Machine Learning Final Project

By: Rambabu Talacheru

Introduction: The effectiveness of cancer prediction system helps the people to know their cancer risk with low cost and it also helps the people to take the appropriate decision based on their cancer risk status. The data is collected from the website online lung cancer prediction system and taken by Kaggle.

We build a machine learning model using this data which effectively predicts the risk of each person with the given attributes getting diagnosed with lung cancer.

What does the data tell?

We have the data which shows the survey results of whether people with certain symptoms (Like Chest pain, Shortness of breath, Fatigue etc.,) and habits (Like Smoking, Alcohol Consumption etc.,) are diagnosed with lung cancer taken from Kaggle.

Features of the data:

No. of instances = 284, Total no. of attributes = 16, Classes = 2

Attributes with their respective column numbers:

1. Gender: M(male), F(female)
2. Age
3. Smoking
4. Yellow fingers

5. Anxiety
6. Peer_pressure
7. Chronic Disease
8. Fatigue
9. Allergy
10. Wheezing
11. Alcohol
12. Coughing
13. Shortness of Breath
14. Swallowing Difficulty
15. Chest pain
16. Lung Cancer (Response variable: Yes or No)

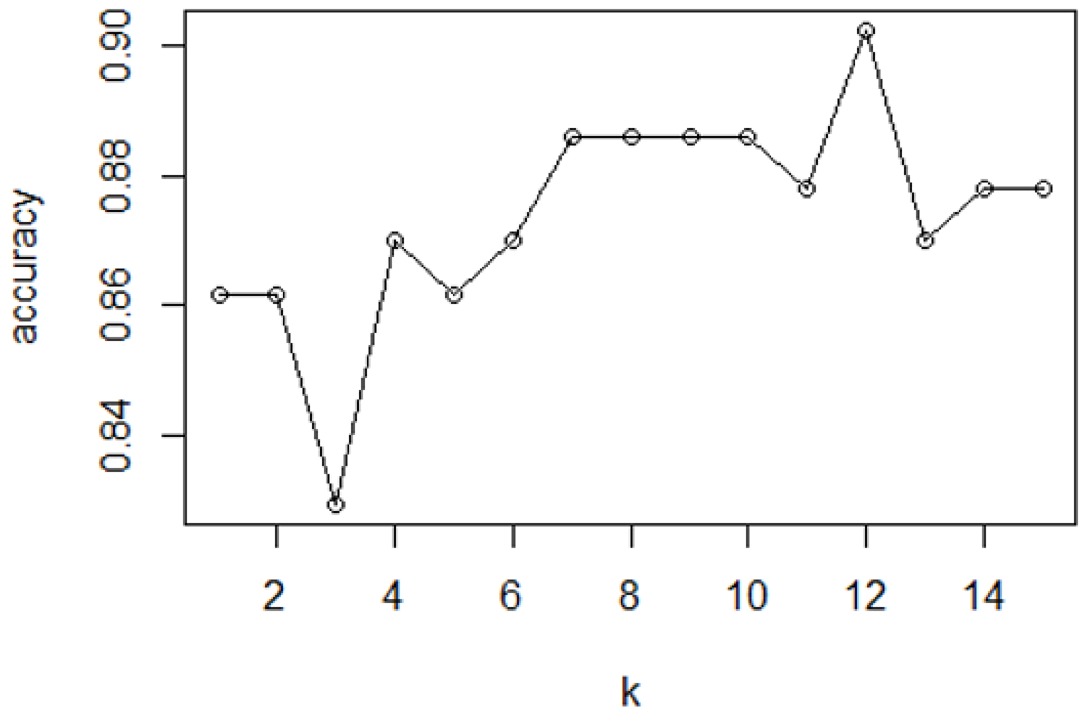
Objective: To build a model based on the given data that predicts whether the person with attributes mentioned above will be diagnosed with lung cancer using K-nearest neighbors algorithm.

Why Knn is chosen to solve the problem? Knn is one of the algorithms which gives the results with high accuracy and the given data best fits into this model as this model calculates the distance (Euclidean) of between each variable of test data with its respective variable in the training data.

Model:

The data is split into Training (50%), Validation(30%) and Test(20%) sets and Knn algorithm is applied on the data.

The value of K where the accuracy is highest (90.24%) on the validation set (Best K) is 12.



Conclusion:

When the model is applied on the Test data with k=12, it is found that the prediction is 91.94% accurate.

So, based on the given data, we are able to build a model using K-nearest neighbors algorithm which effectively predicts (With an accuracy of 91.94%) whether any person with the given attributes is likely to be diagnosed with lung cancer or not.

