First Deliverable: K-NN for Regression

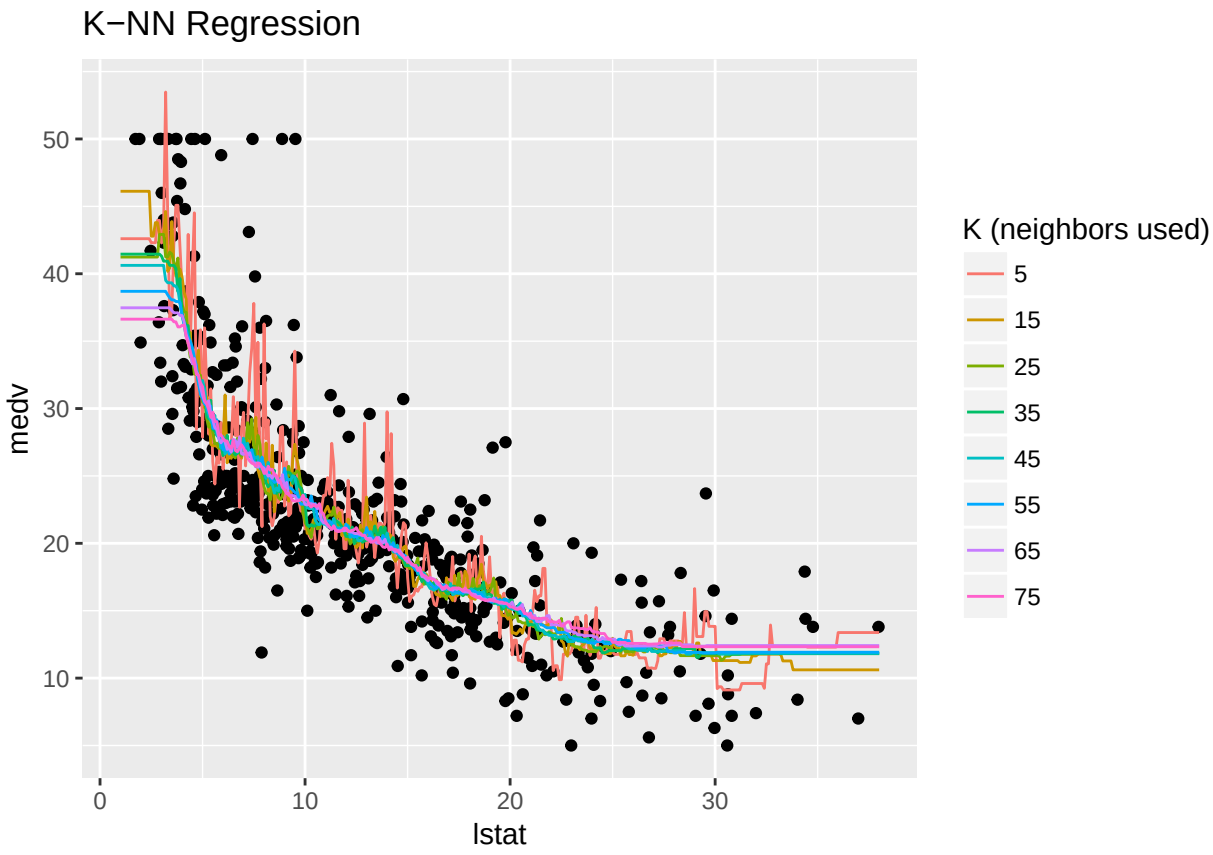
Daniel Fuentes
Marc Valenti

11 February 2018

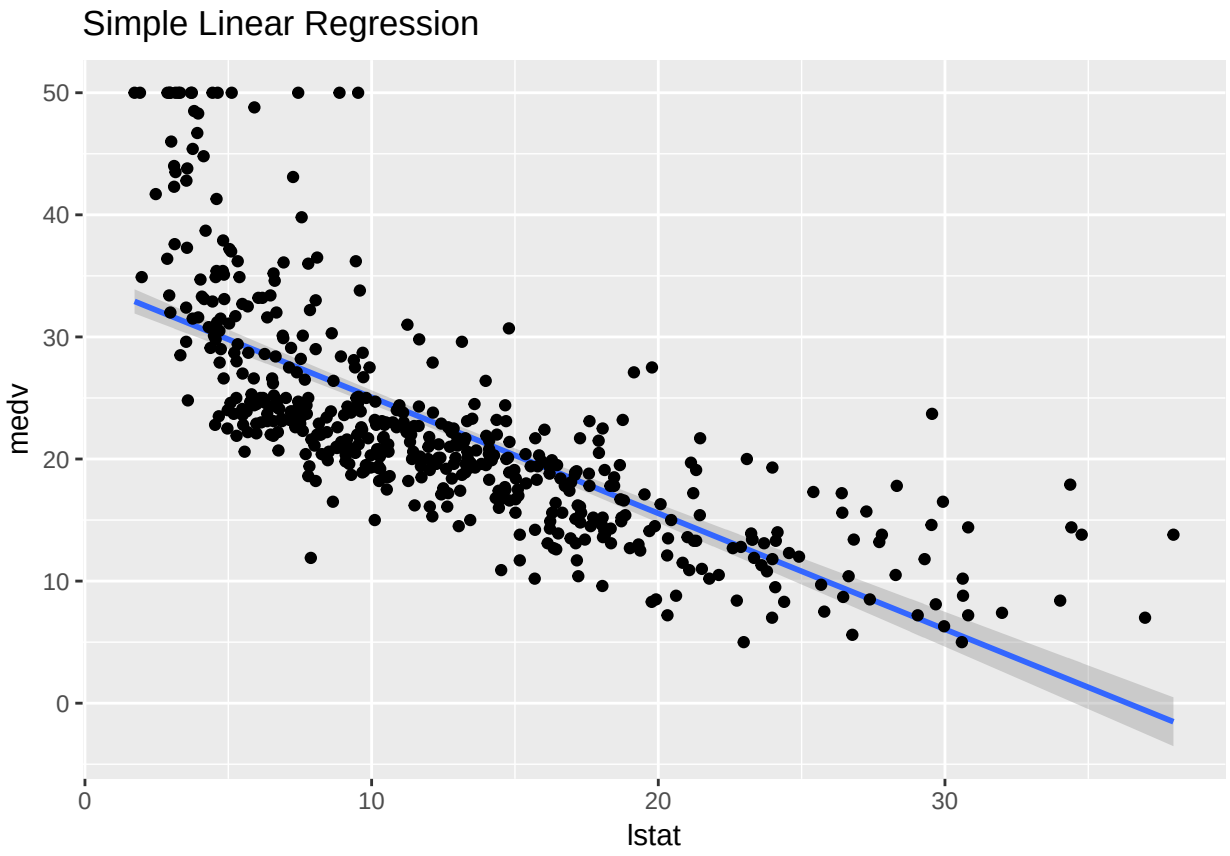
Estimate the regression function of `Boston$medv` as a function of `Boston$lstat` using k-nn In this case, is is a regression problem instead of a classification one, because the response variable is a continous instead of a categorical. Therefore, the prediction function will get the average of the response variable of all those k observations that are closer to a observation.

```
v<-rep(seq(from=floor(min(Boston$lstat)),
    to=ceiling(max(Boston$lstat)),
    by=0.1),8)
k_vector<-seq(from=5,
              to=75,
              by=10)
res_matrix<-expand.grid(k_vector,v) #i get all the combinations between the x points and the grid k
colnames(res_matrix)<-c("k","v")


knn_class_continous<-function(x,y,x_aux_vector,k){
  #gets the current euclidean distance
  d_st_xy <- (x-x_aux_vector)^2
  #partial sorting only sorts until x; ordena i busca la que es la numero k
  d_st_xy_k <- sort(d_st_xy,partial=k)[k]
  #obte els indexs de les observacions amb distancia inferior o igual a k (els k-nn)
  N_st_k <- unname( which(d_st_xy <= d_st_xy_k) )
  #gets the mean of the y. with weighted knn we'd have a weighted mean instead
  pr_1_k_st <- sum(y[N_st_k])/k
  return(pr_1_k_st)
}

for (i in 1:nrow(res_matrix)){
  res_matrix[i,3]<-knn_class_continous(x=Boston$lstat,
                    y=Boston$medv,
                    x_aux_vector=res_matrix[i,'v'],
                    k=res_matrix[i,'k'])
  }
colnames(res_matrix)<-c("k","v","hat_p")




p1<-ggplot() +
  geom_point(data=Boston,
       aes(x=lstat,
           y=medv)) +
  geom_line(data=res_matrix,
            aes(x=v,
                y=hat_p,
