Data Analytics using R

SUVEN CONSULTANTS | NIRAJ SHARMA

Topics Covered

- **Machine learning**
  - Supervised
  - Unsupervised
  - Reinforcements

- Linear Regression
- Multiple Linear Regression
- Logistic Regression
- KNN
- Decision Tree
- Random Forest
- Neural Network
- Apriori Algorithm

- Machine learning comes in many different flavors, depending on the algorithm and its objectives. You can divide machine learning algorithms into three main groups based on their purpose:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

SUPERVISED LEARNING

- ***Supervised learning*** occurs when an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples.

- You need to distinguish between regression problems, whose target is a numeric value, and classification problems, whose target is a qualitative variable, such as a class or a tag.

- E.g A regression task determines the average prices of houses in the Boston area, and a classification tasks distinguishes between kinds of iris flowers based on their sepal and petal measures.

UNSUPERVISED LEARNING

- ***Unsupervised learning*** occurs when an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own.
- This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values.
- They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.
- As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects.
- E.g The marketing automation algorithm derives its suggestions from what you've bought in the past.

REINFORCEMENT LEARNING

- ***Reinforcement learning*** occurs when you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes.
- Reinforcement learning is connected to applications for which the algorithm must make decisions (so the product is prescriptive, not just descriptive, as in unsupervised learning), and the decisions bear consequences.
- An interesting example of reinforcement learning occurs when computers learn to play video games by themselves.

Linear Regression

- Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

- In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1.

- Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

Linear Regression

- The general mathematical equation for a linear regression is –
  - y = ax + b
- Following is the description of the parameters used –
  - **y -** is the response variable.
  - **x -** is the predictor variable.
  - **a** and **b** **-** are constants which are called the coefficients.

Linear Regression

- **Steps to Establish a Regression**
  A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

- The steps to create the relationship is -

  - Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
  - Create a relationship model using the **lm()** functions in R.
  - Find the coefficients from the model created and create the mathematical equation using data.
  - Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
  - To predict the weight of new persons, use the **predict()** function in R.

Linear Regression

- Below is the sample data representing the observations -
  # Values of height
  151, 174, 138, 186, 128, 136, 179, 163, 152, 131
  # Values of weight.
  63, 81, 56, 91, 47, 57, 76, 72, 62, 48

- lm() Function
  This function creates the relationship model between the predictor and the response variable.

- Syntax
  The basic syntax for **lm()** function in linear regression is –
  - lm(formula,data)

  Following is the description of the parameters used –
  - **formula** is a symbol presenting the relation between x and y.
  - **data** is the vector on which the formula will be applied.

