- View(titanic)

#Data wrangling

#Change the name of the data as “data” after removing the 3rd column containing names of the passengers

- **data <- titanic[,-3]**

#Check the structure of the data

- **str(data)**

Data wrangling continues:

#Pclass variable is imported as numeric. Pclass variable contained data of passengers in 1st, 2nd and 3rd class

- table(data$Pclass)

#Let us change it as factor variable

data$Pclass <- as.factor(data$Pclass)

#Check this variable

str(data$Pclass)

#Let us do the same for the sex variable as well

data$Sex <- as.factor(data$Sex)

str(data$Sex)

#Let us retain age as it is, we could scale it and other variables, if required!

Now, Let us fit the Binary Logistic regression: glm does not require "Survival" to be factor!

#Logistic regression with generalized linear model function

#family = binomial means dependent variable is binary: 0 and 1 coded

- model.full <- glm(Survived ~., data=data, family = binomial)
- summary(model.full)

 # McFadden's pseudo R-square

(mfpr2 <- 1 - (model.full$deviance/model.full$null.deviance))

I strongly discourage you to use this as accuracy for the binary logistic regression model.

- Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	4.109777	0.463602	8.865	< 2e-16 ***
Pclass2	-1.161491	0.300960	-3.859	0.000114 ***
Pclass3	-2.350022	0.304666	-7.713	1.22e-14 ***
Sexmale	-2.756710	0.200642	-13.739	< 2e-16 ***
Age	-0.043410	0.007790	-5.573	2.51e-08 ***
`S/S Aboard`	-0.401572	0.110795	-3.624	0.000290 ***
`P/C Aboard`	-0.106884	0.118767	-0.900	0.368151
Fare	0.002823	0.002468	1.144	0.252771

- ---
- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
- (Dispersion parameter for binomial family taken to be 1)
- Null deviance: 1182.77 on 886 degrees of freedom
- Residual deviance: 780.93 on 879 degrees of freedom
- Number of Fisher Scoring iterations: 5

Model Fit = McFadden's Pseudo R-square = 1 – (Residual Deviance/Null Deviance) = 0.3397464

The fitted model: Intercept and first five variables are statistically significant!

- exp(coef(model.full)) #Code to get odds ratios of predictors

(Intercept)	Pclass2 (vs Pclass1)	Pclass3 (vs Pclass1)
60.93314383	**0.31301925**	**0.09536703**

Sexmale (vs female)	Age	Siblings.Spouses.Aboard (yes)
0.06350032	**0.95751854**	**0.66926742**

Parents.Children.Aboard (yes)	Fare
0.89863033	1.00282685

How to interpret these results?

