K-Nearest Neighbours

ROBERT CARULLA & ANKIT TEWARI

Introduction

In this session, we have an independent variable X and a dependent variable Y which we will try to predict using the regression based on K-Nearest Neighbour method. We define the regression function based on K-Nearest Neighbours in a similar way as we had previously studied based on least squares estimates. The dataset that we have used consist of X being the **lstat** and Y being the **medv** and our objective in the present scenario is to estmate the regression function of X using the Y.

```
library(MASS)
data("Boston")
X <- Boston$lstat
Y <- Boston$medv
mydf <- as.data.frame( cbind(X,Y) )
```

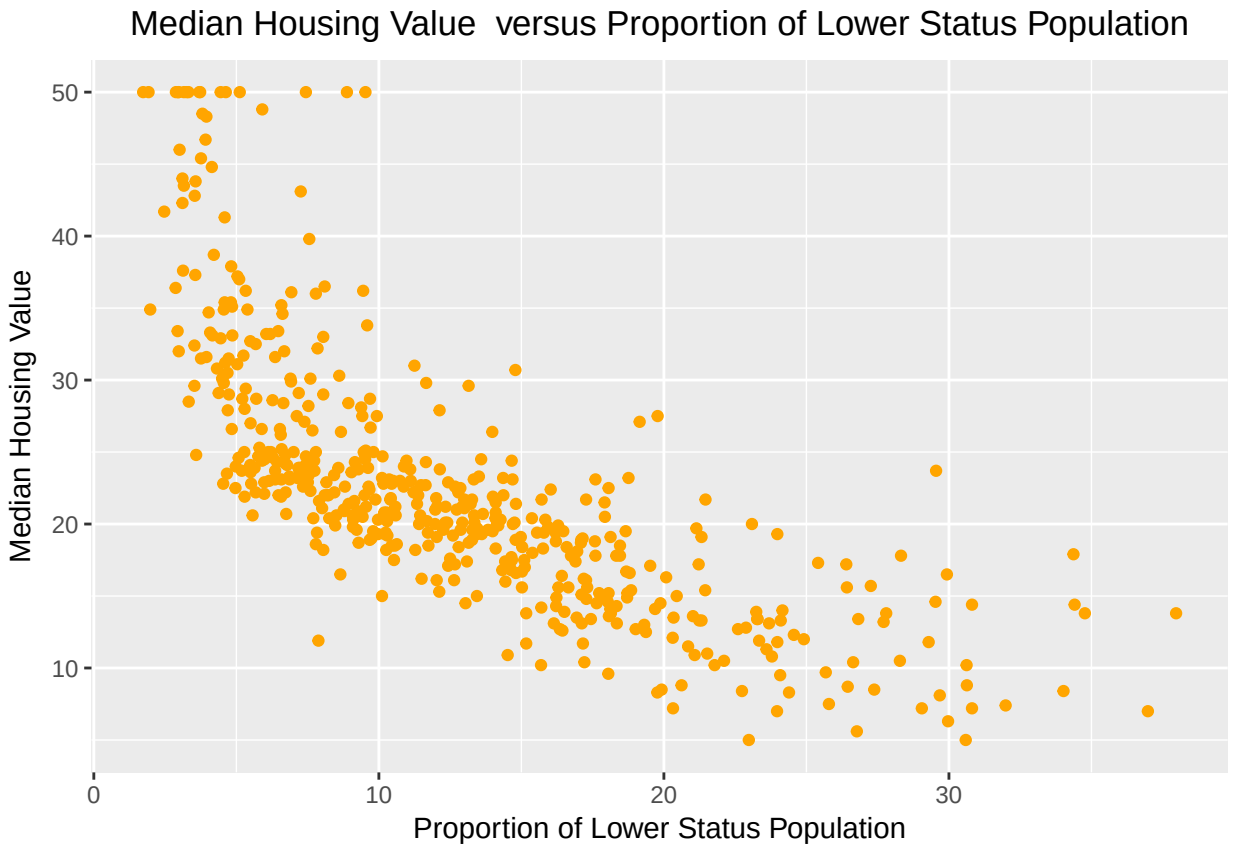
Let us see the structure of our dataset using the **str** fucntion-

```
str(mydf)
```

```
## 'data.frame':    506 obs. of  2 variables:
##  $ X: num  4.98 9.14 4.03 2.94 5.33 ...
##  $ Y: num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

An exploratory data analysis of the dataset using the scatterplot is described here below-