Confusion Matrix and Statistics:

		Predicted condition	
	Total 57 + 5 = 62	Cancer	Non-cancer
Actual condition	Cancer	53	4
	Non-cancer	1	4

Accuracy = 0.9194.

There is a possibility that we arrive at a misleading information only if the Accuracy is considered.

We can tell that the model is giving better results only after considering certain other values calculated using confusion matrix as below.

Precision or **Positive predicted value** = 0.9815

Sensitivity or **true positive rate** = 0.9298

Specificity or **True negative rate** = 0.80

From the above figures, it can be inferred that the model is effective in predicting the people who are prone to Lung-Cancer based on the given data.

Below is the code which is used to execute the program in 'R' software:

Loading the required libraries

```
library(class)

## Warning: package 'class' was built under R version 4.1.2

library(caret)

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Loading required package: lattice

library(ISLR)

## Warning: package 'ISLR' was built under R version 4.1.2
```

Read the required data

```
survey.lung.cancer <- read.csv("C:/Users/ramne/Desktop/ML Assignment/Final
Project/survey lung cancer.csv")
View(survey.lung.cancer)
```

Creating the dummy variables for Column ‘Gender’.

```
library(psych)

## Warning: package 'psych' was built under R version 4.1.2

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

dummy_Data <- as.data.frame(dummy.code(survey.lung.cancer$GENDER))
names(dummy_Data) <- c("GENDER_1", "GENDER_2")
```

Removing the original Column ‘Gender’ as we have new dummy variables ‘GENDER_1’ and ‘GENDER_2’

```
Data_without_Gender = subset(survey.lung.cancer,select = - c(GENDER))
New_data <- cbind(Data_without_Gender,dummy_Data)
New_data$LUNG_CANCER <- as.factor(New_data$LUNG_CANCER)
```

Data is split into training(60%) and validation(40%) data

```
set.seed(123)
train_index <- createDataPartition(New_data$LUNG_CANCER,p=0.6,list = FALSE)
```

```
train_data <- New_data[train_index,]
validation_data <- New_data[-train_index,]
```

Normalize the training data using preProcess function.

Apply the normalized model on the training and validation data using Predict Function.

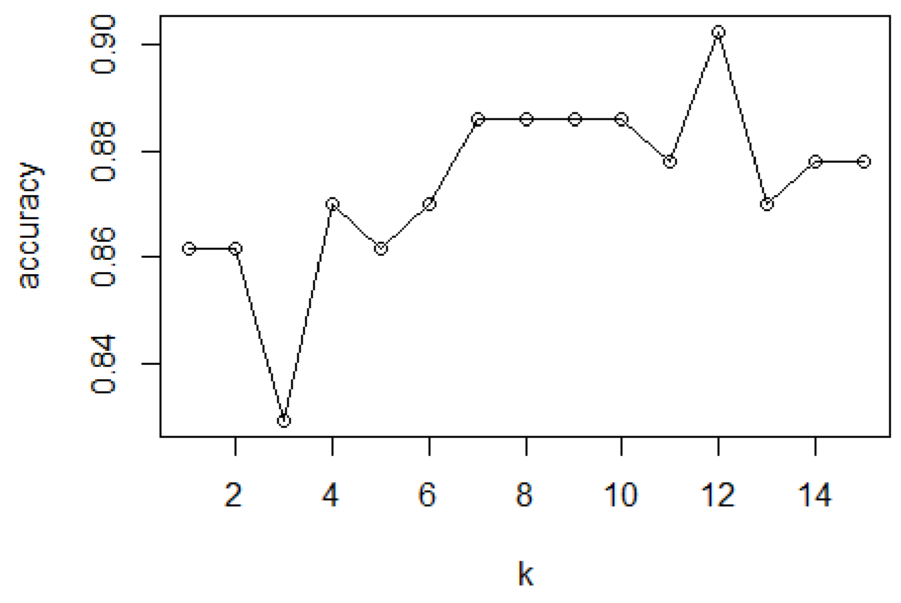
```
train.norm.df <- train_data
valid.norm.df <- validation_data
traval.norm.df <- New_data #(Training + Validation data)
#Use preProcess() function to normalize numerical columns from the dataset
Values_z_normalised <- preProcess(train_data[,-15],method =
c("center","scale"))
#Applying the normalized model on the training, validation and test data
train.norm.df[,-15] <-  predict(Values_z_normalised,train_data[,-15])
valid.norm.df[,-15] <-  predict(Values_z_normalised,validation_data[,-15])
traval.norm.df[,-15] <-  predict(Values_z_normalised,New_data[,-15])
```