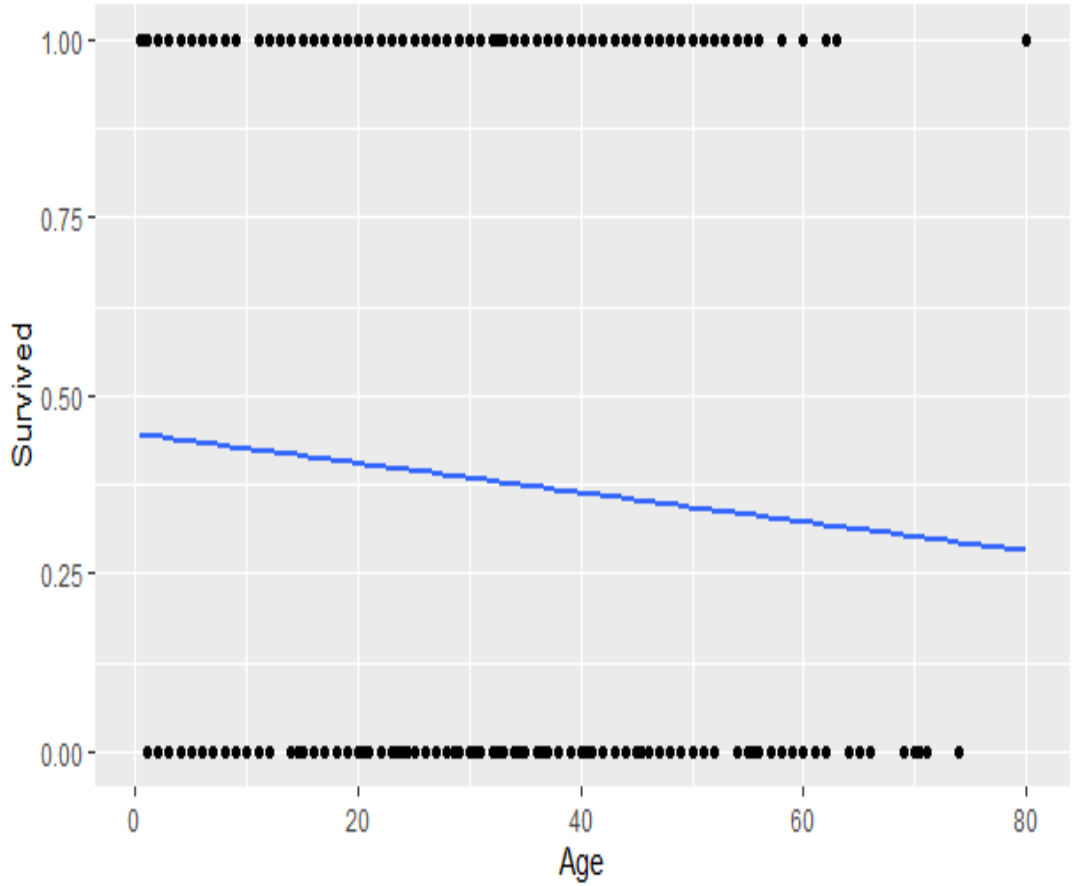
Logistic regression plot for a predictor (age): How to interpret it?

#plot glm with age as independent variable

library(ggplot2)

ggplot(data, aes(x=Age, y=Survived)) + geom_point() +

stat_smooth(method="glm", family="binomial", se=FALSE)



Multicollinearity of the full model: **Can we retain all the variables?**

- library(car)
- vif(model.full)

Variable:	GVIF	Df	GVIF^(1/(2*Df))
• Pclass	2.041787	2	1.195371
• Sex	1.201233	1	1.096008
• Age	1.477422	1	1.215492
• Siblings.Spouses.Aboard	1.290358	1	1.135939
• Parents.Children.Aboard	1.267656	1	1.125902
• Fare	1.578965	1	1.256569

Which VIF should be used here? GVIF or GVIF^(1/(2*Df))??

https://stats.stackexchange.com/questions/70679/which-variance-inflation-factor-should-i-be-using-textgvif-or-textgvif

Confusion matrix of the logistic regression model of full dataset (statistical approach):

#Prediction

- predict <- predict(model.full, type="response")

#Prediction to binary variable

predcted.fm <- as.numeric(ifelse(predict>0.5,1,0))

#Confusion matrix

- (cm <- table(predcted.fm, data$Survived))

Confusion matrix

		Observed	
predcted.fm		0	1
	0	**473**	103
	1	72	**239**

- Accuracy? Error?
- Sensitivity or True Positive Rate?
- Specificity? True Negative Rate?

Accuracy, error, sensitivity/fnr & specificity/fpr:

#Sensitivity, Specificity, Accuracy

- (accuracy <- sum(diag(cm))/sum(cm))
- (error <- 1 – accuracy)
- (sensitivity <- cm[1,1]/(cm[1,1]+cm[2,1]))
- (FNR <- 1 - sensitivity)
- (specificity <- cm[2,2]/(cm[2,1]+cm[2,2]))
- (FPR <- 1 - specificity)

#Accuracy

- **[1] 0.8027057**

#Error

- [1] 0.1972943

#Sensitivity

- **[1] 0.8678899**

#False negative rate

- [1] 0.1321101
- #Specificity
- **[1] 0.6988304**

#False positive rate

- [1] 0.3011696

Confusion matrix and diagnostic measures from “caret” package:

- #Confusion Matrix and diagnostic accuracy from caret package
- library(caret)
- predicted <- factor(ifelse(predict>0.5,1,0))
- reference <- factor(data$Survived)
- confusionMatrix(predicted, reference)

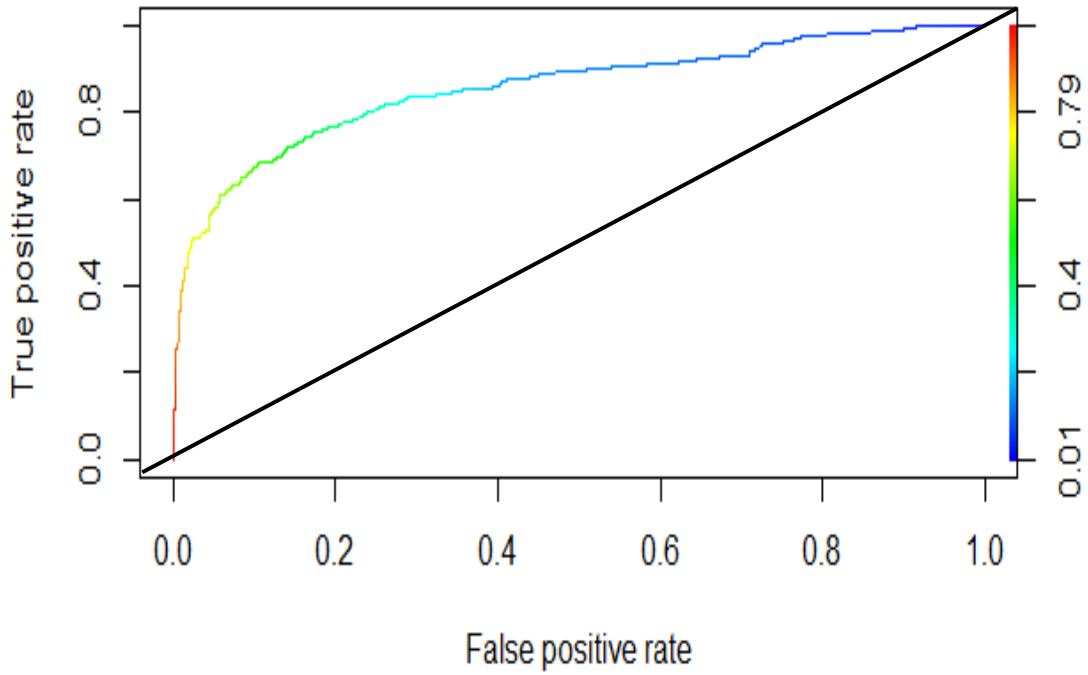
- Confusion Matrix and Statistics
- Reference
- Prediction 0 1
- 0 473 103
- 1 72 239

Accuracy : 0.8027 [95% CI : (0.775, 0.8284)]

- Sensitivity : 0.8679
- Specificity : 0.6988
- Positive Predicted Value : 0.8212
- Negative Predicted Value : 0.7685
- Prevalence : 0.6144
- Detection Rate : 0.5333
- Detection Prevalence : 0.6494
- Balanced Accuracy : 0.7834

Receiver Operating Characteristics (ROC) curve of full model:

- #ROCR Curve
- library(ROCR)
- ROCRpred <- prediction(predict, data$Survived)
- ROCRperf <- performance(ROCRpred, 'tpr','fpr')
- plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))



ROC curve to find Area Under Curve (AUC):

#pROC for accuracy with AUC

- library(pROC)
- predicted <- as.numeric(predicted)
- roc1 <- roc(reference, predicted)
- print(roc1)
- plot(roc1)

- Call:
- roc.default(response = reference, predictor = predicted)
- Data: predicted in 545 controls (reference 0) < 342 cases (reference 1).

