R Notebook

Title: "IST687 – Support Vector Machines HW"
Name: Sathish Kumar Rajendiran
Week: 9
Date: 06/08/2020

Exercise: Support Vector Machines HW

Install necessary packages

```
install.packages( pkgs=c("kernlab","e1071","gdata","RCurl","ggplot2","ggcorrplot","reshape2","ggeas
```

```
##
## The downloaded binary packages are in
##  /var/folders/_z/ltmjkt4156b37rsk7cgvj7l80000gn/T//RtmpzSpqZ3/downloaded_packages
```

```
library(kernlab)
library(e1071)
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith
```

```r
library(RCurl)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha
```

```r
library(ggcorrplot)
library(reshape2)
library(ggeasy)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
library(viridisLite)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:gdata':
##
##     combine
```

```r
# function printDataInfo
  printDataInfo <- function(myData)
    {
      strinfo <- str(myData)
      cat("str:",strinfo,"\n")

      colnamesinfo <- colnames(myData)
      cat("colnames:",colnamesinfo,"\n")

      diminfo <- dim(myData)
      cat("dim:",diminfo,"\n")

      nrowinfo <- nrow(myData)
      cat("nrow:",nrowinfo,"\n")

      nrowsinfo <- myData[1:3,]
      return(nrowsinfo)
     }
```

```r
  # import airquality dataset
  ?airquality

  myairquality <- data.frame(airquality,stringsAsFactors=FALSE)

  # import airquality dataset
  printDataInfo(myairquality)
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
## str:
## colnames: Ozone Solar.R Wind Temp Month Day
## dim: 153 6
## nrow: 153
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
```

```r
#look for columns having NAs
clnames <- colnames(myairquality)[colSums(is.na(myairquality)) > 0]
clnames
```

```
## [1] "Ozone"   "Solar.R"
```

```r
 #  create subset of dataframe rows having NAs
na_data <- myairquality[rowSums(is.na(myairquality)) > 0,]
na_data # 680- rows
```

```
##     Ozone Solar.R Wind Temp Month Day
## 5      NA      NA 14.3   56     5   5
## 6      28      NA 14.9   66     5   6
## 10     NA     194  8.6   69     5  10
## 11      7      NA  6.9   74     5  11
## 25     NA      66 16.6   57     5  25
## 26     NA     266 14.9   58     5  26
## 27     NA      NA  8.0   57     5  27
## 32     NA     286  8.6   78     6   1
## 33     NA     287  9.7   74     6   2
## 34     NA     242 16.1   67     6   3
## 35     NA     186  9.2   84     6   4
## 36     NA     220  8.6   85     6   5
## 37     NA     264 14.3   79     6   6
## 39     NA     273  6.9   87     6   8
## 42     NA     259 10.9   93     6  11
## 43     NA     250  9.2   92     6  12
## 45     NA     332 13.8   80     6  14
## 46     NA     322 11.5   79     6  15
## 52     NA     150  6.3   77     6  21
## 53     NA      59  1.7   76     6  22
## 54     NA      91  4.6   76     6  23
## 55     NA     250  6.3   76     6  24
## 56     NA     135  8.0   75     6  25
## 57     NA     127  8.0   78     6  26
## 58     NA      47 10.3   73     6  27
```

```
## 59     NA      98 11.5   80     6   28
## 60     NA      31 14.9   77     6   29
## 61     NA     138  8.0   83     6   30
## 65     NA     101 10.9   84     7    4
## 72     NA     139  8.6   82     7   11
## 75     NA     291 14.9   91     7   14
## 83     NA     258  9.7   81     7   22
## 84     NA     295 11.5   82     7   23
## 96     78      NA  6.9   86     8    4
## 97     35      NA  7.4   85     8    5
## 98     66      NA  4.6   87     8    6
## 102    NA     222  8.6   92     8   10
## 103    NA     137 11.5   86     8   11
## 107    NA      64 11.5   79     8   15
## 115    NA     255 12.6   75     8   23
## 119    NA     153  5.7   88     8   27
## 150    NA     145 13.2   77     9   27
```

```
 # find and replace NAs on Ozone Column
myairquality$Ozone[is.na(myairquality$Ozone)] <- round(as.numeric(mean(myairquality$Ozone, na.rm=TRU

 # find and replace NAs on Solar.R Column
myairquality$Solar.R[is.na(myairquality$Solar.R)] <- round(as.numeric(mean(myairquality$Solar.R, na

 #  Verify if the dataframe has NAs
na_data <- myairquality[rowSums(is.na(myairquality)) > 0,]
na_data # 680- rows
```

```
## [1] Ozone   Solar.R Wind    Temp    Month   Day
## <0 rows> (or 0-length row.names)
```

```
myairquality[1:10,]
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     42     186 14.3   56     5   5
## 6     28     186 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    42     194  8.6   69     5  10
```