Determine the best K

```
library(caret)
set.seed(123)
accuracy.df <- data.frame(k = seq(1, 15, 1), accuracy = rep(0, 15))
# compute knn for different k on validation.
for(i in 1:15) {
  knn.pred <- knn(train.norm.df[,-15], valid.norm.df[,-15],
                  cl = train.norm.df[, 15], k = i)
  accuracy.df[i, 2] <-
confusionMatrix(knn.pred,valid.norm.df[,15])$overall[1]
}
accuracy.df

##     k  accuracy
## 1   1 0.8617886
## 2   2 0.8617886
## 3   3 0.8292683
## 4   4 0.8699187
## 5   5 0.8617886
## 6   6 0.8699187
## 7   7 0.8861789
## 8   8 0.8861789
## 9   9 0.8861789
## 10 10 0.8861789
## 11 11 0.8780488
## 12 12 0.9024390
## 13 13 0.8699187
## 14 14 0.8780488
## 15 15 0.8780488

plot(accuracy.df,type="o")
```



From the above results, best K is 12.

Confusion Matrix for the best K(12).

```
Model.k.12 <- knn(train.norm.df[,-15], valid.norm.df[,-15],
                  cl = train.norm.df[,15], k = 12,prob = TRUE)
confusionMatrix(Model.k.12,valid.norm.df[,15])

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   1   2
##            1 105  10
##            2   3   5
##
##                Accuracy : 0.8943
##                  95% CI : (0.826, 0.9425)
##     No Information Rate : 0.878
##     P-Value [Acc > NIR] : 0.35062
##
##                   Kappa : 0.3824
##
##  Mcnemar's Test P-Value : 0.09609
##
##             Sensitivity : 0.9722
##             Specificity : 0.3333
##          Pos Pred Value : 0.9130
##          Neg Pred Value : 0.6250
##               Prevalence : 0.8780
##           Detection Rate : 0.8537
##     Detection Prevalence : 0.9350
##        Balanced Accuracy : 0.6528
##
##         'Positive' Class : 1
##
```

Splitting the data into Training (50%), Validation (30%) and Test (20%) sets

```
## Data Splitting (50% Training Data and 30% for validation data and 20% test
data)
set.seed(123)
str(New_data)

## 'data.frame':    309 obs. of  17 variables:
##  $ AGE                  : int  69 74 59 63 63 75 52 51 68 53 ...
##  $ SMOKING              : int  1 2 1 2 1 1 2 2 2 2 ...
##  $ YELLOW_FINGERS       : int  2 1 1 2 2 2 1 2 1 2 ...
##  $ ANXIETY              : int  2 1 1 2 1 1 1 2 2 2 ...
##  $ PEER_PRESSURE        : int  1 1 2 1 1 1 1 2 1 2 ...
##  $ CHRONIC.DISEASE      : int  1 2 1 1 1 2 1 1 1 2 ...
##  $ FATIGUE              : int  2 2 2 1 1 2 2 2 2 1 ...
##  $ ALLERGY              : int  1 2 1 1 1 2 1 2 1 2 ...
##  $ WHEEZING             : int  2 1 2 1 2 2 2 1 1 1 ...
##  $ ALCOHOL.CONSUMING    : int  2 1 1 2 1 1 2 1 1 2 ...
##  $ COUGHING             : int  2 1 2 1 2 2 2 1 1 1 ...
##  $ SHORTNESS.OF.BREATH  : int  2 2 2 1 2 2 2 2 1 1 ...
##  $ SWALLOWING.DIFFICULTY: int  2 2 1 2 1 1 1 2 1 2 ...
##  $ CHEST.PAIN           : int  2 2 2 2 1 1 2 1 1 2 ...
##  $ LUNG_CANCER          : Factor w/ 2 levels "1","2": 1 1 2 2 2 1 1 1 2 1
...
##  $ GENDER_1             : num  1 1 0 1 0 0 1 0 0 1 ...
##  $ GENDER_2             : num  0 0 1 0 1 1 0 1 1 0 ...

test_index1 <- createDataPartition(New_data$LUNG_CANCER,p=0.2,list = FALSE)
Test_Data2 <- New_data[test_index1,]#  (Test data)
train_vali_data <- New_data[-test_index1,]
train_index2 <- createDataPartition(train_vali_data$LUNG_CANCER,p=0.625,list
= FALSE)
train_data2 <- train_vali_data[train_index2,] # (Training data)
validation_data2 <- train_vali_data[-train_index2,]# (validation data)
 NROW(Test_Data2)

## [1] 62

 NROW(train_data2)

## [1] 155

 NROW(validation_data2)
```

```
## [1] 92
```