```
library(ggplot2)
theme_update(plot.title = element_text(hjust = 0.5))
g <- ggplot(data = mydf, aes(x= X, y= Y))
g + geom_point(color="orange") +
  ggtitle("Median Housing Value  versus Proportion of Lower Status Population")+
  labs(x= "Proportion of Lower Status Population", y= "Median Housing Value")
```



The scatterplot between X and Y suggests a possible relationship between X and Y and therefore we can conclude that Y varies with X as being some function of X (although precisely, a non linear relationship between X and Y sounds more obvious for e.g. the exponential decay of Y with respect to X).

Method

We define the K-nearest neighbour estimator $m(t) = E(Y|X = t)$ for t as

$$\hat{m}(t) = \frac{1}{k} \sum_{i \in N_k(t)} y_i$$

where $t \in \mathbb{R}$ can be a vector of numeric values. So, the following function of code implements the algorithm for the estimator $\hat{m}(t)$.

```
knn.reg <- function(klist,x.train,y.train,x.test) {
  x.train <- as.data.frame(x.train)
  x.test  <- as.data.frame(x.test)
  n.train <- nrow(as.data.frame(x.train))
  n.test <- nrow(as.data.frame(x.test))

  # Matrix to store predictions
  p.test <- matrix(NA, n.test, length(klist))

  # Vector to store the distances of a point to the training points
  dsq <- numeric(n.train)

  # Loop on the test instances
```

```
  for (tst in 1:n.test)
  {
    # Compute distances to training instances
    for (trn in 1:n.train)
    {
      dsq[trn] <- sum((x.train[trn,] - x.test[tst,])^2)
    }

    # Sort distances from smallest to largest
    ord <- order(dsq)

    # Make prediction by averaging the k nearest neighbors
    for (ik in 1:length(klist)) {
      p.test[tst,ik] <- mean(y.train[ord[1:klist[ik]]])
    }
  }

  # Return the matrix of predictions
  invisible(p.test)
}
```

So, we may define a sequence of values from 1 to 40 and store this sequence as a vector t such that $t = \{1, 2, ..., 40\}$. Now, using a value of $K = 50$, we will attempt to fit the nearest neighbour algoritm as a regression function for estimating $m(t)$ using the sequence t

