

# Nearest Neighbor Prediction with a Single Predictor

A. Burnetas

In this note we create a function that performs regression on a dataset with a single predictor  $X$  and outcome  $Y$  using the Nearest-Neighbor method. The distance metric used is the absolute difference. Specifically:

1. Given is a dataset  $D = \{(x_i, y_i), i = 1, \dots, N\}$  (training set). Based on this, we need to make a prediction  $\hat{y} = f(x_0)$  for the value of  $y$  at an arbitrary point  $x_0$  (which may be one of the values  $x_i$  or not).
2. The Nearest-Neighbor (nnk) method with parameter  $k$  works as follows: We find the  $k$  points of the dataset that are closest to  $x$  in the  $x$  dimension, i.e., we define the  $k$ -neighborhood of  $x$  :  $N_k(x) \subseteq D$  such that  $|N_k(x)| = k$  and for all  $(x_i, y_i) \in N_k(x)$  it is true that  $|x_i - x| \leq |x_j - x|$  for all  $(x_j, y_j) \notin N_k(x)$ . Ties are resolved arbitrarily.
3. The prediction of  $y$  at  $x$  is equal to the average value of  $y$  for all datapoints in the neighborhood of  $x$ :

$$\hat{y} = f(x) = \frac{1}{k} \sum_{(x_i, y_i) \in N_k(x)} y_i.$$

To implement this algorithm, we build a `nnk1` function which takes as arguments the  $x$  and  $y$  vectors, the parameter  $k$  and the point  $x_0$  at which the prediction will be made, and returns the prediction  $f(x_0)$ . There are several possible ways to program finding the  $k$  closest neighbors of  $x_0$ . Here we use the following idea: Create the vector  $(x_i - x_0, i = 1, \dots, N)$ , and sort it in increasing order. In this way, the first  $k$  elements of the sorted vector correspond to the distances from  $x_0$  of the  $k$  closest neighbors. However we still need to know which observations these correspond to, so that we can find the corresponding  $y_i$ . To do this, we use the option `index.return` of the R sort function. This function works as follows: If we want to sort a vector  $b$  in ascending order then the command

```
a=sort(b, index.return=TRUE)
```

returns a list of two elements: `a$x` contains the sorted vector  $b$  and `a$ix` the indices of the sorted elements (e.g., if the minimum element of  $b$  is  $b_4 = 8$ , then `a$x=(8,...)` and `a$ix=(4,...)`)

```
nnk1=function(x,y,k,x0)
{
  xdist=abs(x-x0)
  dsort=sort(xdist, index.return=TRUE)
  Nk=dsort$ix[1:k]
  f=mean(y[Nk])
  return(f)
}
```

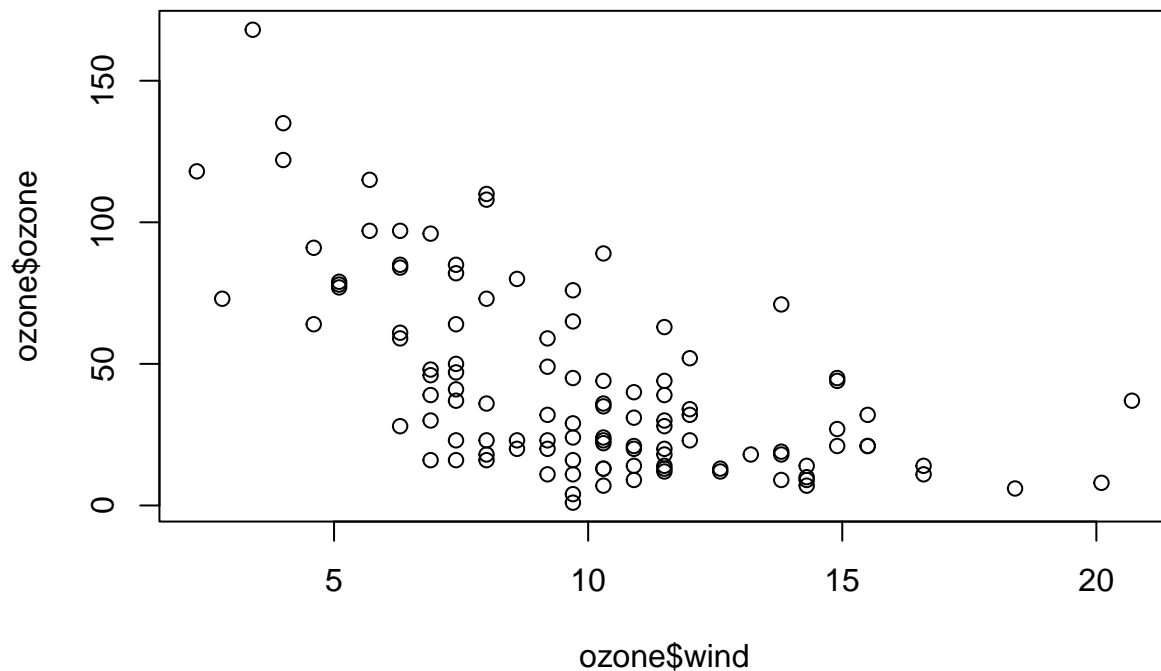
We can now apply the function on dataset `ozone.data` from Friedman et al. (2009). We first import the text file to an R dataframe. The file `ozone.data` uses tabs to separate the fields in each line, so we must use the option `sep="\t"` in order to read it correctly.

```
ozone=read.csv("ozone.data", sep="\t")
```