```
# Plot histogram on countries data to analyze the data spread for missing data elements
hcolor <- c("orange")
hfill <- c("steelblue")
htitle <- c("Histogram - Data availability")
theme <-theme(plot.title = element_text(hjust = 0.5),axis.title = element_text())

gghistAll <- ggplot(data=melt(myairquality),mapping = aes(x= value))
```

```
## No id variables; using all as measure variables
```

```r
gghistAll <- gghistAll+geom_histogram(bins = 20,color=hcolor,fill=hfill,na.rm = TRUE) + facet_wrap(
gghistAll +
  labs(x = NULL, y = NULL,title = "New York Air Quality Measurements", caption = "Data: Daily air qu
```


```r
# find total number of rows

airQ <- myairquality
nrows <- nrow(airQ)
nrows
```

```
## [1] 153
```

```r
# Prepare train and test datasets based on the total number of rows | 2/3 --> Train dataset and 1/3 -

#find the cut point by taking 2/3rd of the total number of rows
cutPoint <- floor(2*nrows/3)
cutPoint
```

```
## [1] 102
```

```r
#prepare random sample - as we're not sure about the data arrangement

rand <- sample(1:nrows)
head(rand)
```

```
## [1]  87  82 116  68   7  49
```

```r
airQ.train <- airQ[rand[1:cutPoint],]
airQ.test  <- airQ[rand[(cutPoint+1):nrows],]

str(airQ.train)  # 102 observations
```

```
## 'data.frame':    102 obs. of  6 variables:
##  $ Ozone  : num  20 16 45 77 23 20 64 42 23 1 ...
##  $ Solar.R: num  81 7 212 276 299 37 253 135 115 8 ...
##  $ Wind   : num  8.6 6.9 9.7 5.1 8.6 9.2 7.4 8 7.4 9.7 ...
##  $ Temp   : int  82 74 79 88 65 65 83 75 76 59 ...
##  $ Month  : int  7 7 8 7 5 6 7 6 8 5 ...
##  $ Day    : int  26 21 24 7 7 18 30 25 18 21 ...
```

```r
str(airQ.test) # 51 observations
```

```
## 'data.frame':    51 obs. of  6 variables:
##  $ Ozone  : num  59 42 42 66 122 42 28 14 42 65 ...
##  $ Solar.R: num  254 64 222 186 255 255 186 334 332 157 ...
##  $ Wind   : num  9.2 11.5 8.6 4.6 4 12.6 14.9 11.5 13.8 9.7 ...
##  $ Temp   : int  81 79 92 87 89 75 66 64 80 80 ...
##  $ Month  : int  7 8 8 8 8 8 5 5 6 8 ...
##  $ Day    : int  31 15 10 6 7 23 6 16 14 14 ...
```

```r
airQ.train[1:10,]
```

```
##     Ozone Solar.R Wind Temp Month Day
## 87     20      81  8.6   82     7  26
## 82     16       7  6.9   74     7  21
## 116    45     212  9.7   79     8  24
## 68     77     276  5.1   88     7   7
## 7      23     299  8.6   65     5   7
## 49     20      37  9.2   65     6  18
## 91     64     253  7.4   83     7  30
## 56     42     135  8.0   75     6  25
## 110    23     115  7.4   76     8  18
## 21      1       8  9.7   59     5  21
```

```r
airQ.test[1:10,]
```

```
##     Ozone Solar.R Wind Temp Month Day
## 92     59     254  9.2   81     7  31
## 107    42      64 11.5   79     8  15
## 102    42     222  8.6   92     8  10
```

```
## 98      66      186  4.6   87      8   6
## 99     122      255  4.0   89      8   7
## 115     42      255 12.6   75      8  23
## 6       28      186 14.9   66      5   6
## 16      14      334 11.5   64      5  16
## 45      42      332 13.8   80      6  14
## 106     65      157  9.7   80      8  14
```

```
# Plot histogram on countries data to analyze the data spread for missing data elements

    gghist <- ggplot(data=melt(airQ),mapping = aes(x= value))
```

```
## No id variables; using all as measure variables
```

```
    gghist <- gghist+geom_histogram(bins = 20,color=hcolor,fill=hfill,na.rm = TRUE) + facet_wrap(~varia
    gghist +
      labs(x = NULL, y = NULL, title="New York Air Quality Measurements", caption = "Data: Daily air qua
```


```
# 1) Build a model (using the 'ksvm' function, trying to predict onzone).
# You can use all the possible attributes, or select the attributes that you think would be the most he

    #predict Ozone values from all the variables in the train dataset

    model <- ksvm(Ozone~.,data=airQ.train,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.model=TRU
    model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 10
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.199322268125054
##
## Number of Support Vectors : 81
##
## Objective Function Value : -174.466
## Training error : 0.140995
## Cross validation error : 504.8321
## Laplace distr. width : 43.37837
```

```
# 2) Test the model on the testing dataset, and compute the Root Mean Squared Error

    pred <- predict(model,airQ.test,type="votes")
    str(pred)
```