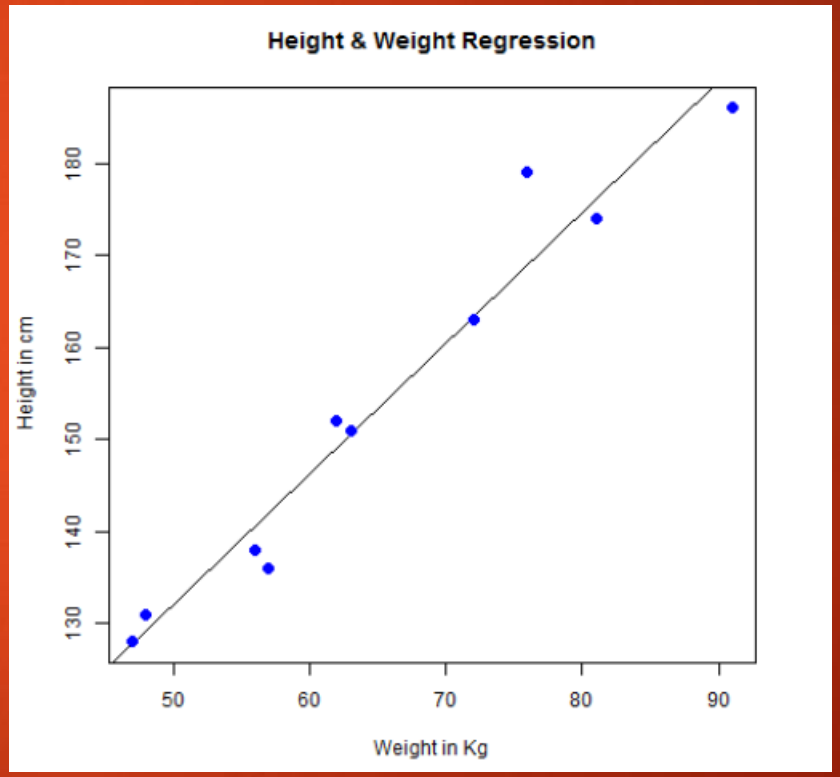
Linear Regression

- Code

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)


# Apply the lm() function.
relation <- lm(y~x)
print(summary(relation)) # Gives Significance of Features & Accuracy


# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <- predict(relation,a) # For predicting values based on relation
print(result)
```

Linear Regression


```
# Plot the chart.
plot( x,y,col = "blue",
      main = "Height & Weight Regression",
      abline(lm(y~x)),cex = 1.3,
      pch = 16,
      xlab = "Weight in Kg",
      ylab = "Height in cm")
```



Refer Salary Dataset for Case Study.

Multiple Linear Regression

- Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

- The general mathematical equation for multiple regression is –
  - y = a + b1x1 + b2x2 +...bnxn

- Following is the description of the parameters used –
  - **y -** is the response variable.
  - **a, b1, b2...bn -** are the coefficients.
  - **x1, x2, ...xn -** are the p

Multiple Linear Regression

- Input Data
  Consider the data set "mtcars" available in the R environment.

- It gives a comparison between different car models in terms of mileage per gallon (mpg), cylinder displacement("disp"), horse power("hp"), weight of the car("wt") and some more parameters.

- The goal of the model is to establish the relationship between "mpg" as a response variable with "disp","hp" and "wt" as predictor variables.

Multiple Linear Regression

- Code

```
input <- mtcars[,c("mpg","disp","hp","wt")]

# Create the relationship model.
model <- lm(mpg~disp+hp+wt, data = input)

# Show the model.
print(model)

# Get the Intercept and coefficients as vector elements.
a <- coef(model)[1]

Xdisp <- coef(model)[2]

Xhp <- coef(model)[3]

Xwt <- coef(model)[4]

print(a)

print(Xdisp)

print(Xhp)

print(Xwt)

newdata <- data.frame(hp=102,wt=2.91,disp=221)

Y <- predict(model,newdata)

print(Y)
```

Refer Startups Profit Dataset for Case Study.

Logistic Regression

- The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1.

- It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

- The general mathematical equation for logistic regression is –
  - y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

  Following is the description of the parameters used –
  - **y** is the response variable.
  - **x** is the predictor variable.
  - **a** and **b** are the coefficients which are numeric constants.

Logistic Regression

- glm(formula,data,family)
  Following is the description of the parameters used –
  - **formula** is the symbol presenting the relationship between the variables.
  - **data** is the data set giving the values of these variables.
  - **family** is R object to specify the details of the model. It's value is binomial for logistic regression.

- Example
  - The in-built data set "mtcars" describes different models of a car with their various engine specifications.
  - In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1).
  - We can create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

Logistic Regression

- Code

```
input <- mtcars[,c("am","hp","wt")]
head(input)

# Split into Training & Testing
training_am = input[1:20,] #subset(input, split_part == T)
testing_am  = input[21:32,]#subset(input, split_part == F)

View(training_am)
am.data = glm(formula = am ~  hp + wt, family = binomial,  data = training_am)
am.data = glm(formula = am ~ cyl + hp + wt, family = binomial,  data = input)

print(summary(am.data))
varImp(am.data)
```

Logistic Regression


```
am.pred = predict(am.data,newdata=testing_am,type = 'response')
print(am.pred)