```
t <- seq(40)
klist <- c(10,20,30,40,50,60,70)
predictions <- knn.reg(klist, X,Y,t)
#fitted <- knn.reg(klist, X,Y,X)
(predictions)
```

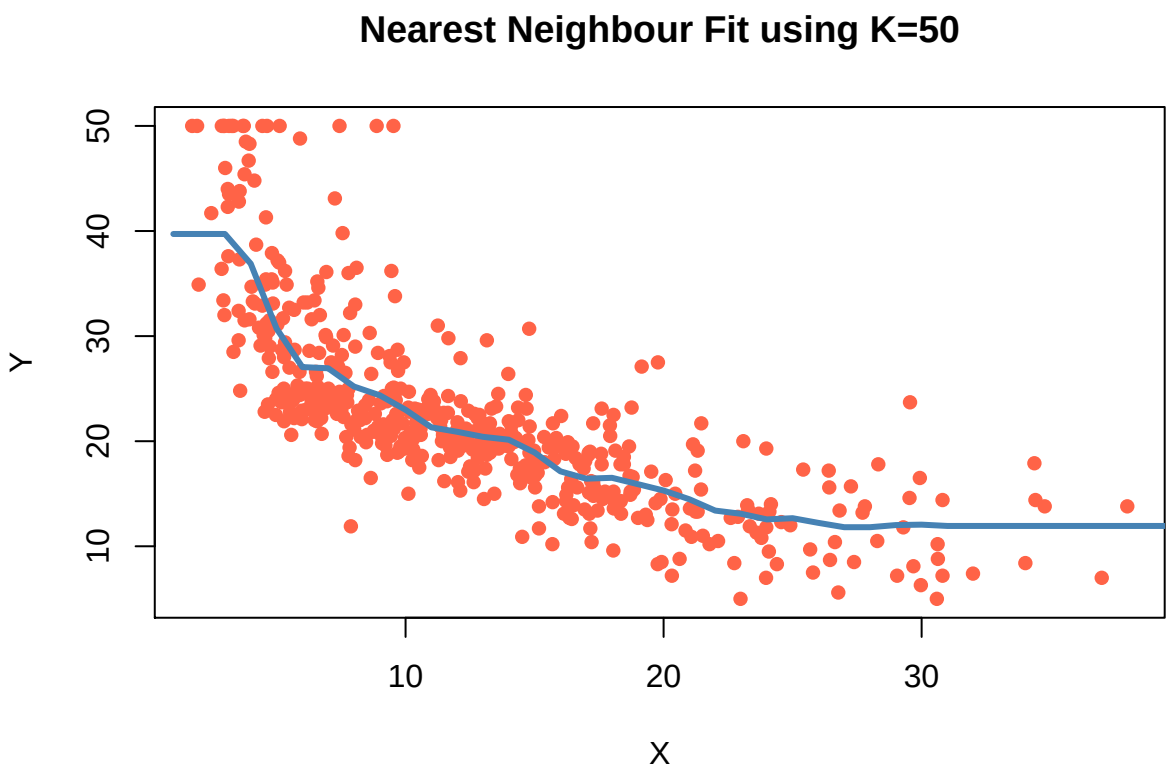
```
##         [,1]   [,2]     [,3]    [,4]   [,5]     [,6]     [,7]
##  [1,] 42.84 42.495 41.88000 40.6875 39.718 38.27333 37.24143
##  [2,] 42.84 42.495 41.88000 40.6875 39.718 38.27333 37.24143
##  [3,] 42.17 41.375 41.60333 40.6875 39.718 38.27333 37.24143
##  [4,] 39.12 39.265 37.18667 36.7975 36.890 36.83000 36.30571
##  [5,] 32.12 30.700 31.81000 31.3025 30.724 30.87000 31.30000
##  [6,] 27.53 27.585 26.53667 27.0150 27.054 27.41833 27.51714
##  [7,] 27.12 26.630 26.62667 27.4000 26.940 26.86167 26.45429
##  [8,] 24.84 24.015 24.63333 25.0700 25.202 25.76333 25.32714
##  [9,] 25.26 23.940 24.29333 24.8325 24.368 24.45000 23.98714
## [10,] 22.00 21.155 22.04000 23.0475 22.986 22.82000 23.25571
## [11,] 23.45 21.880 22.05000 21.4375 21.318 21.38667 21.46000
## [12,] 20.58 21.095 20.54333 20.8200 20.894 21.05667 20.85714
## [13,] 19.82 20.540 20.16000 20.2525 20.416 20.40833 20.59571
## [14,] 20.83 20.420 19.75667 19.9675 20.140 19.93167 19.74000
## [15,] 16.78 18.870 18.28000 18.5975 18.922 18.77000 18.57000
## [16,] 18.07 17.145 16.89667 16.9625 17.132 17.40500 17.36143
## [17,] 16.17 15.970 16.34333 16.6325 16.434 16.50667 16.73143
## [18,] 16.48 16.405 16.54000 16.3250 16.516 16.31000 16.25429
## [19,] 17.43 16.765 16.01000 15.7650 15.914 15.73167 15.77429
## [20,] 13.70 14.195 14.96000 15.2025 15.304 15.46000 15.35143
## [21,] 14.28 13.440 14.13333 14.3050 14.472 14.62833 14.55429
## [22,] 13.56 13.680 13.02333 13.3100 13.396 14.08167 14.02857
```