- Area under the curve: 0.7834 **(This is another measure of model accuracy)**

Translating the logistic model with/for data science approach:

#Data partition

- ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))
- train <- data[ind==1,]
- test <- data[ind==2,]

#Fit logistic regression on train data

- model.train <- glm(Survived ~., data=train, family = binomial)
- summary(model.train)

Logistic model prediction and confusion matrix for the “train” data:

- #Confusion matrix of train data
- predict.train <- predict(model.train, type="response")
- predicted.train <- as.numeric(ifelse(predict.train>0.5,1,0))
- (cm <- table(predicted.train,train$Survived))

#Confusion matrix:

	Observed.train	
predicted.train	0	1
0	**327**	65
1	54	**183**

Get the accuracy, sensitivity, specificity etc. using the confusion matrix!

Accuracy and diagnostics of logistic regression model fitted for “train” data:

- #Sensitivity, Specificity, Accuracy
- (accuracy <- sum(diag(cm))/sum(cm))
- (sensitivity <- cm[1,1]/(cm[1,1]+cm[2,1]))
- (FNR <- 1 - sensitivity)
- (specificity <- cm[2,2]/(cm[1,2]+cm[2,2]))
- (FPR <- 1 - specificity)

#Accuracy
- **0.8108108**

#Error
- 0.1891892

#Sensitivity
- **0.8582677**

#False negative rate
- 0.1417323

#Specificity
- **0.737903**
- #False positive rate
- 0.2620968

Accuracy and diagnostics of logistic regression model with the “caret” package:

- #Confusion Matrix and diagnostic accuracy from **caret** package
- predicted.train <- factor(ifelse(predict.train>0.5,1,0))
- reference.train <- factor(train$Survived)
- confusionMatrix(predicted.train, reference.train)

- Confusion Matrix and Statistics

		Reference	
Prediction		0	1
	0	**327**	65
	1	54	**183**

Accuracy : 0.8108

95% CI : (0.778, 0.8407)

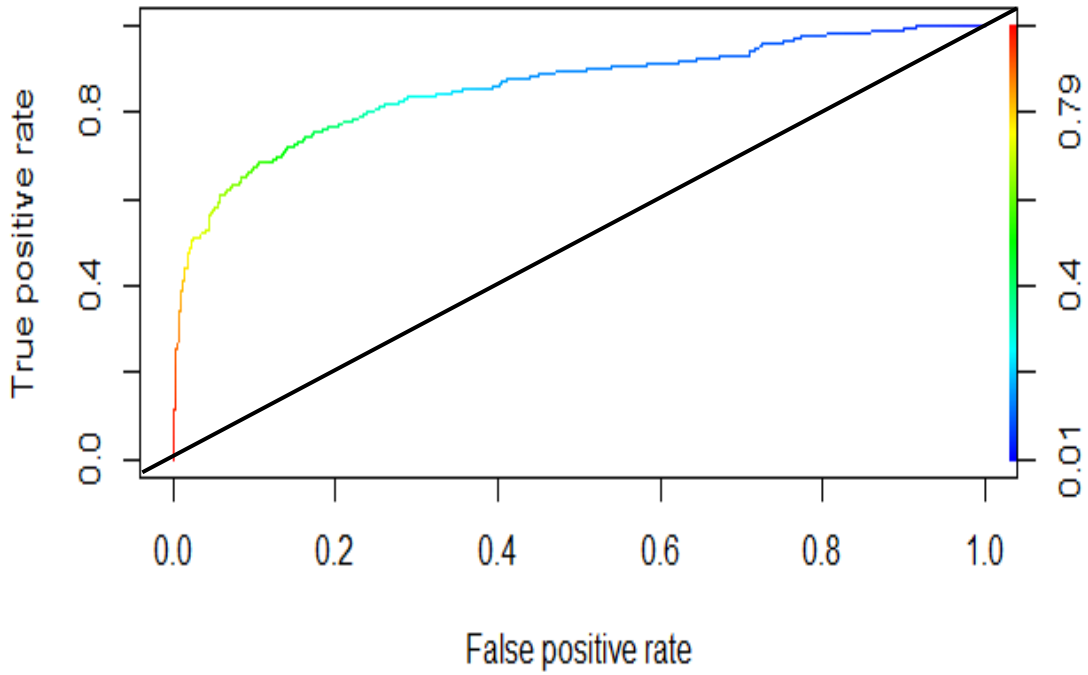
Sensitivity : 0.8583

Specificity : 0.7379

ROC curve of “train” data:

- #ROC Curve of Train data with ROCR package
- ROCRpred <- prediction(predict.train, test$Survived)
- ROCRperf <- performance(ROCRpred, 'tpr','fpr')
- plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))

We need to find the Area Under Curve too!



Area Under Curve (AUC) of “train” data:

#AUC of ROC curve with pROC package

- predicted.train <- as.numeric(predicted.train)
- roc3 <- roc(reference.train, predicted.train)
- print(roc3)
- plot(roc3)

- Call:
- roc.default(response = reference.train, predictor = predicted.train)

- Data: predicted.train in 381 controls (reference.train 0) < 248 cases (reference.train 1).

- Area under the curve: 0.7981

Prediction and accuracy for “test” data based on fitted model on the “train” data:

#Prediction with fitted logistic regression model for train data

- predict.test <- predict(model.train, test, type="response")
- predicted.test <- factor(ifelse(predict.test>0.5,1,0))
- reference.test <- factor(test$Survived)
- confusionMatrix(predicted.test, reference.test)

- Confusion Matrix and Statistics

		Reference	
Prediction		0	1
	0	**147**	36
	1	17	**58**

Accuracy : **0.7946**

95% CI : (0.7401, 0.8422)

Sensitivity : 0.8963

Specificity : 0.6170

Questions?

https://www.kaggle.com/questions-and-answers/250648

- **Why accuracy of train model is higher than the test model here?**
- What does it mean?
- When does such result can happen?
- How to handle such problem?

- Why accuracy of test model is higher than the train model?
- What does it mean?
- When does such result can happen?
- How to handle such problem?

When should we call “train” and “test” results as: **What is Bias-Variance tradeoff?**

- Underfitting?
- When does it happen?
- How to overcome this problem?

- Overfitting?
- When does it happen?
- How to overcome this problem?

More here:

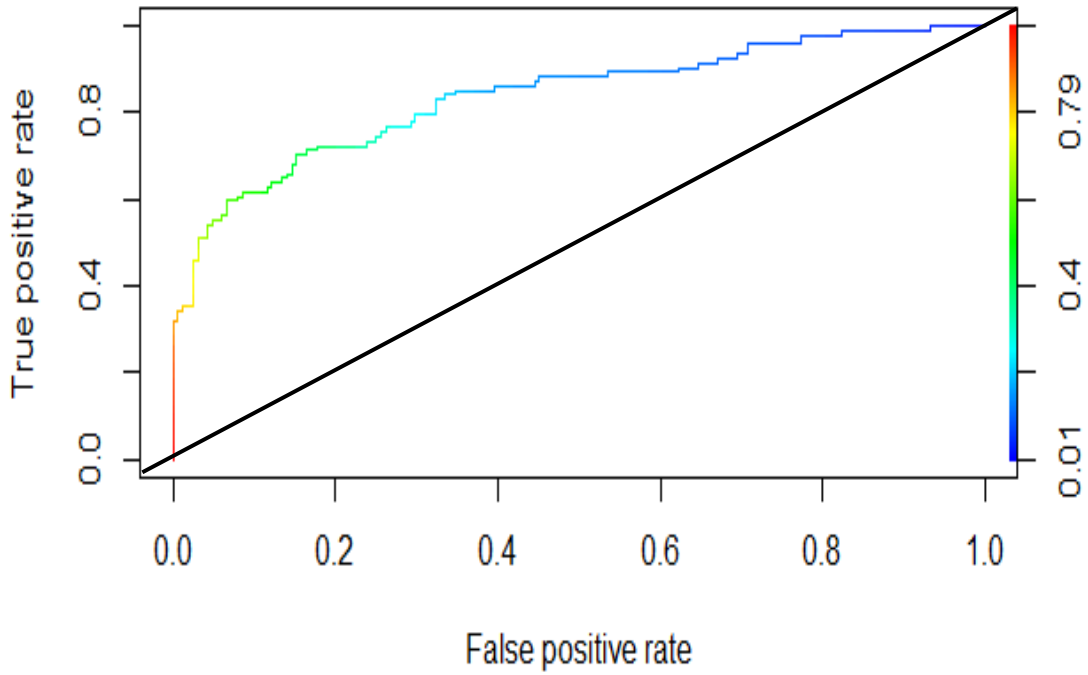
https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765