## Values = 1.000 is Automatic rest all are Manual

# Accuracy of Model
am.pred_num <- ifelse(am.pred > 0.5, 1, 0)
am.pred_act <- factor(am.pred_num, levels=c(0, 1))
am.pred_avi <- testing_am$am

mean(am.pred_act == am.pred_avi)
table(am.pred_act,am.pred_avi) # Predicted Vs Actual Data
```

KNN Algorithm

- This algorithm is a supervised learning algorithm, where the destination is known, but the path to the destination is not. Understanding nearest neighbors forms the quintessence of machine learning.

- What is kNN Algorithm?
  - Let's assume we have several groups of labeled samples. The items present in the groups are homogeneous in nature. Now, suppose we have an unlabeled example which needs to be classified into one of the several labeled groups. How do you do that? Unhesitatingly, using kNN Algorithm.
  - k nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. This algorithms segregates unlabeled data points into well defined groups

KNN Algorithm

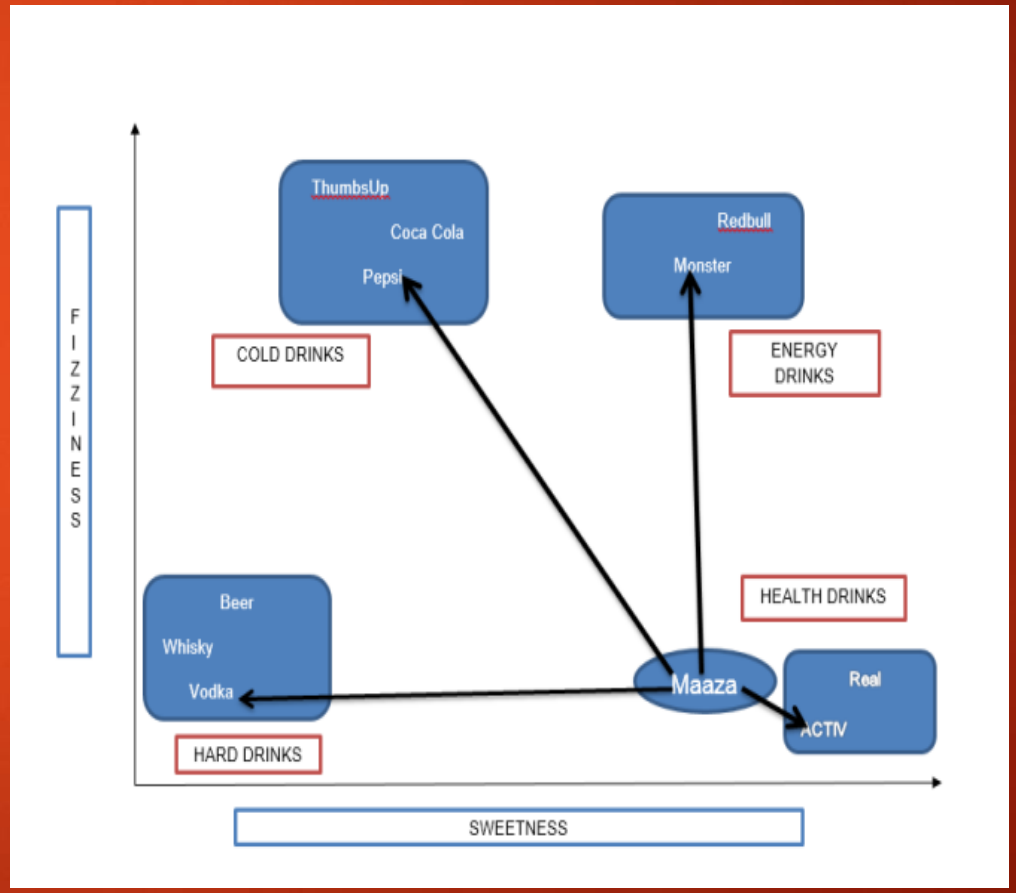
- Example of kNN Algorithm
  - Let's consider 'drinking items' which are rated on two parameters on a scale of 1 to 10. The two parameters are "sweetness" and "fizziness".

Ingredient	Sweetness	Fizziness	Type of Drink
Monster	8	8	Energy booster
ACTIV	9	1	Health drink
Pepsi	4	8	Cold drink
Vodka	2	1	Hard drink

  - "Sweetness" determines the perception of the sugar content in the items. "Fizziness" ascertains the presence of bubbles in the drink due to the carbon dioxide content in the drink.

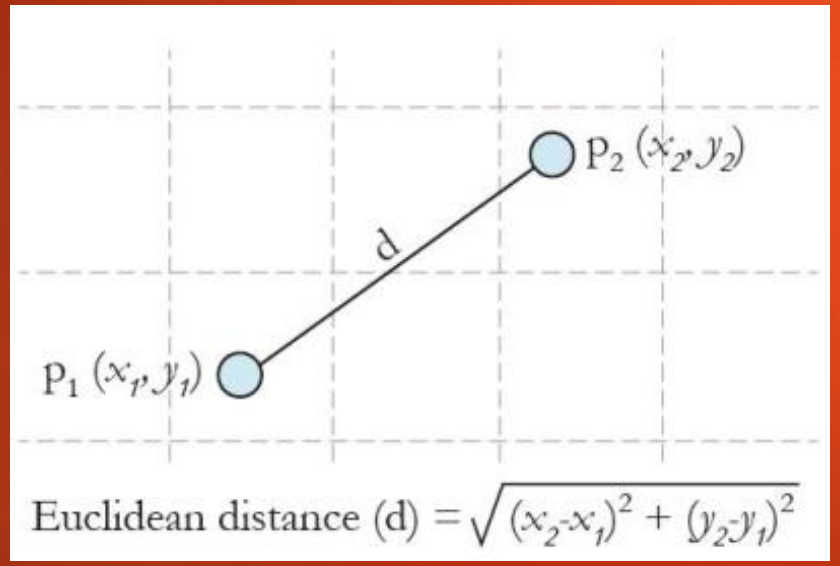
KNN Algorithm