```
## [23,] 12.28 12.115 13.12000 12.8275 13.094 13.17667 13.67429
## [24,] 11.84 12.075 12.06333 12.7750 12.556 12.87167 13.31857
## [25,] 12.32 12.050 12.08667 12.3125 12.674 12.74667 12.75429
## [26,] 11.74 12.265 12.21000 12.4100 12.226 12.48000 12.33714
## [27,] 12.21 12.140 12.45333 11.9625 11.816 12.37667 12.25000
## [28,] 11.75 12.305 11.71000 11.7200 11.816 12.25833 12.38143
## [29,] 13.03 11.615 11.68000 11.6475 12.018 12.26667 12.37857
## [30,] 11.94 11.705 11.65000 12.0275 12.058 12.07333 12.38286
## [31,] 10.76 11.860 11.78667 11.7700 11.934 12.07333 12.38286
## [32,]  9.23 11.890 11.77000 11.7700 11.934 12.07333 12.38286
## [33,] 10.75 11.550 11.65667 11.7700 11.934 12.07333 12.38286
## [34,] 10.95 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [35,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [36,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [37,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [38,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [39,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
## [40,] 11.31 11.715 11.65667 11.7700 11.934 12.07333 12.38286
```

Results

The results based on our implementation of the regression using the K-Nearest Neighbour algorithm are described in the below in terms of the fitted blue regression curve in the plot below.

```
plot(X, Y, pch=16, col = "tomato", main = "Nearest Neighbour Fit using K=50")
points(t, predictions[,5], type="l", col="steelblue", lwd=3)
```




```
par(mfrow=c(2,3))
plot(X, Y, pch=16, col = "tomato", main = "K=10")
points(t, predictions[,1], type="l", col="blue", lwd=3)
```

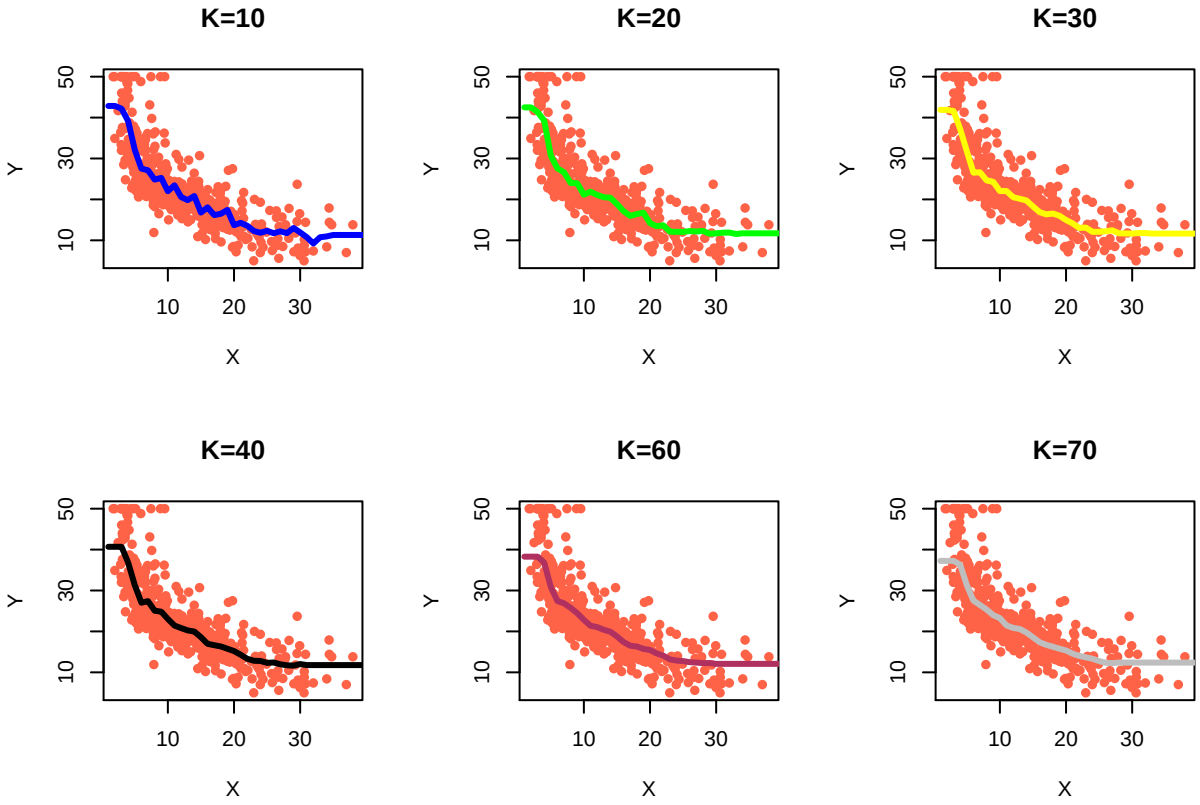
```
plot(X, Y, pch=16, col = "tomato", main = "K=20")
points(t, predictions[,2], type="l", col="green", lwd=3)