https://www.v7labs.com/blog/overfitting-vs-underfitting

ROC curve for “test” data”

#ROCR Curve of Test data

- ROCRpred <- prediction(predict.test, test$Survived)
- ROCRperf <- performance(ROCRpred, 'tpr','fpr')
- plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))



AUC for “test” data:
This will confirm the “accuracy” of the model!

#Use pROC package for AUC

- library(pROC)
- predicted.test <- as.numeric(predicted.test)
- roc4 <- roc(reference.test, predicted.test)
- print(roc4)
- plot(roc4)

- Call:
- roc.default(response = reference.test, predictor = predicted.test)

- Data: predicted.test in 164 controls (reference.test 0) < 94 cases (reference.test 1).

- Area under the curve: 0.7567

We can now check this result with:
KNN and ANN-MLP are self-learning!

- **KNN classifier**
  - **Do it on your own** following the slides and the links provided in the last class
- **ANN-MLP classifier**
  - **Do it on your own** following the slides and the links provided in the last class
- **Naïve Bayes classifier**
  - We will do it now
- **Support Vector Machine classifier**
  - We will do it now
- **Decision Tree**
  - We will use it in next class

Naïve Bayes classifier: More at “An Introduction to Statistical Learning with Applications in R” book!

- Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye’s theorem with strong (Naive) independence assumptions between the features or variables.
- The Naive Bayes algorithm is called “Naive” because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

where,

P(A|B) = Conditional probability of A given B.
P(B|A) = Conditional probability of B given A.
P(A) = Probability of event A.
P(B) = Probability of event B.

Reference: https://www.geeksforgeeks.org/naive-bayes-classifier-in-r-programming/

Fitting Naïve Bayes classifier in the train data:

- library(e1071)

 # Setting Seed

- set.seed(120)

#Fitting model

- model.nb <- naiveBayes(Survived~., data=train)

#Checking model

- model.nb

Naive Bayes Classifier for Discrete Predictors

Call: naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities (dependent var?):

Y	0	1
	0.6152542	0.3847458

Conditional probabilities:

	Pclass		
Y	1	2	3
0	0.1570248	0.1707989	0.6721763
1	0.4317181	0.2026432	0.3656388 **etc.**

Prediction for the test data and confusion matrix:

 # Predicting on test data'

- y_pred <- predict(model.nb, newdata = test)

 # Confusion Matrix

- cm <- table(test$Survived, y_pred)
- cm

#Confusion matrix

	Reference	
y_pred	0	1
0	164	42
1	18	73

Get the accuracy, sensitivity, specificity etc. using the confusion matrix!

Accuracy and diagnostics of the prediction:

- library(caret)
- # Model Evaluation
- confusionMatrix(cm)

#Confusion Matrix and Statistics

y_pred	0	1
0	164	42
1	18	73

Accuracy : 0.798 (better than the logistic regression classifier?)

