Statistical Computing with R: Masters in Data Sciences 503, S22 First Batch, SMS, TU, 2021

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.

Outline: Supervised Learning with Regression Models (continued)

- Polynomial Regression
    - Model Accuracy
    - Model Prediction
    - Model Validation
- K Nearest neighbor
    - Model Accuracy
    - Model Prediction
    - Model Validation
- Neural Networks (NN) or Artificial Neural Networks (ANN)
    - Feed forward
    - Feedback
    - Model Accuracy
    - Model Prediction
    - Model Validation

Polynomial regression:

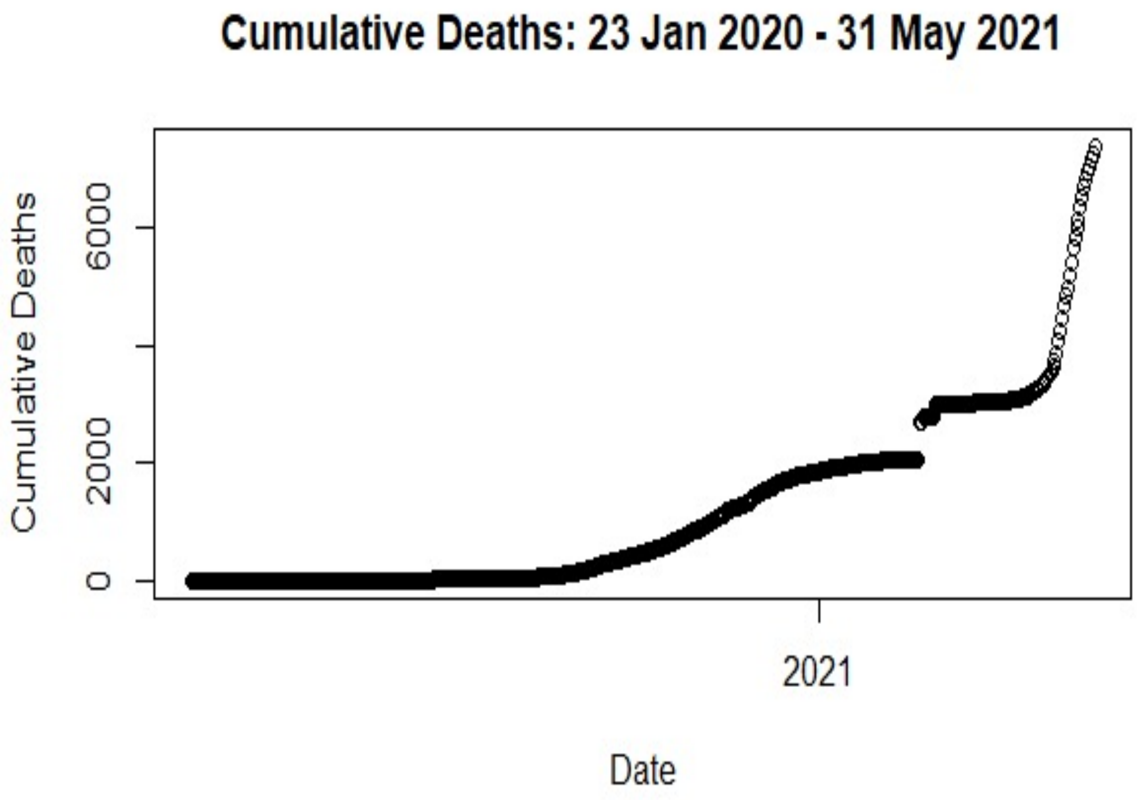
- Also known as curve-linear regression or curve fitting
- It is used when scatterplot shows non-linear relationship
- Most suitable for time series data but can be used for other situations too

- It is quite common to use:
- Quadratic linear model
  - $y = a + b_1x + b_2x^2$
  - x = t = for time for time series data
- Cubic linear model
  - $y = a + b_1x + b_2x^2 + b_3x^3$
  - x = t = time for time series data

We can use/check higher models polynomial models, if required

Let’s use the Nepal Covid data from Wikipedia and fit a polynomial models on Covid deaths:

- The cleaned covid_tbl_final.xlxs file contains COVID-19 cases from 23 Jan – 31 May 2021
- Import this file in R Studio
- Check the structure of the data
- Get scatterplot of total deaths and date variable