- it is clear we have bucketed items into 4 groups namely, 'COLD DRINKS', 'ENERGY DRINKS', 'HEALTH DRINKS' and 'HARD DRINKS'.
- The question here is, to which group would 'Maaza' fall into? This will be determined by calculating distance.
- Now, calculating distance between 'Maaza' and its nearest neighbors requires the usage of a distance formula, Euclidean distance formula i.e. the shortest distance between the 2 points which may be obtained using a ruler.



KNN Algorithm

- Using the co-ordinates of Maaza (8,2) and Vodka (2,1), the distance between 'Maaza' and 'Vodka' can be calculated as:



dist(Maaza, Vodka) = 6.08

Ingredient	Sweetness	Fizziness	Type of Drink	Distance to Maaza
Monster	7	8	Energy booster	6.08
ACTIV	9	1	Health drink	1.41
Pepsi	4	8	Cold drink	7.21
Vodka	2	1	Hard drink	6.08

- Distance between Maaza and ACTIV being the least, it may be inferred that Maaza is same as ACTIV in nature which in turn belongs to the group ofdrinks (Health Drinks).

KNN Algorithm

- If k=1, the algorithm considers the nearest neighbor to Maaza i.e, ACTIV.
- if k=3, the algorithm considers '3' nearest neighbors to Maaza to compare the distances (ACTIV, Vodka, Monster) – ACTIV stands the nearest to Maaza.

- **Example :**
  - We will use a data set of 100 patients to implement the knn algorithm and thereby interpreting results & predict type of cancer (Benign & Malignant).
  - The data set consists of 100 observations and 10 variables (out of which 8 numeric variables and one categorical variable and is ID) which are as follows:
    Radius, Texture, Perimeter, Area, Smoothness, Compactness, Symmetry, Fractal dimension

KNN Algorithm

- Code