                col=as.factor(k))) +
```

```
  labs(title='K-NN Regression',
       col='K (neighbors used)')
p2<-ggplot(data=Boston,
       aes(x=lstat,
           y=medv)) +
  geom_smooth(method = "lm") +
  geom_point() +
  labs(title='Simple Linear Regression')


p1
```




```
p2
```



Having a quick Machine Learning approach; optimizing k so that MSE of the test split is minimized.

```
set.seed(100218)
train_idx<-sample(nrow(Boston),floor(nrow(Boston)*0.7))

for (i in 1:nrow(res_matrix)){
  #now the x and y are restricted to those of the training set.
  #rest is completely the same
  res_matrix[i,3]<-knn_class_continous(x=Boston$lstat[train_idx],
                    y=Boston$medv[train_idx],
                    x_aux_vector=res_matrix[i,'v'],
                    k=res_matrix[i,'k'])
  }
colnames(res_matrix)<-c("k","v","hat_p")



res_RMSE<-res_matrix %>% #in res_matrix there is m(t) for all t in R
  #select the test set
  inner_join(Boston[-train_idx,c('lstat','medv')],
  #and join where the x on the test set equals the m(t)
             by=c('v'='lstat')) %>%
  #obtain the squared residuals
  mutate(residuals_squared=(hat_p-medv)^2) %>%
  #aggregate by K, so that we get the MSE
  group_by(k) %>%
  summarise(MSE=mean(residuals_squared))
```

```
p3<-ggplot() +
  geom_point(data=Boston[train_idx,],
       aes(x=lstat,
           y=medv)) +
  geom_line(data=res_matrix,
            aes(x=v,
                y=hat_p,
                col=as.factor(k))) +
  labs(title='K-NN Regression using only training set')


kable(res_RMSE,digits=2)
```

k	MSE
5	42.17
15	28.03
25	25.11
35	27.58
45	30.01
55	31.26
65	34.28
75	34.99

Seems that MSE is minimized with k=25. However, further work would be iterating on this expanding the grid.