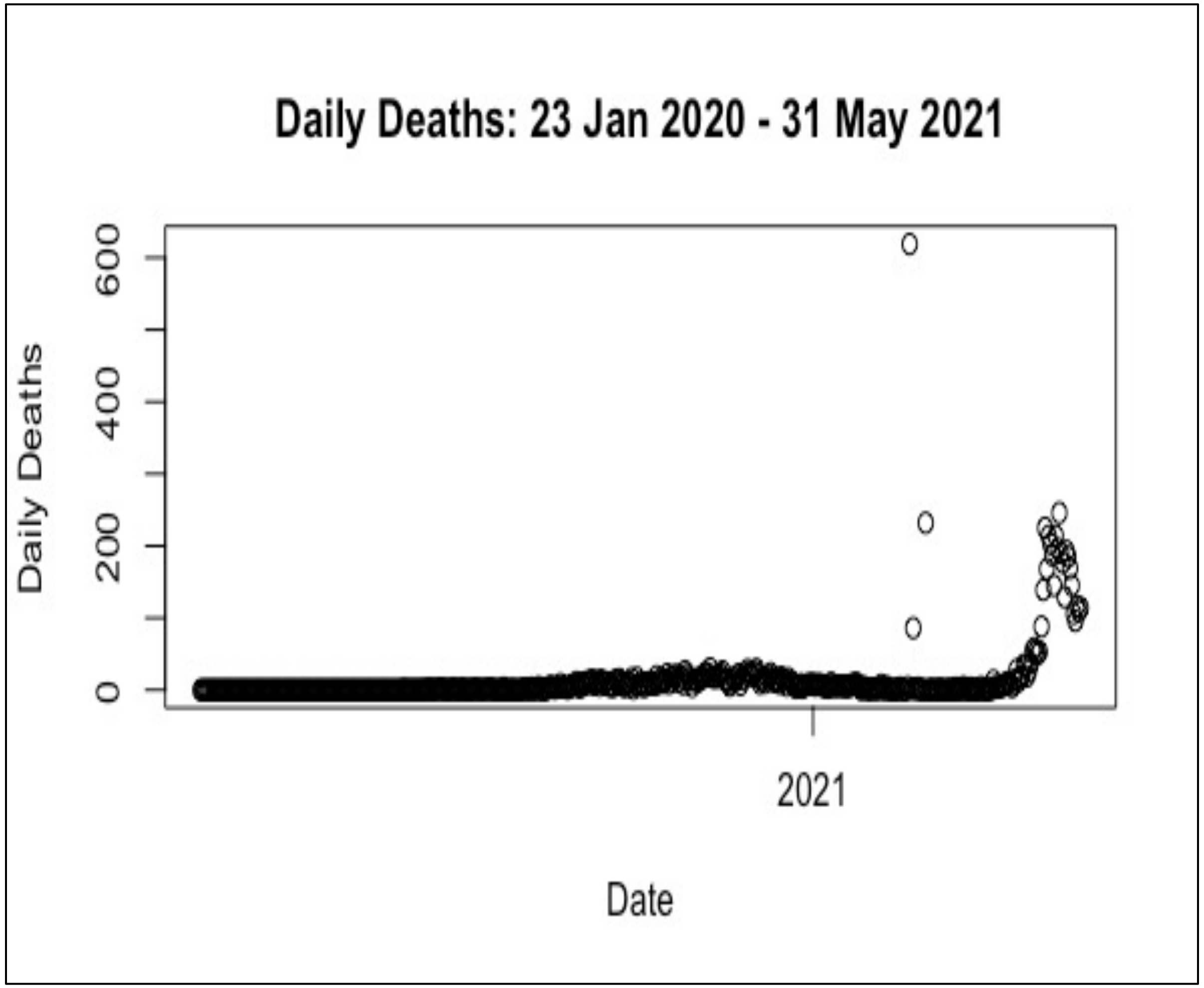


Let us plot the daily deaths by date and see what is causing the problem:

- #Daily deaths

```
plot(covid_tbl_final$Date, covid_tbl_final$Deaths_daily,
    main = "Daily Deaths: 23 Jan 2020 - 31 May 2021",
    xlab = "Date",
    ylab = "Daily Deaths")
```

#The problem is associated with the three outliers (all the missed deaths *a priori* added to the data on those 3 days!)



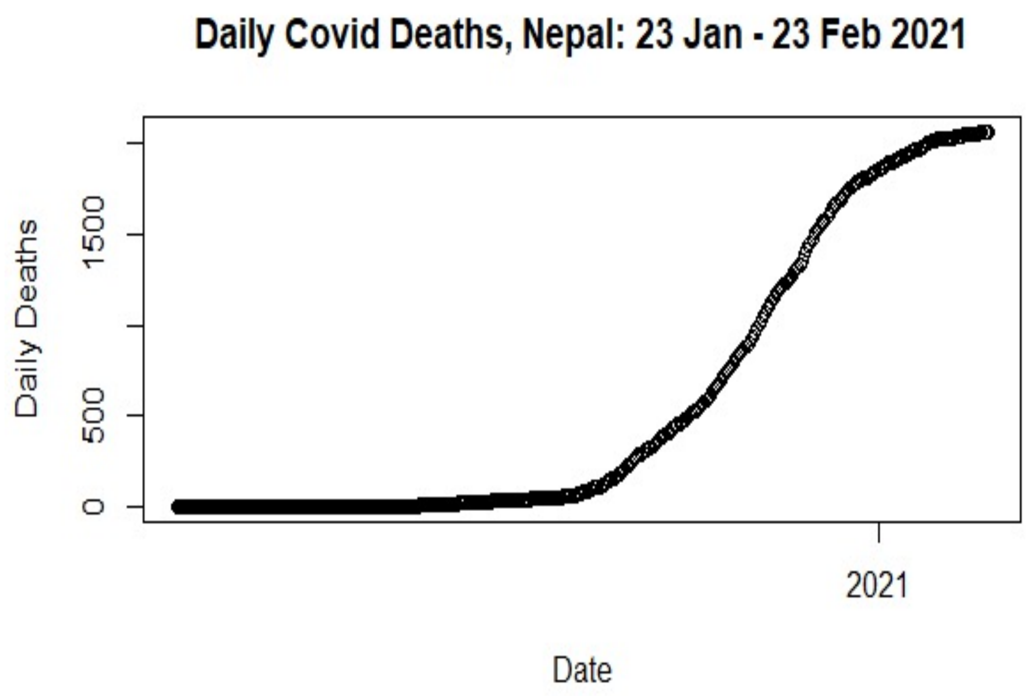
Let us plot the cumulative deaths again before these outliers i.e. till 23 Feb 2021

#Cumulative deaths upto 398 cases i.e. 23 Feb 2021

```
plot.data <- covid_tbl_final[covid_tbl_final$SN <=398,]
```

#Plot with filtered data

```
plot(plot.data$Date, plot.data$Deaths_total,
    main = "Daily Covid Deaths, Nepal: 23 Jan - 23 Feb 2021",
    xlab = "Date",
    ylab = "Daily Deaths")
```

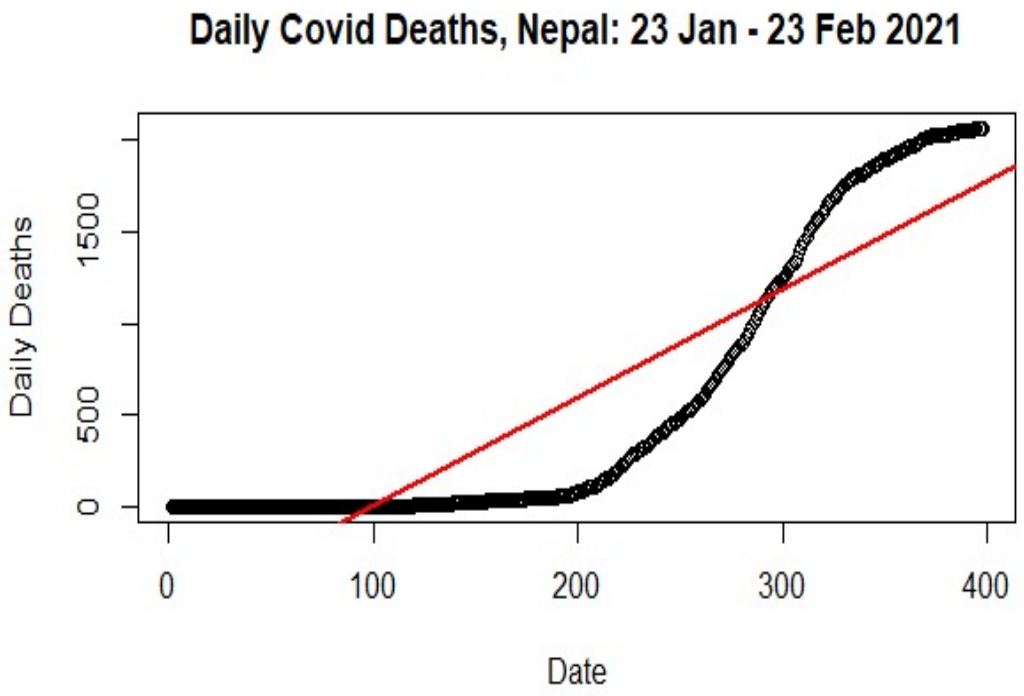


We will use this data to fit polynomial/curvelinear regression models now!