```
prc <- read.csv("Prostate_Cancer.csv",stringsAsFactors = FALSE)

# install.packages("gmodels")
library(gmodels)
prc <- prc[-1]

table(prc$diagnosis_result)

prc$diagnosis <- factor(prc$diagnosis_result,
                        levels = c("B", "M"), labels = c("Benign", "Malignant"))
```

KNN Algorithm


```
print(prc$diagnosis)


normalize <- function(x) {
 return ((x - min(x)) / (max(x) - min(x))) }


head(prc)


prc_n <- as.data.frame(lapply(prc[2:9], normalize))
head(prc_n)
summary(prc_n$radius)
```

```
library(caTools)
split <- sample.split(prc$diagnosis_result , SplitRatio = 0.8)
prc_train <- subset(prc_n , split == T)
nrow(prc_train)
prc_test <- subset(prc_n , split == F)
nrow(prc_test)
prc_train_labels <- subset(prc , split == T)
nrow(prc_train_labels)
prc_train_labels <- prc_train_labels$diagnosis_result
prc_train_labels
```

KNN Algorithm


```
prc_test_labels <- subset(prc , split == F)

nrow(prc_test_labels)


prc_test_labels <-
prc_test_labels$diagnosis_result


prc_test_labels


library(class)

library(gmodels)
```

```
# Creating Model & Testing

prc_test_pred <- knn(train = prc_train, test =
prc_test, cl = prc_train_labels, k=5)


print(prc_test_pred)


# Validating the result & accuracy

CrossTable(x=prc_test_labels,y=prc_test_pred)
```

Decision Tree

- Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions.
- It is mostly used in Machine Learning and Data Mining applications using R. Examples of use of decision tress is - predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these.
- Example
  - We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

Decision Tree

- Code

```
library("party")
print(readingSkills)
library("caTools")
library(rpart)
library(rpart.plot)

dataset = readingSkills
sam     = sample.split(dataset$age, SplitRatio = 0.70)
print(sam)

# Split data
training= subset(dataset,sam==TRUE)
testing = subset(dataset,sam==FALSE)

# Create Model
model   = rpart(nativeSpeaker ~. , data=training)
rpart.plot(model)
model$variable.importance

# Predict
result = predict(model,testing,type = "class")
table(result,testing$nativeSpeaker)
```

Random Forest

- In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output.
- A new observation is fed into all the trees and taking a majority vote for each classification model.
- An error estimate is made for the cases which were not used while building the tree. That is called an **OOB (Out-of-bag)** error estimate which is mentioned as a percentage.
- Syntax
  The basic syntax for creating a random forest in R is –
  - randomForest(formula, data)

  Following is the description of the parameters used –
  - **formula** is a formula describing the predictor and response variables.
  - **data** is the name of the data set used.

Random Forest

- Example:
  - ReadingSkills Dataset
- Code

```
library(party)
library(randomForest)

= readingskills
print(head(dataset))
set.seed(10)
```

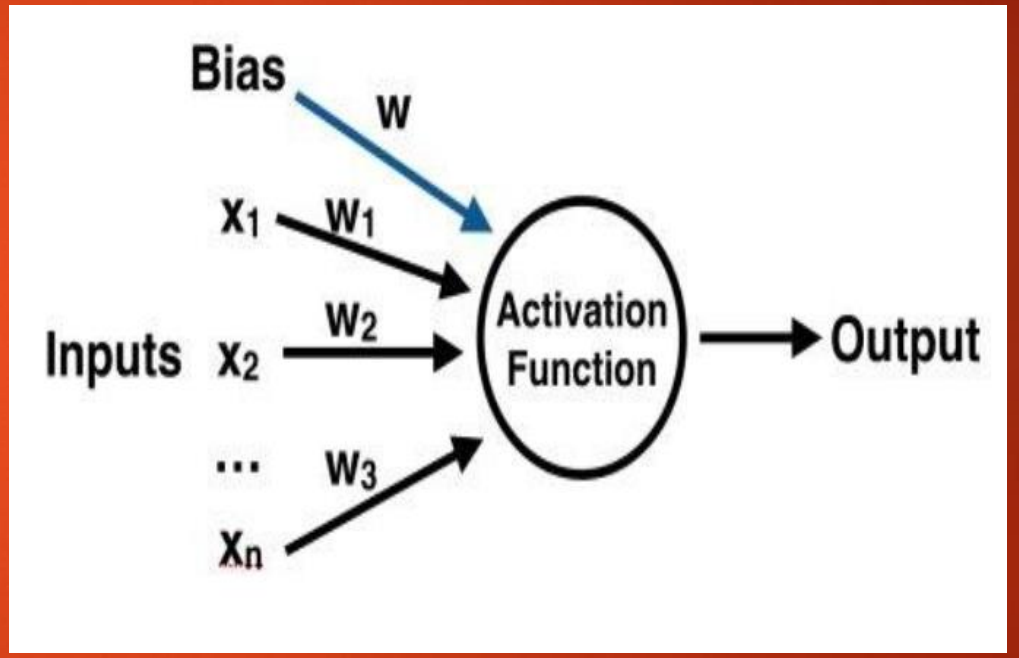
```
# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score, data = dataset, ntree = 1000)