plot(X, Y, pch=16, col = "tomato", main = "K=30")
points(t, predictions[,3], type="l", col="yellow", lwd=3)

plot(X, Y, pch=16, col = "tomato", main = "K=40")
points(t, predictions[,4], type="l", col="black", lwd=3)

plot(X, Y, pch=16, col = "tomato", main = "K=60")
points(t, predictions[,6], type="l", col="maroon", lwd=3)

plot(X, Y, pch=16, col = "tomato", main = "K=70")
points(t, predictions[,7], type="l", col="grey", lwd=3)
```



In order to examine the fit of our regression model, we have used the **geom_smooth** fuction from the ggplot2 package to visually examine the fit. The model fit using the LOESS Curve Fitting (Local Regression) method is descrived below in form of the blue curve. We can observe that it almost follows the similar fit that we had obtained using our method.

Also, we had tried to further go on towards comparing the model fit visually using the fits of "Generalized Linear Models". The fit obtained using a standard GLM method with families gaussian, log normal and gamma respectively are described below-

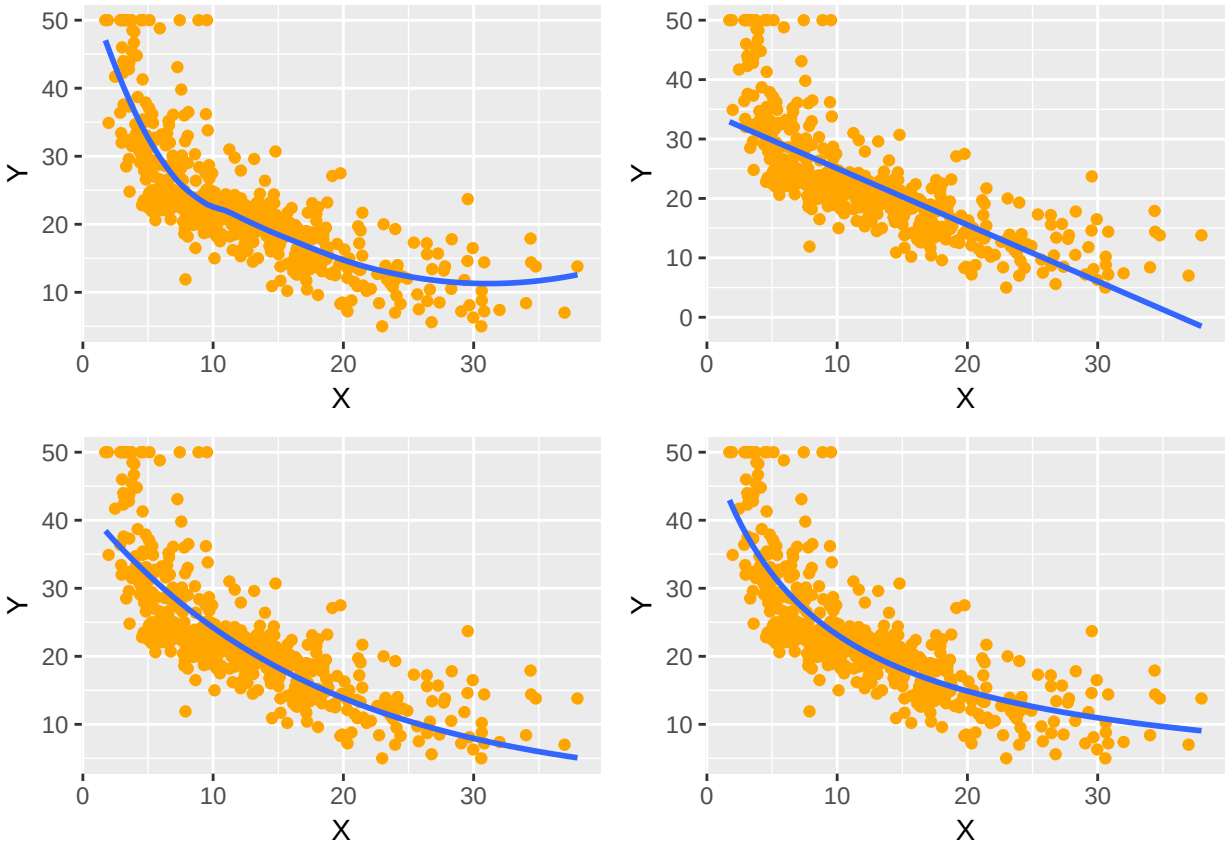
```
l <- ggplot(data = mydf, aes(x= X, y= Y))
l <- l + geom_point(color="orange") +
  geom_smooth(method = "glm",
              method.args = list(family = "gaussian"), se=F)
```