Let us fit a linear model in the filtered data (plot.data) using SN as time variable:

```
#Linear model:
lm <- lm(Deaths_total ~ SN, data = plot.data)
summary(lm)


#Plot with linear model
plot(SN, Deaths_total, data = plot.data,
    main = "Daily Covid Deaths, Nepal: 23 Jan - 23 Feb 2021",
    xlab = "Date",
    ylab = "Daily Deaths")
abline(lm(Deaths_total ~ SN, data = plot.data), col = "red", lwd=2)
```



What is the value of R-squared: ???
What is the value of Regression standard error: ???

Let us fit a quadratic linear model in the filtered data (plot.data):

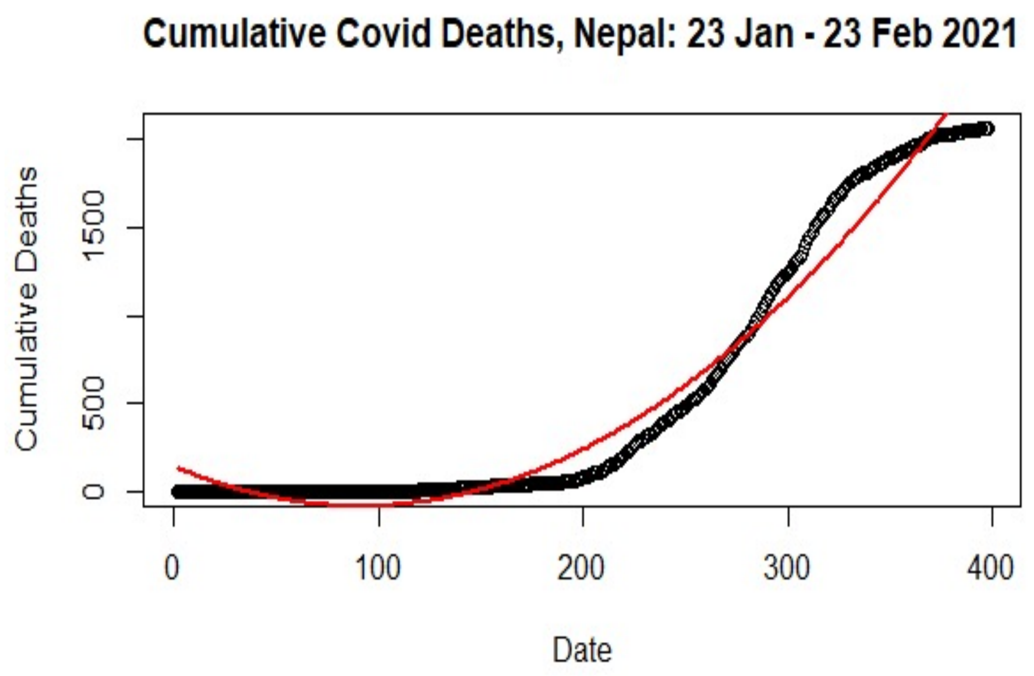
```
#Quadratic Linear model
qlm <- lm(Deaths_total ~ SN + I(SN^2), data = plot.data)
summary(qlm)
```

OR

qlm <- lm(Deaths_total ~ poly(SN, 2, raw=T), data = plot.data)

summary(qlm)

What is the difference? https://datascienceplus.com/fitting-polynomial-regression-r/




```
#Plot with quadratic linear model
plot(Deaths_total ~ SN, data=plot.data,
    main = "Cumulative Covid Deaths, Nepal: 23 Jan - 23 Feb 2021",
    xlab = "Date",
    ylab = "Cumulative Deaths")
lines(fitted(qlm) ~ SN, data=plot.data, col="red", lwd=2)
```

Let us fit a cubic linear model in the filtered data (plot.data):

#Cubic Linear model

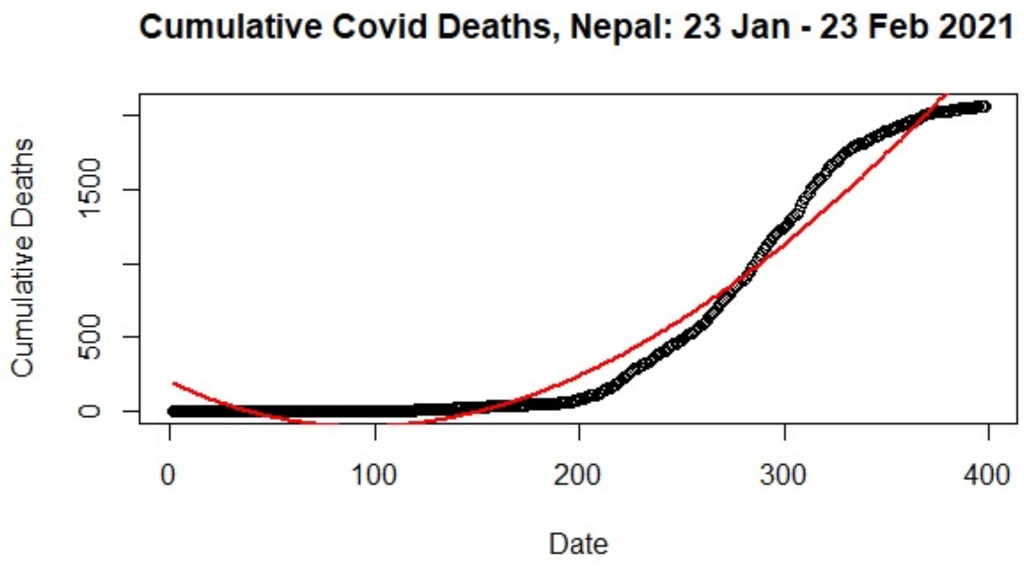
clm <- lm(Deaths_total ~ SN + I(SN^2) + I(SN^3, data = plot.data)

summary(clm)

OR

qlm <- lm(Deaths_total ~ poly(SN, 3, raw=T), data = plot.data)

summary(clm)



We can fit higher polynomials as long as we get the statistically significant F-test and coefficients (BLUE estimates).