# View the forest results.
print(output.forest)
plot(output.forest)

# Importance of each predictor.
varImpPlot(output.forest)
```

Neural Network

- Neural Networks are a machine learning framework that attempts to mimic the learning pattern of natural biological neural networks. Biological neural networks have interconnected neurons with dendrites that receive inputs, then based on these inputs they produce an output signal through an axon to another neuron.
- We will try to mimic this process through the use of Artificial Neural Networks (ANN), which we will just refer to as neural networks from now on. The process of creating a neural network begins with the most basic form, a single perceptron.
- **The Perceptron**
  Let's start our discussion by talking about the Perceptron! A perceptron has one or more inputs, a bias, an activation function, and a single output. The perceptron receives inputs, multiplies them by some weight, and then passes them into an activation function to produce an output.

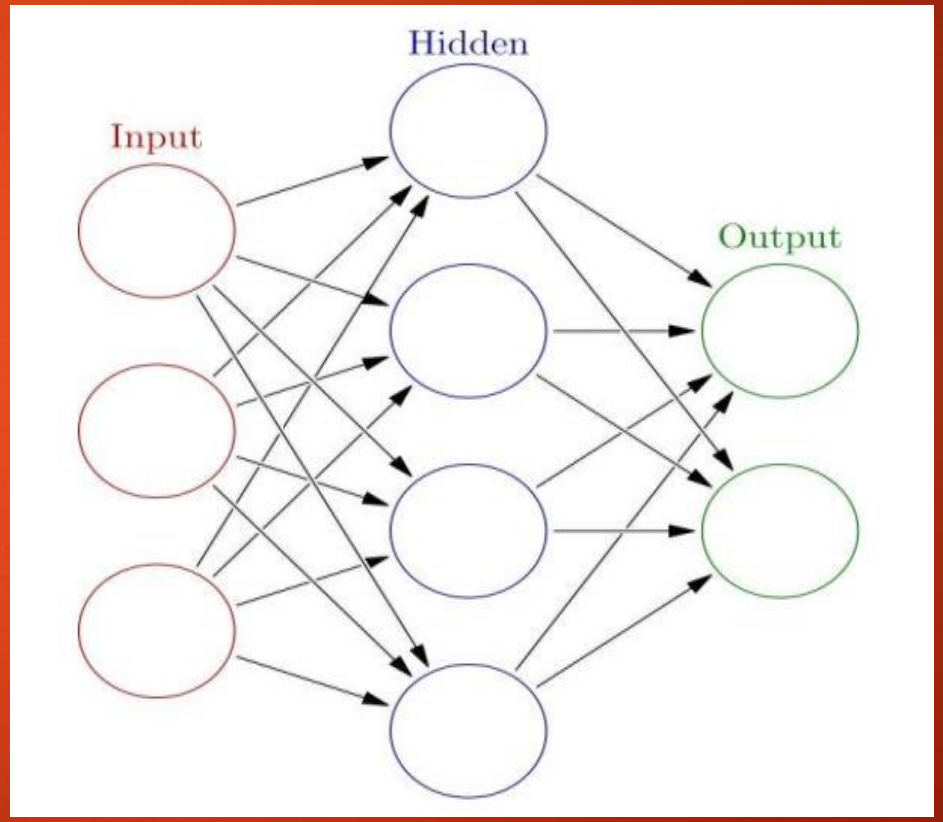
Neural Network

- There are many possible activation functions to choose from, such as the logistic function, a trigonometric function, a step function etc. We also make sure to add a bias to the perceptron, this avoids issues where all inputs could be equal to zero (meaning no multiplicative weight would have an effect).
- Once we have the output we can compare it to a known label and adjust the weights accordingly (the weights usually start off with random initialization values).
- We keep repeating this process until we have reached a maximum number of allowed iterations, or an acceptable error rate.



Neural Network

- To create a neural network, we simply begin to add layers of perceptrons together, creating a multi-layer perceptron model of a neural network. You'll have an input layer which directly takes in your feature inputs and an output layer which will create the resulting outputs.
- Any layers in between are known as hidden layers
because they don't directly "see" the feature inputs or outputs.



Neural Network

- Example:
- **Data**
  We'll use ISLR's built in College Data Set which has several features of a college and
  a categorical column indicating whether or not the School is Public or Private.

- Code