95% CI : (0.7478, 0.8422)

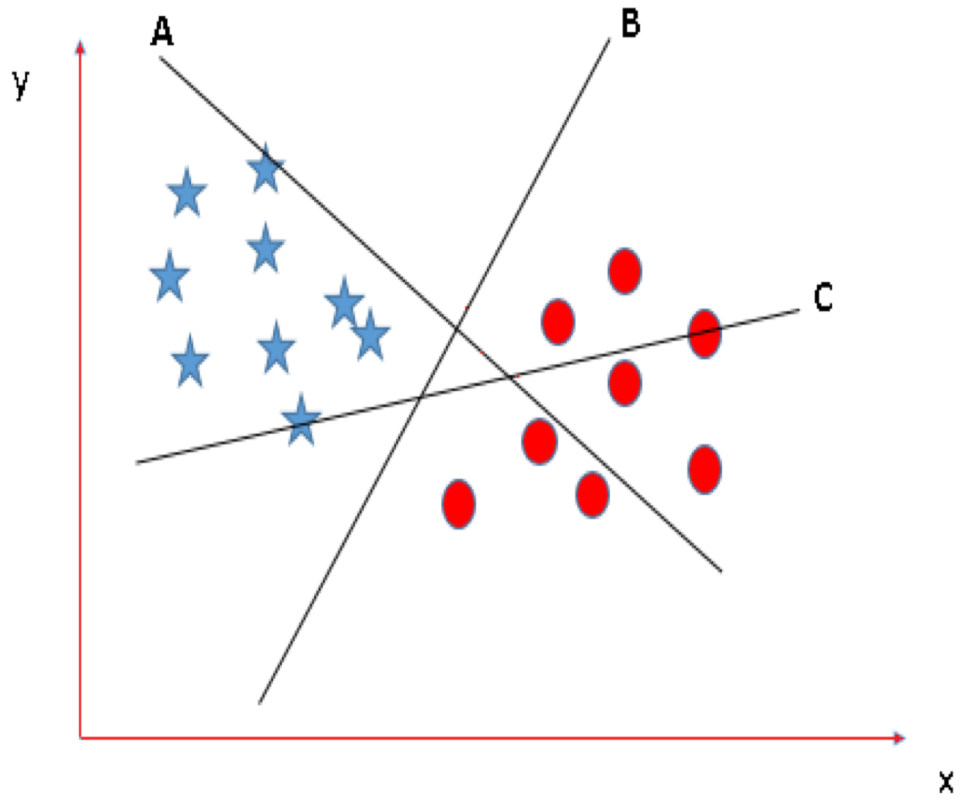
Sensitivity : 0.9011 (Better?)

Specificity : 0.6348 (Better?)

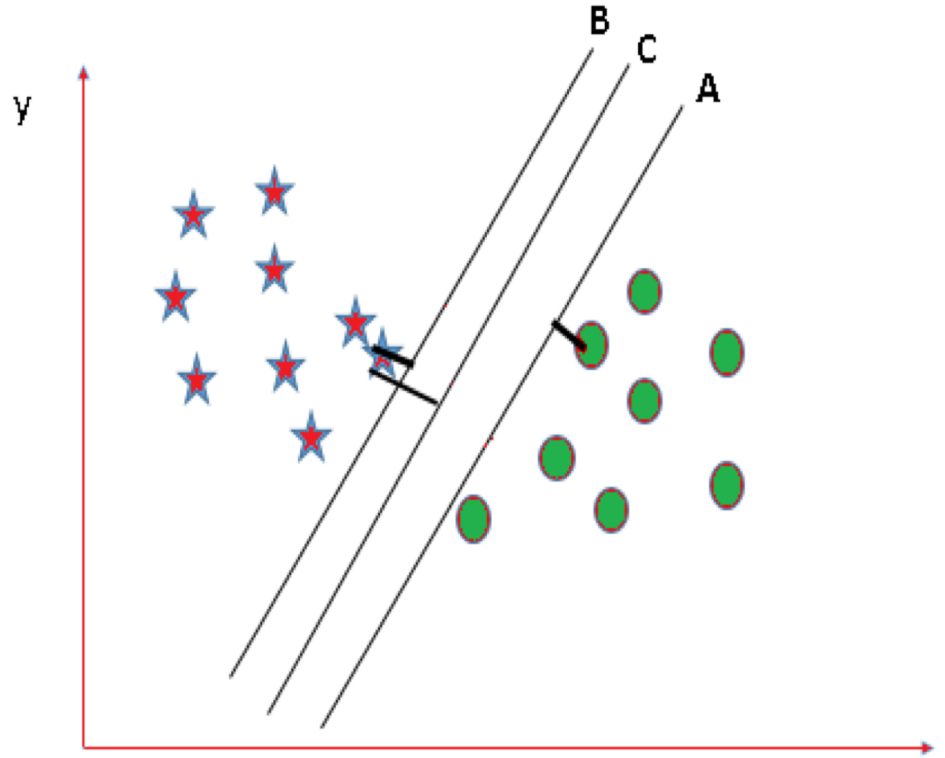
SVM classifier: More at “An Introduction to Statistical Learning with Applications in R” book!