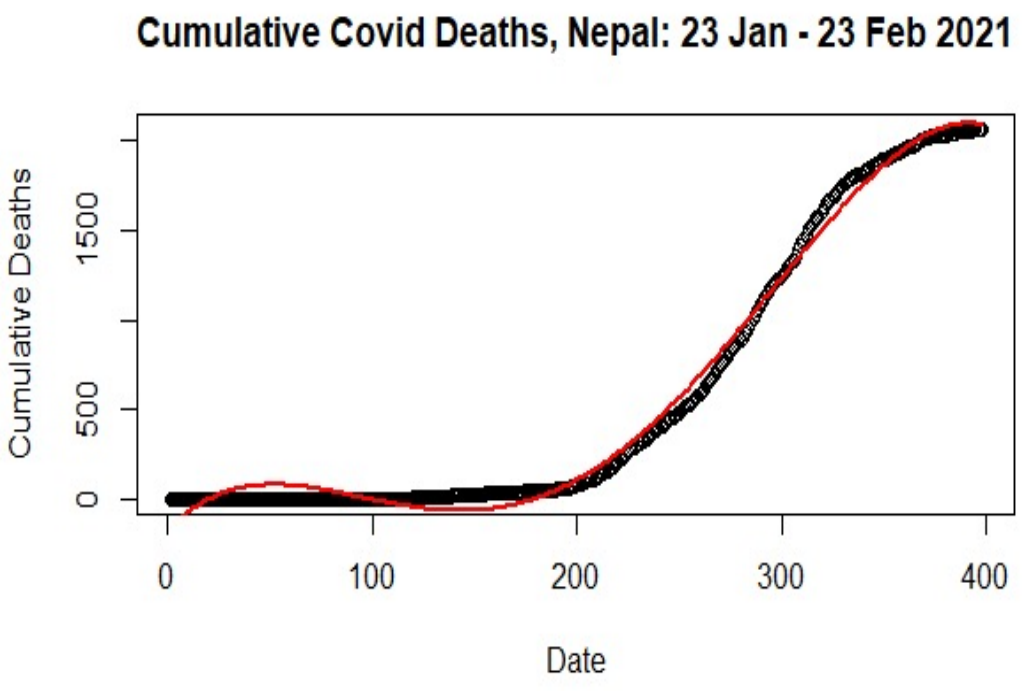
However, we need to ask **“Does it make sense?”** See how the R-square and Residual standard error changes with higher polynomials!

Fit a double quadratic linear model in the filtered data (plot.data):

#Double quadratic linear model

```
dqlm <- lm(Deaths_total ~ poly(SN, 4, raw=T), data = plot.data)

summary(dqlm)
```



This fitted line looks more closer to the observed data! **The question is: Is this "overfitting"?**

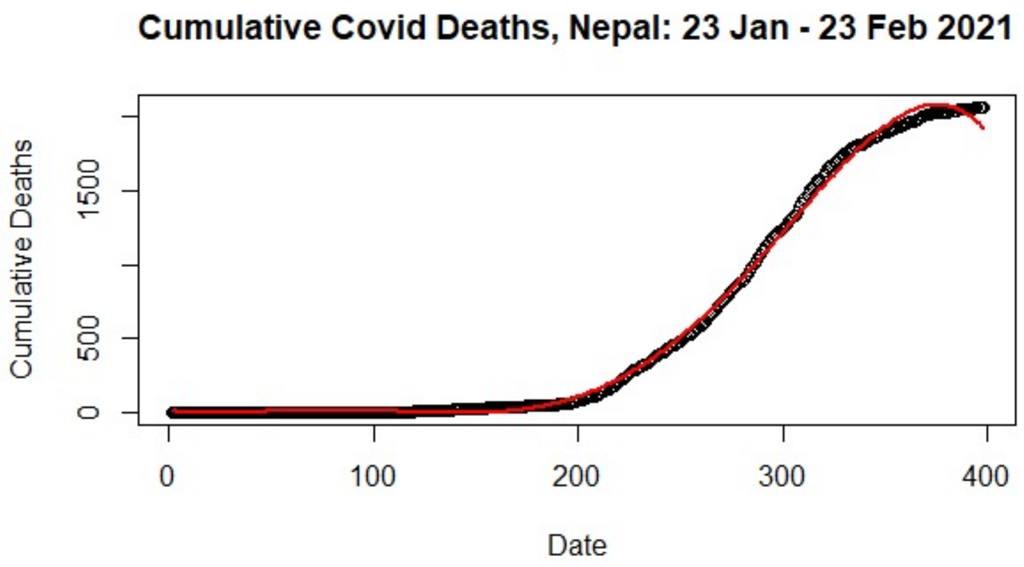
Fit a fifth order polynomial model in the filtered data (plot.data):

#Fifth order polynomial fit

folm <- lm(Deaths_total ~ poly(SN, 5, raw=T), data = plot.data)

summary(folm)

Higher polynomial fit will give higher R-squared and lower residual standard error for decreasing cumulative deaths!



This fitted line looks closer to the observed data! It suggests that cumulative deaths started to "decrease" around 395 days in Nepal!

Which linear/curve-linear model to choose?
Make sure to avoid GIGO here!

- R-squared
- Linear (1st order polynomial): 0.7921 (↑ deaths)
- Quadratic: 0.9692 (↑ deaths)
- Cubic: 0.9699 (↑ deaths)
- Double quadratic: 0.9934 (↔ deaths continue as usual)
- Fifth order polynomial: 0.9980 (↓ deaths)

- Residual standard error
- Linear (1st order polynomial): 350 (↑ deaths)
- Quadratic: 134.9 (↑ deaths)
- Cubic: 133.6 (↑ deaths)
- Double quadratic: 62.45 (↔ deaths continue as usual)
- Fifth order polynomial: 34.24 (↓ deaths)

How to translate polynomial regressions to the “data science” work/projects?

- Data partition
  - Train data (70% or 80%)
  - Test data (30% or 20%)
- Prediction on test data after fitting the best model in the train data (double quadratic?)
- Validation with R-squared and RMSE for test data

- You need to do this on your own
- **I will create an assignment for this in the MS Teams!**
- **I will add more models (KNN and ANN) to this assignment so that you can compare them with the polynomial regression models too!**

Is there other better way to deal with the time series data?