```
library("ISLR")
print(head(College,2))
#Extracting Max & Min of Feature
maxs <- apply(College[,2:18], 2, max)
mins <- apply(College[,2:18], 2, min)

# Normalizing data
scaled.data <- as.data.frame(scale(
College[,2:18] , center = mins, scale =
maxs- mins))
```

Neural Network


```
print(head(scaled.data,2))
Private = as.numeric(College$Private)-1
data = cbind(Private,scaled.data)

# Split Data in Training & Testing Set
library(caTools)
set.seed(101)
split = sample.split(data$Private, SplitRatio = 0.70)
train = subset(data, split == TRUE)
test = subset(data, split == FALSE)
```

```
# Create Formula
feats <- names(scaled.data)
print(feats)
f <- paste(feats,collapse=' + ')
print(f)
f <- paste('Private ~',f)
print(f)
f <- as.formula(f)
print(f)
```

Neural Network


```
#install.packages('neuralnet')

library(neuralnet)

nn <- neuralnet(f,train,hidden = c(10,10), linear.output=FALSE)

# Plot

plot(nn)


# Predict

predicted.nn.values <- compute(nn, test[2:18])

print(head(predicted.nn.values$net.result))
```

```
# Validate Accuracy

predicted.nn.values$net.result <- sapply(predicted.nn.values$net.result,round,digits=0)


table(test$Private,predicted.nn.values$net.result)
```

Apriori Algorithm (Market Basket Analysis)

- Association Rule Mining is used when you want to find an association between different objects in a set, find frequent patterns in a transaction database, relational databases or any other information repository.
- The applications of Association Rule Mining are found in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering and classification. It can tell you what items do customers frequently buy together by generating a set of rules called **Association Rules**. In simple words, it gives you output as rules in form **if this then that**.
- Clients can use those rules for numerous marketing strategies:
  - Changing the store layout according to trends
  - Customer behavior analysis
  - Catalogue design
  - Cross marketing on online stores
  - What are the trending items customers buy
  - Customized emails with add-on sales

Apriori Algorithm (Market Basket Analysis)

- # support : It is a measure of how frequently the collection of items occur together as a percentage of all transactions.
- # eg : s = (milk + cheese) / total_customer
- # Confidence : the ratio of the number of transactions that include all items in {B} as well as the number of transactions that include all items in {A} to the number of transactions that include all items in {A}.
- # eg : c = (milk + cheese) / milk

- Data

  mba_data(data frame in R) has two variable customer_id and products.

  Each customer id has bought some products for example:

  # customer id 1 has bought

  # Bread

  # Butter

  # Eggs

  # Milk

Apriori Algorithm (Market Basket Analysis)

- Code

```
# Import csv
mba_data<-
read.csv("MBA_data_new.csv")
print(mba_data)

# Transform in transaction list
trans <- split(mba_data$Products,
mba_data$Customer_Id, "transactions")
head(trans)

# import arules package
library(arules)

# Create rule using apriori
rules = apriori(trans ,
parameter=list(support=0.10,
confidence=0.5, maxlen=2, minlen=2))

# Find Relations in Buying
inspect(rules)
```

Instructor Contact Information

nirajshar67@gmail.com /
rocky@suvenconsultants.com

9167687087 / 9892544177

10 am – 7 pm

https://www.linkedin.com/in/niraj7654/