- In machine learning, Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- Variable is called feature in data science & feature are shown in the column!
- It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in **n-dimensional space** (where n is a number of features), with the value of each feature being the value of a particular coordinate.
- Then, classification is performed by finding the hyper-plane that best differentiates the two classes.

SVM classifier with “hyperplanes”, say, titanic data with star for 1 and circle for 0:



Starts with the wrong classification but takes help of support vectors in each iteration to make it right!



SVM found few support vectors that that helped to classify the two groups with the helps of 3 hyperplanes or a tube formed by these 3 hyperplanes!

Fitting SVM classifier on “titanic train” data:

SVM has four “kernel”, we will use linear now!

#Fitting the SVM classifier model

- model.svm = svm(formula = Survived~.,
- data = train,
- type = 'C-classification',
- kernel = 'linear')

#Check the model

- model.svm

#You can check with other three kernel too e.g. radial basis function

Call:

- svm(formula = Survived ~ ., data = train, type = "C-classification",
- kernel = "linear")

Parameters:

- SVM-Type: C-classification
- SVM-Kernel: linear
- cost: 1

- Number of Support Vectors: 270

More here: https://www.rdocumentation.org/packages/e1071/versions/1.7-9/topics/svm

Prediction for the test data and confusion matrix:

 # Predicting on test data'

- y_pred.svm <- predict(model.svm, newdata = test)

 # Confusion Matrix

- cm.svm <- table(y_pred.svm,test$Survived)
- cm.svm

#Confusion matrix

		Reference.test	
y_pred.svm		0	1
	0	154	30
	1	28	85

SVM can be fitted as classification or regression model:
C-classification,
nu-classification
one-classification (for novelty detection)
eps-regression
nu-regression

Accuracy and diagnostics of the prediction:

#Load “caret” library if not done already

- library(caret)

Model Evaluation

confusionMatrix(cm.svm)

- Confusion Matrix and Statistics

	Reference.test	
y_pred.svm	0	1
0	154	30
1	28	85

Accuracy : 0.8047 (Better than Naïve Bayes and Logistic regression classifiers!)

95% CI : (0.755, 0.8482)

Sensitivity : 0.8462 (Better?)

Specificity : 0.7391 (Better?)

You can fit the SVM for this data with other kernels and check if that improves the accuracy of the model. **Use polynomial, radial basis and sigmoid functions.**

Introduction to Statistical Learning with R (2nd Edition): https://www.statlearning.com/

- https://hastie.su.domains/ISLR2/ISLRv2_website.pdf

- Details on classification models are available in these units or chapters:
  - Unit 4 – Classification
  - Unit 4.3 – Logistic regression
  - Unit 4.4.4 – Naïve Bayes
  - Unit 5 – Resampling methods
  - Unit 6 – Linear models
  - Unit 7 – Moving beyond linearity
  - Unit 8 – Tree-Based methods
  - Unit 9 – Support Vector Machine
  - Unit 10 – Deep Learning

Question/queries?

- Next class

- Classification models for supervised learning
  - Decision trees

- Ensemble learning
  - Random forests
  - Bagging
  - Boosting

Thank you!

@shitalbhandary