- **Yes. Time Series Decompositions.**
- We can use “moving average” methods: 3-day, 5-day, 7-day etc.
- We can use classical additive or multiplicative “decomposition” models and then forecast based on the adjusted trend
- **We can also use X-11, SEATS, STL decomposition methods!**

- We can use exponential smoothing models as polynomial regression, moving average and decomposition methods have limitations
- We can also use Autoregressive Integrated Moving Average (ARIMA) models as exponential models don’t use stationary data and can’t control the autocorrelations and the model residuals, which ARIMA can!

More here: https://otexts.com/fpp2/decomposition.html

Not part of this course but there is a separate course to learn this in the TU MDS curriculum, if you choose it!

Do we have a simple supervised learning algorithm for learning data science?

- Yes. It is "K nearest neighbor - KNN"!
- The KNN algorithm assumes that similar things exist in **close proximity** (near to each other). It normally uses Euclidean distance!

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- More on KNN algorithm is here: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

- K nearest neighbor regression
- Part of the "caret" package
- We need to load caret package to use it!
- library(caret)
- We will use Boston data of MASS package

More on KNN distance metrics: https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

Let’s do it: https://www.datatechnotes.com/2020/10/knn-regresion-example-in-r.html

#Define the data

boston = MASS::Boston

#Check the structure

str(boston)

#Data partition

set.seed(123)

ind <- sample(2, nrow(boston), replace = T, prob = c(0.8, 0.2))

train.data <- boston[ind==1,]

test.data <- boston[ind==2,]

#Training data scaling

train_x = train.data[, -14]

train_x = scale(train_x)[,]

train_y = train.data[,14]

#Test (validation) data scaling

test_x = test.data[, -14]

test_x = scale(test.data[,-14])[,]

test_y = test.data[,14]

KNN regression model for boston data:

#KNN regression, structure and prdiction

knnmodel = knnreg(train_x, train_y)

str(knnmodel)

Why K = 5? (Selected automatically!)

If we do it manually then we can start with the K that is close to number of features (variables)/3 i.e. 15/3 = 5 for regression based supervised learning and square root of number of variables for classification based supervised learning!

pred_y = predict(knnmodel, data.frame(test_x))

> str(knnmodel)

```
List of 3
 $ learn  :List of 2
  ..$ y: num [1:412] 24 21.6 34.7 28.7 22.9 16.5 18.9 18.9 21.7 20.4 ...
  ..$ X: num [1:412, 1:13] -0.414 -0.411 -0.411 -0.411 -0.405 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:412] "1" "2" "3" "6" ...
  .. .. ..$ : chr [1:13] "crim" "zn" "indus" "chas" ...
 $ k      : num 5
 $ theDots: list()
 - attr(*, "class")= chr "knnreg"
```

>summary(knnmodel) ???

	Length	Class	Mode
learn	2	-none-	list
k	1	-none-	numeric
theDots	0	-none-	list

KNN regression model prediction errors:

#Print the accuracy indices

```
print(data.frame(test_y, pred_y))

mse = mean((test_y - pred_y)^2)
mae = caret::MAE(test_y, pred_y)
rmse = caret::RMSE(test_y, pred_y)

cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)
```

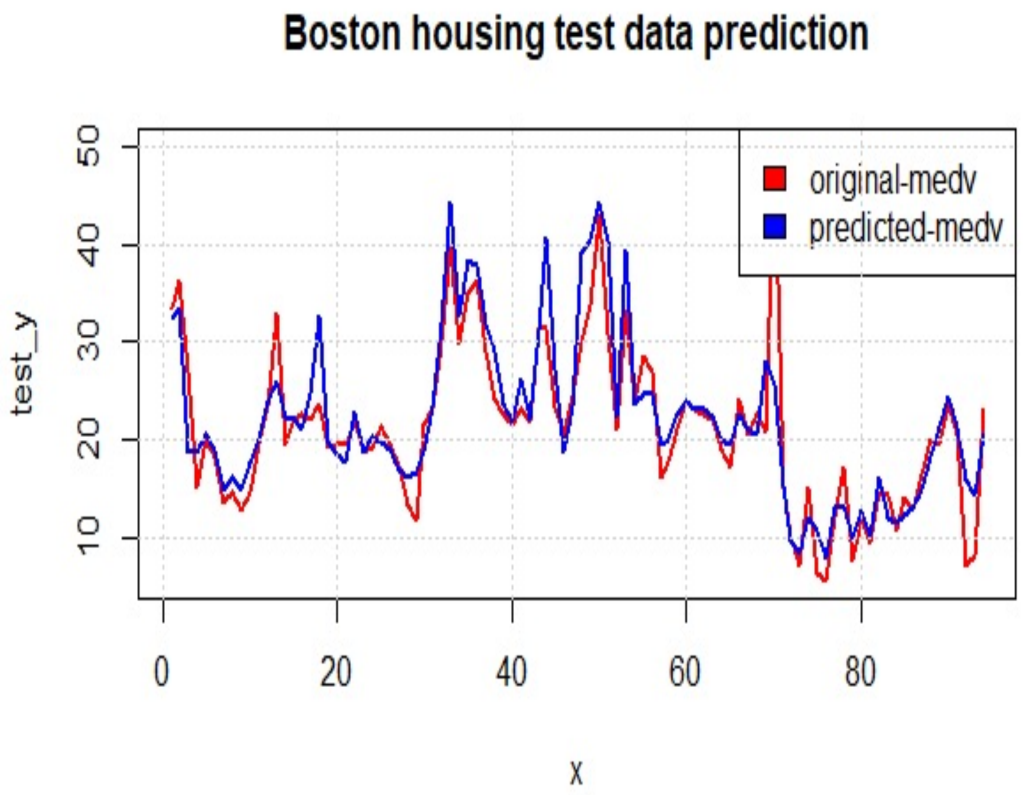
#Output in R:

- MSE: 17.57582
- MAE: 2.65617
- RMSE: 4.192353
- **R-squared: SSR/TSS = SSR/(SSR+SSE) = ??? (ANOVA???)**

- **Use KNN regression model for the Nepal COVID-19 data and compare it with polynomial regression model accuracy indices and take decision for the Assignment!**

KNN regression model validation plot:

```
#Plot
x = 1:length(test_y)
plot(x, test_y, col = "red", type = "l", lwd=2,
    main = "Boston housing test data prediction")
lines(x, pred_y, col = "blue", lwd=2)
legend("topright",  legend = c("original-medv", "predicted-medv"),
      fill = c("red", "blue"), col = 2:3,  adj = c(0, 0.6))
grid()
```



We can also fit:

- Decision Tree regression
  - https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda
- Support Vector Machine regression
  - https://koalatea.io/r-svm-regression/
- **Neural Network (NN) regression**
  - https://www.geeksforgeeks.org/how-neural-networks-are-used-for-regression-in-r-programming/?ref=rp

I strongly suggest to start with: ANN-MLP and ANN-RBF NN models for Supervised Learning in R!