The data contain measurements of ozone emissions in New York in some time period in 1970's together with data on radiation, temperature and wind speed in appropriate units. Since our function applies to data with

a single predictor, we will use as  $x$  the wind and as  $y$  the ozone emissions.

```
plot(ozone$wind, ozone$ozone)
```



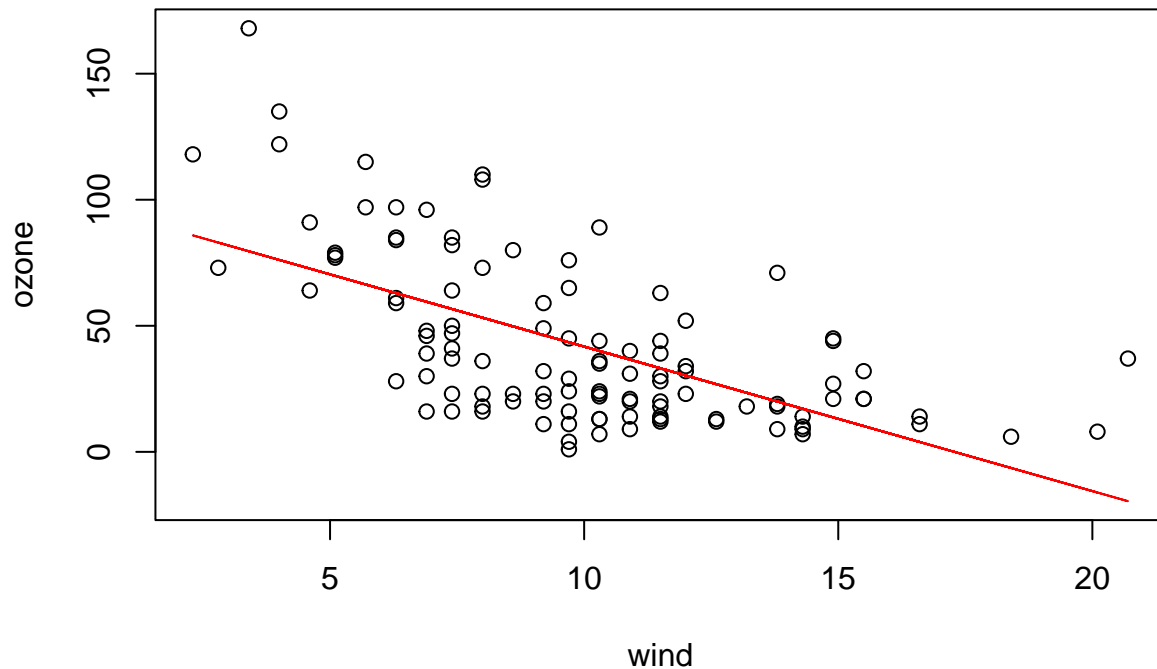
Before we turn to nearest neighbor methods, we apply a simple linear regression model between ozone and wind:

```
l1=lm(ozone~wind, data=ozone)
summary(l1)
```

```
##
## Call:
## lm(formula = ozone ~ wind, data = ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -43.513 -18.590  -5.029  15.819  88.430
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  99.0542     7.4656  13.268  < 2e-16 ***
## wind        -5.7306     0.7075  -8.099  8.65e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.41 on 109 degrees of freedom
## Multiple R-squared:  0.3757, Adjusted R-squared:  0.37
## F-statistic: 65.6 on 1 and 109 DF, p-value: 8.652e-13
```

We see that the correlation between ozone and wind speed is statistically significant, and the wind explains 37% of the variation of ozone concentration.

```
plot(ozone~wind, data=ozone, ylim=c(min(l1$fitted.values),max(ozone)))
lines(l1$fitted.values~ozone$wind, col='red')
```



We now consider the nearest neighbor method. We apply it for values of  $k = 2, 10, 50$ , each on a vector of values of wind speed  $x_0 = 0, 0.01, 0.02, \dots, 22$ . The predicted values are written on matrix  $F$ . Each column of  $F$  includes the predictions of ozone for all values of  $x$  for the corresponding value of  $k$ :

```
x0=seq(0,22,by=0.01)
K=c(2,10,50)
nx0=length(x0)
nK=length(K)
F=matrix(rep(0,nx0*nK), nx0, nK)
for (i in 1:nK)
{
  F[,i] = sapply(x0, nnk1, x=ozone$wind, y=ozone$ozone, k=K[i])
}
```

We next plot the predictions for the linear regression and the nearest neighbor method with  $k = 2, 10, 50$ .

```
plot(ozone~wind, data=ozone, ylim=c(min(l1$fitted.values),max(ozone)))
lines(l1$fitted.values~ozone$wind, col='red')
lines(F[,1]~x0,col='blue')
lines(F[,2]~x0,col='green')
lines(F[,3]~x0,col='brown')
legend(17,140, legend=c("Regression", "NN(2)", "NN(10)", "NN(50)"), col=c("red", "blue", "green", "brown"))
```