Applying the K-nn model for the data

```
# Renormalizing the (training+validation) data
set.seed(123)
Values_z_normalised2 <- preProcess(traval.norm.df[,-15], method =
c("center","scale"))
traval.norm.df[,-15] <-  predict(Values_z_normalised2,New_data[,-15])
 # Data Normalization

# Copy the original data
train.norm.df2 <- train_data2
valid.norm.df2 <- validation_data2
train_vali.norm.df <- train_vali_data
test.norm.df2 <-Test_Data2
#Use preProcess() function to normalize numerical columns from the New_data
dataset
Values_z_normalised_repartition <- preProcess(train_data2[,-15],method =
c("center","scale"))
train.norm.df2[,-15] <-
predict(Values_z_normalised_repartition,train_data2[,-15])
valid.norm.df2[,-15] <-
predict(Values_z_normalised_repartition,validation_data2[,-15])
train_vali.norm.df[,-15] <- predict(Values_z_normalised2,train_vali_data[,-
15])
test.norm.df2[,-15] <-predict(Values_z_normalised_repartition,Test_Data2[,-
15])

## Modeling k-NN for validation data
set.seed(123)
train_knn_12<- knn(train.norm.df2[,-15],train.norm.df2[,-
15],cl=train.norm.df2[,15],k=12,prob=TRUE)
valid_knn_12<- knn(train.norm.df2[,-15],valid.norm.df2[,-15],cl=
train.norm.df2[,15],k=12,prob= TRUE)
#print(ModelNew.k.1)
head(train_knn_12)

## [1] 1 1 1 1 1 2
## Levels: 1 2

head(valid_knn_12)

## [1] 1 1 1 1 1 1
## Levels: 1 2

actual= valid.norm.df2[,15]
mean(valid_knn_12==actual)

## [1] 0.8695652
```

```
class_prob = attr(valid_knn_12,"prob")
head(class_prob)

## [1] 0.9166667 0.7500000 0.8333333 0.5833333 0.7692308 0.5833333

# Knn for test data
Values_z_normalised3<- preProcess(train_vali_data[,-15], method =
c("center","scale"))
train_vali.norm.df[,-15] <- predict(Values_z_normalised3,train_vali_data[,-
15])
test.norm.df2[,-15]<- predict(Values_z_normalised3,Test_Data2[,-15])
test_knn_12<- knn(train_vali.norm.df[,-15],test.norm.df2[,-
15],cl=train_vali.norm.df[,15],k=12)
#print(Model_new3)
head(test_knn_12)

## [1] 1 1 1 1 1 1
## Levels: 1 2

actual= test.norm.df2[,15]
mean(test_knn_12==actual)

## [1] 0.9193548
```

Including Confusion Matrix

Accuracy of the Knn models for traning,validation and test datasets for k=12

```
confusionMatrix(train_knn_12,as.factor(train.norm.df2[,15]),positive = '1')

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 135  16
##          2   0   4
## 
##                Accuracy : 0.8968
##                  95% CI : (0.8378, 0.9398)
##     No Information Rate : 0.871
##     P-Value [Acc > NIR] : 0.2032536
## 
##                   Kappa : 0.3034
## 
##  Mcnemar's Test P-Value : 0.0001768
## 
##             Sensitivity : 1.0000
##             Specificity : 0.2000
##          Pos Pred Value : 0.8940
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8710
##          Detection Rate : 0.8710
##    Detection Prevalence : 0.9742
##       Balanced Accuracy : 0.6000
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(valid_knn_12,as.factor(valid.norm.df2[,15]),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 80 11
##          2  1  0
##
##                Accuracy : 0.8696
##                  95% CI : (0.7832, 0.9307)
##     No Information Rate : 0.8804
##     P-Value [Acc > NIR] : 0.696038
##
##                   Kappa : -0.0203
##
##  Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.9877
##             Specificity : 0.0000
##          Pos Pred Value : 0.8791
##          Neg Pred Value : 0.0000
##              Prevalence : 0.8804
##          Detection Rate : 0.8696
##    Detection Prevalence : 0.9891
##       Balanced Accuracy : 0.4938
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(test_knn_12,as.factor(test.norm.df2[,15]),positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 53  4
##          2  1  4
##
##                Accuracy : 0.9194
##                  95% CI : (0.8217, 0.9733)
##     No Information Rate : 0.871
##     P-Value [Acc > NIR] : 0.1725
##
##                   Kappa : 0.573
```

```
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9815
##             Specificity : 0.5000
##          Pos Pred Value : 0.9298
##          Neg Pred Value : 0.8000
##              Prevalence : 0.8710
##          Detection Rate : 0.8548
##    Detection Prevalence : 0.9194
##       Balanced Accuracy : 0.7407
##
##        'Positive' Class : 1
##
```