- And chose the model with the lowest "error"
- This is what we do in data science!
- **Fit them on your own for the boston data and take decision!**
- **Let us see the Artificial Neural Network with the Nepal COVID-19 data and use it to compare other models used in the assignment 1**

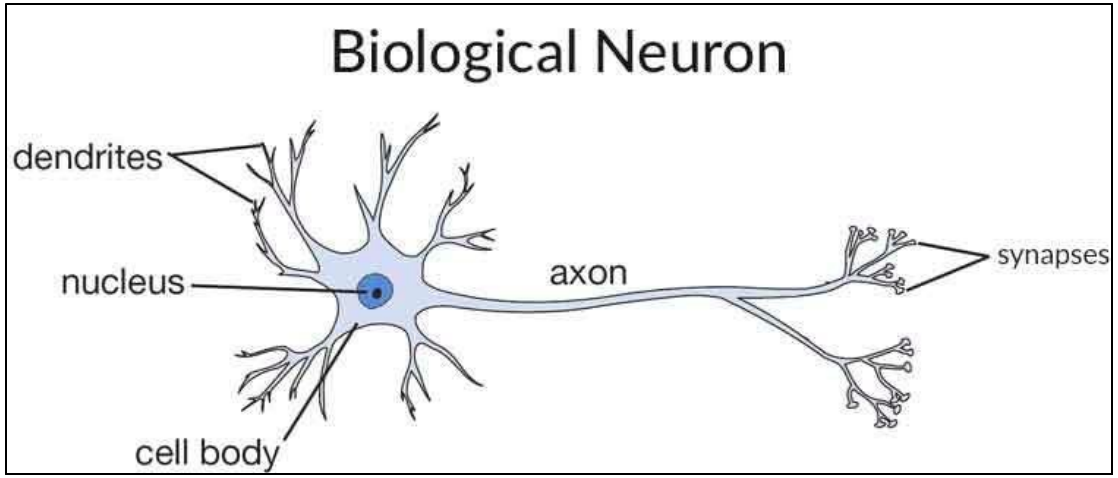
Single layer, feed-forward neural network:

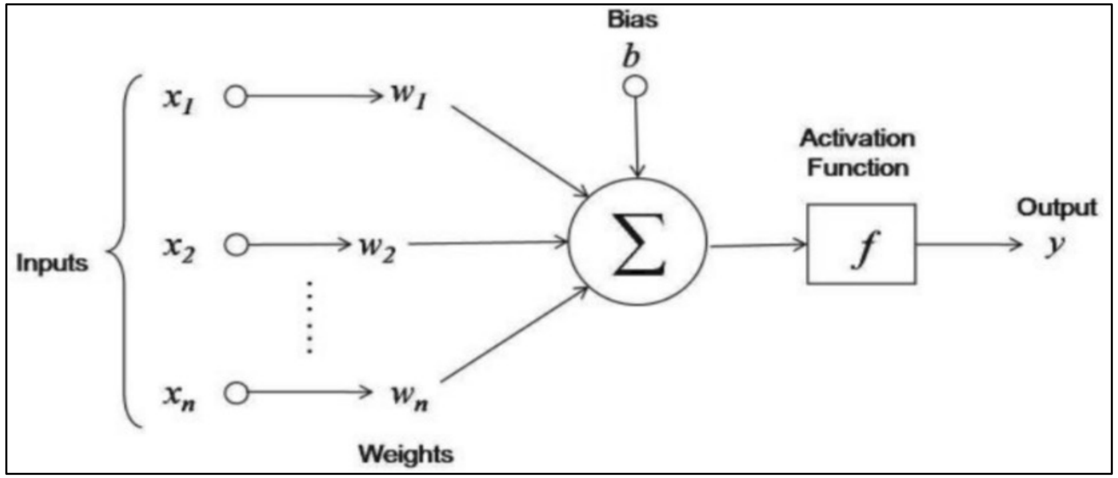
https://www.datacamp.com/community/tutorials/neural-network-models-r

- Neural Network (or Artificial Neural Network - ANN) has the ability to learn by examples.

- ANN is an information processing model inspired by the biological neuron system.