```
g <- ggplot(data = mydf, aes(x= X, y= Y))
g <- g + geom_point(color="orange") +
  geom_smooth(method = "loess", se=F)

m <- ggplot(data = mydf, aes(x= X, y= Y))
m <- m + geom_point(color="orange") +
  geom_smooth(method = "glm",
              method.args = list(family = gaussian(link="log")), se=F)


n <- ggplot(data = mydf, aes(x= X, y= Y))
n <- n + geom_point(color="orange") +
  geom_smooth(method = "glm",
              method.args = list(family = Gamma(link = "inverse")), se=F)

library(gridExtra)
grid.arrange(g,l, m, n, nrow=2,ncol=2)
```



Discussions

In order to determine, how good we have fitted our model, we have tried to compare the result of fit on predicted values of t using the generalized linear models with families gaussian, log normal and gamma. The result can be compared using the values of mean squared error below-

```
glm_gaussian <- glm(Y~X, family = gaussian(link = "identity"))
glm_log_normal <- glm(Y~X, family = gaussian(link = "log"))
```

```
glm_gamma <- glm(Y~X, family = Gamma(link = "inverse"))

#t <- as.data.frame(t)
glm_gaussian_prediction <-predict(glm_gaussian,
                                  data.frame(X =seq(40)), type = "response")
glm_lognormal_prediction <-predict(glm_log_normal,
                                   data.frame(X =seq(40)), type = "response")
glm_gamma_prediction <-predict(glm_gamma,
                               data.frame(X =seq(40)), type = "response")

mse_gaussian <- apply((glm_gaussian_prediction - predictions)^2 , 2, mean)
mse_lognormal <- apply((glm_lognormal_prediction - predictions)^2 , 2, mean)
mse_gamma <- apply((glm_gamma_prediction - predictions)^2 , 2, mean)

print(mse_gaussian)
```

```
## [1] 41.41590 43.90252 42.33468 41.03369 40.90638 40.52811 41.59718
```

```
print(mse_lognormal)
```

```
## [1] 11.91784 13.34406 12.44871 11.95530 12.32555 12.70169 13.81727
```

```
print(mse_gamma)
```

```
## [1] 2.804754 3.116554 2.670774 2.768246 3.309746 4.067819 5.114423
```

Also, we can understand that for a very small as well as very large value of K, we have problem in overfitting and underfitting. Textual references suggest that a proper value of K can be obtained by cross validation and using Elbow method.

Conclusion

It had been investigated thoroughly how this function performs for different values of K. Also, we understood how varying the K changes decision boundary. It now becomes imperative to learn how cross validation works in order to choose correct K. ## References