- Y = ∑ (weight * input) + bias
  - Weight = Synaptic weight

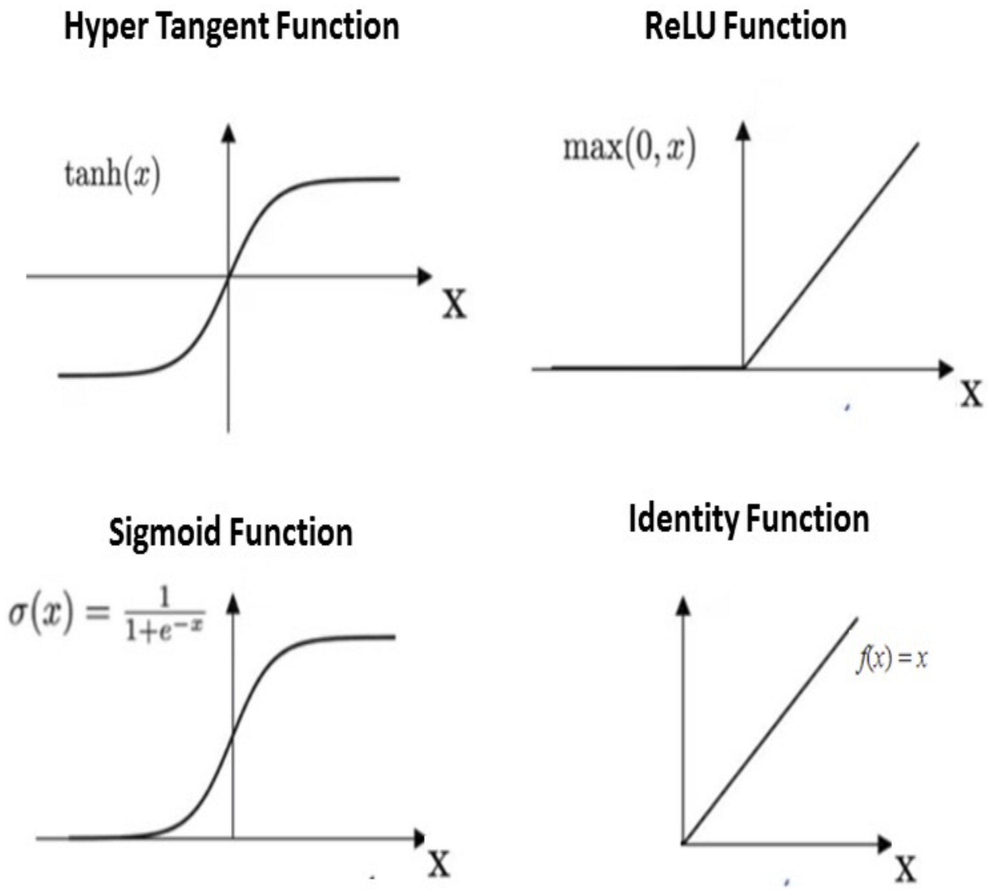




Activation functions:

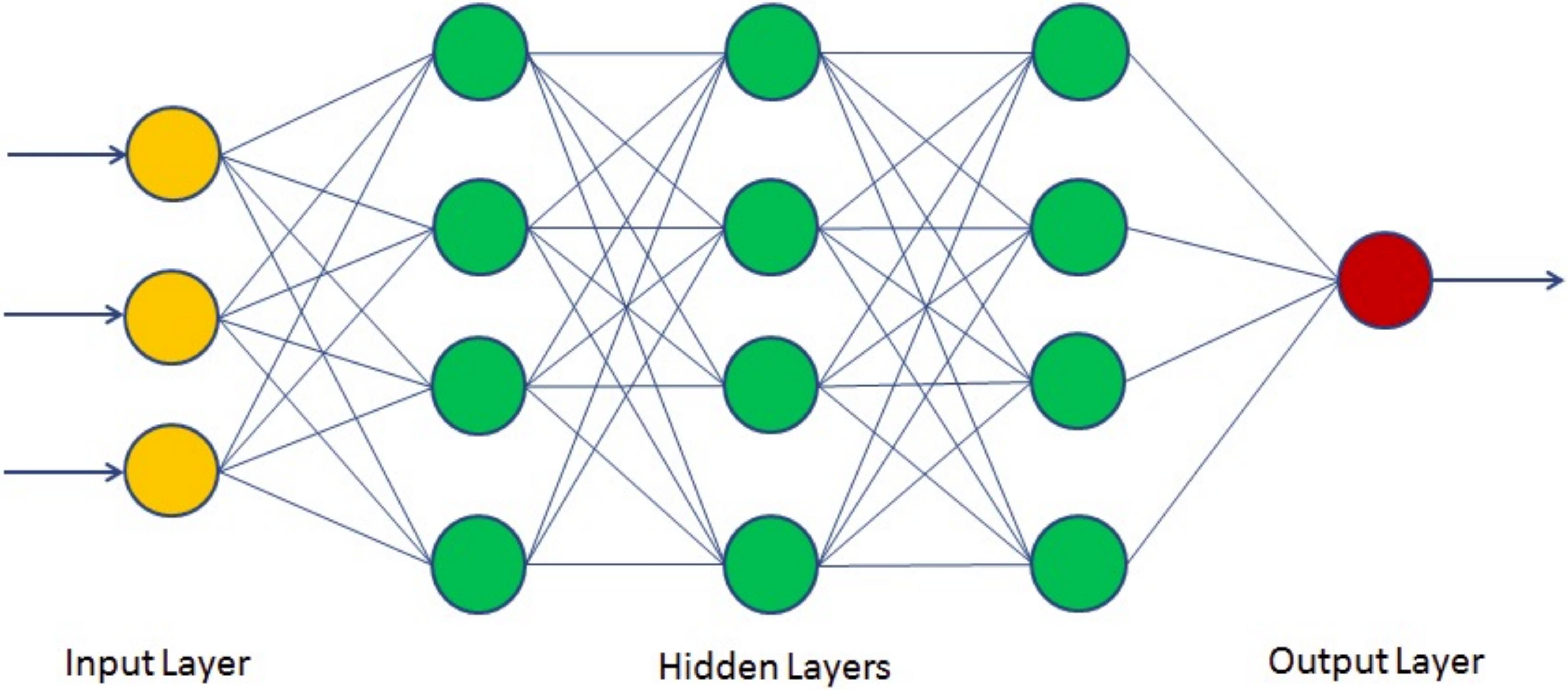
https://www.datacamp.com/community/tutorials/neural-network-models-r

- Identity function
  - Linear operator in vector space
- Binary step function
  - True/False: Classifier
- Sigmoid function
  - Binary sigmoid function (0 to 1)
  - Bipolar Sigmoid (Hyperbolic Tangent) function (-1 to +1)
- Ramp function
  - -ve to 0 and +ve to same output
- ReLu function
  - Rectified linear unit: 0 for negative values of x



Neural Network Model Example:

https://www.datacamp.com/community/tutorials/neural-network-models-r



Type of ANN: Feed forward & Feed backward

https://www.datacamp.com/community/tutorials/neural-network-models-r

- Feedforward neural network is a network which is not recursive.
- Neurons in this layer were only connected to neurons in the next layer, and they are don't form a cycle.
- In Feedforward signals travel in only one direction towards the output layer.

- Feedback neural networks contain cycles.
- Signals travel in both directions by introducing loops in the network.
- The feedback cycles can cause the network's behavior change over time based on its input.
- Feedback neural network also known as recurrent neural networks.

Check Part 1 and 2: https://rviews.rstudio.com/2020/07/20/shallow-neural-net-from-scratch-using-r-part-1/

Single “hidden” layer for Y ~ X (1 hidden layer)

#Library neuralnet

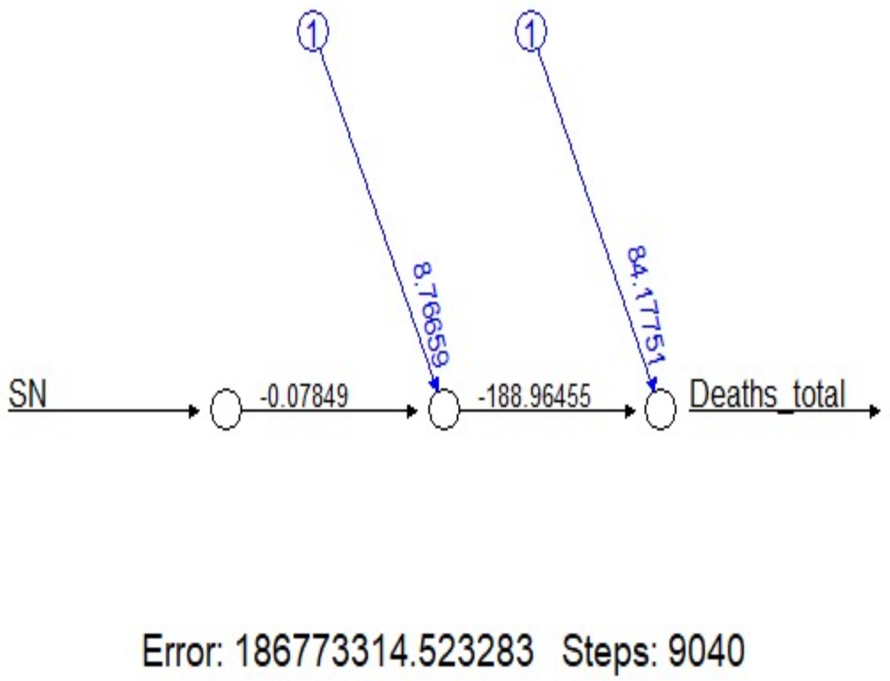
- library(neuralnet)

#NN Model

n <- neuralnet(Deaths_total ~ SN,
data=plot.data,
hidden = 1,
linear.output=F)

- plot(n)

#Linear.output = F means the relationship between Y and X is non-linear!



Single "hidden" layer for Y ~ X (2 hidden layers with 3 and 2 neurons)

#Library neuralnet

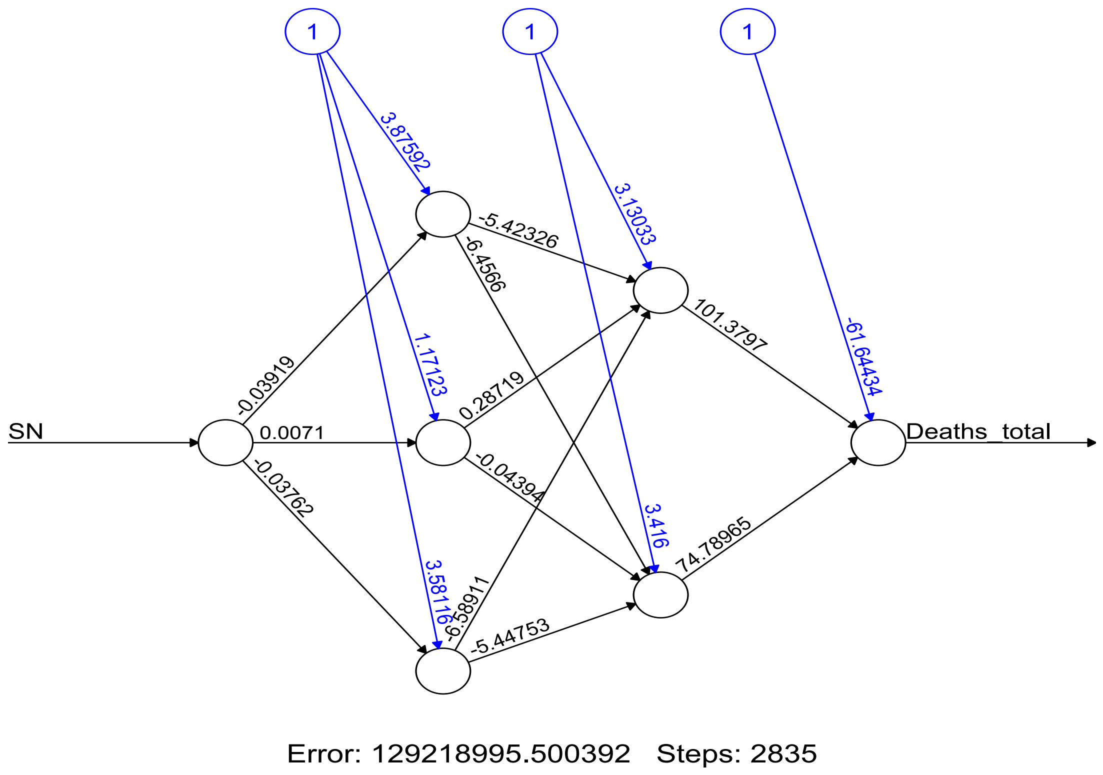
- library(neuralnet)

#NN model

```
n <- neuralnet(Deaths_total ~ SN,
            data=plot.data,
            hidden = c(3,2),
            linear.output=F)
```

##NN Plot

- plot(n)



Using Covid-19 Nepal data from Wikipedia to fit MLP with data science approach:

```
#Data Partition

ind <- sample(2, nrow(plot.data), replace = T, prob = c(0.7,0.3))

trainset <- plot.data[ind==1,]

testset <- plot.data[ind==2,]
```

```
#Neural Network

library(neuralnet)

#NN model

nn <- neuralnet(Deaths_total ~ SN, data=trainset, hidden=c(3,1),
                linear.output=FALSE, threshold=0.01)

#Plot the NN model

plot(nn)
```

More here: https://datascienceplus.com/neuralnet-train-and-test-neural-networks-using-r/

Multilayer perceptron: Y ~ X (2 hidden layers with 3 and 1 neurons i.e. c(3,1))

#Test the resulting output

temp_test <- subset(testset, select = c(“SN”))

head(temp_test)

#Prediction using compute for NN model with neuralnet package!

nn.results <- compute(nn, temp_test)

#Model validation

results <- data.frame(actual = testset$SN, prediction = nn.results$net.result)

results

#Model Accuracy

deviation=((results$actual-results$prediction)/results$actual)

(accuracy=abs(mean(deviation)))

(error=1-accuracy)

MSE = ???,   RMSE = ???

Question/queries?

- Next class
- Classification models for supervised learning
  - Logistic regression
  - Naïve Bayes
  - Support Vector Machine
  - Decision Trees etc.
- Data partition, model fit on training data, prediction and validation on test data

- Last class on Supervised learning
- Ensemble learning
  - Bagging and Boosting
  - Random Forests

Thank you!

@shitalbhandary