```
##  num [1:51, 1] 47 24.7 105.3 84.7 104.9 ...
```

```
    # create a dataframe to have actual and preicted Ozone values
    model.df <- data.frame(airQ.test$Ozone,pred[,1],stringsAsFactors=FALSE)
    colnames(model.df) <- c("Test.Ozone","Pred.Ozone")
    head(model.df)
```

```
##   Test.Ozone Pred.Ozone
## 1         59   47.00424
## 2         42   24.71415
## 3         42  105.28084
## 4         66   84.71054
## 5        122  104.93586
## 6         42   28.97585
```

```
    #compute the Root Mean Squared Error

    rmseKSVM <- sqrt(mean((model.df$Test.Ozone-model.df$Pred.Ozone)^2))
    cat("Root Mean Squared Error from KSVM model is :",rmseKSVM)
```

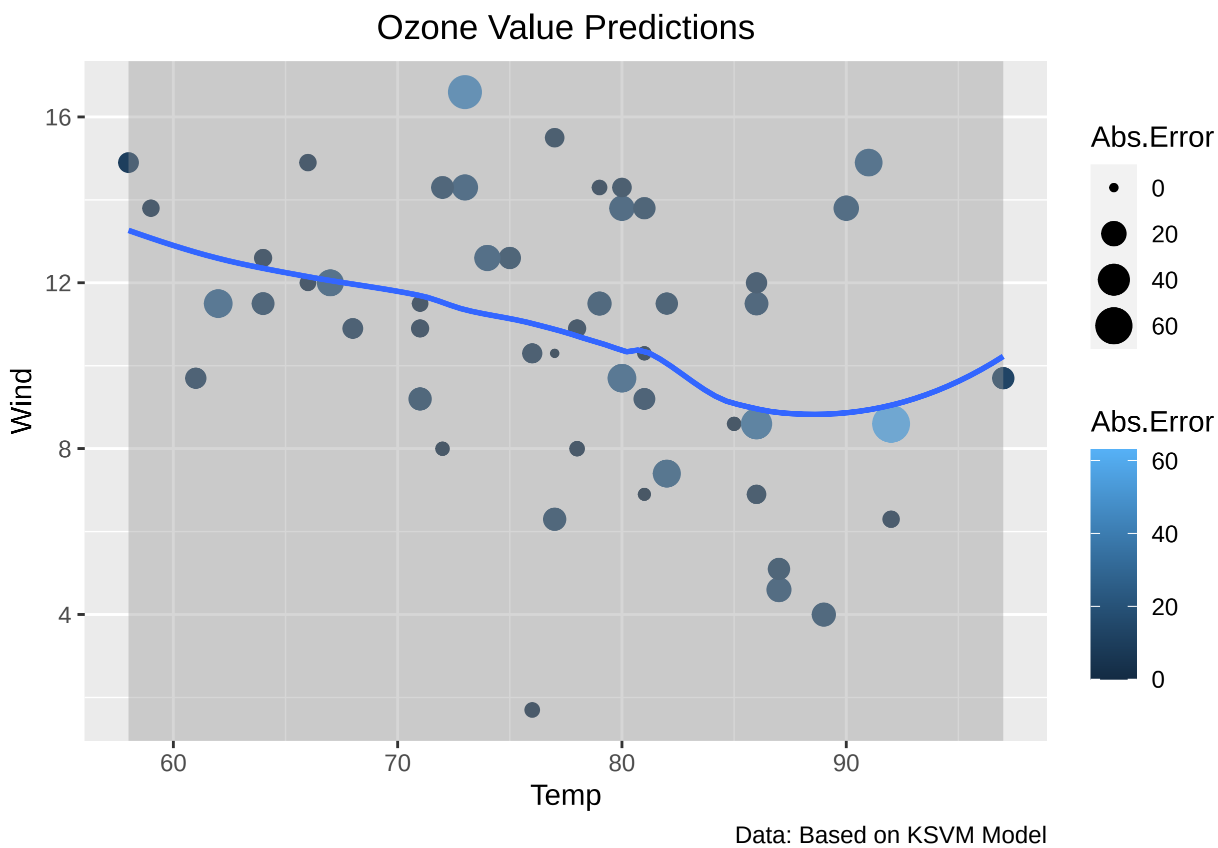
```
## Root Mean Squared Error from KSVM model is : 18.22862
```

```
#  3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis represent
    # the point size and color represent the error, as defined by the actual ozone level minus the pred

    abs.Error <- round(abs(model.df$Test.Ozone-model.df$Pred.Ozone),0)
    abs.Error
```

```
##  [1] 12 17 63 19 17 13  5 14 20 29 11 15  6 13 22  2  8  3  9  8 37 14 29  6 15
## [26] 13 13 22  1  2  4  4  2 16 10 20  3 13 47  5 27 24 11  6  0  3  1  8 26  5
## [51] 10
```

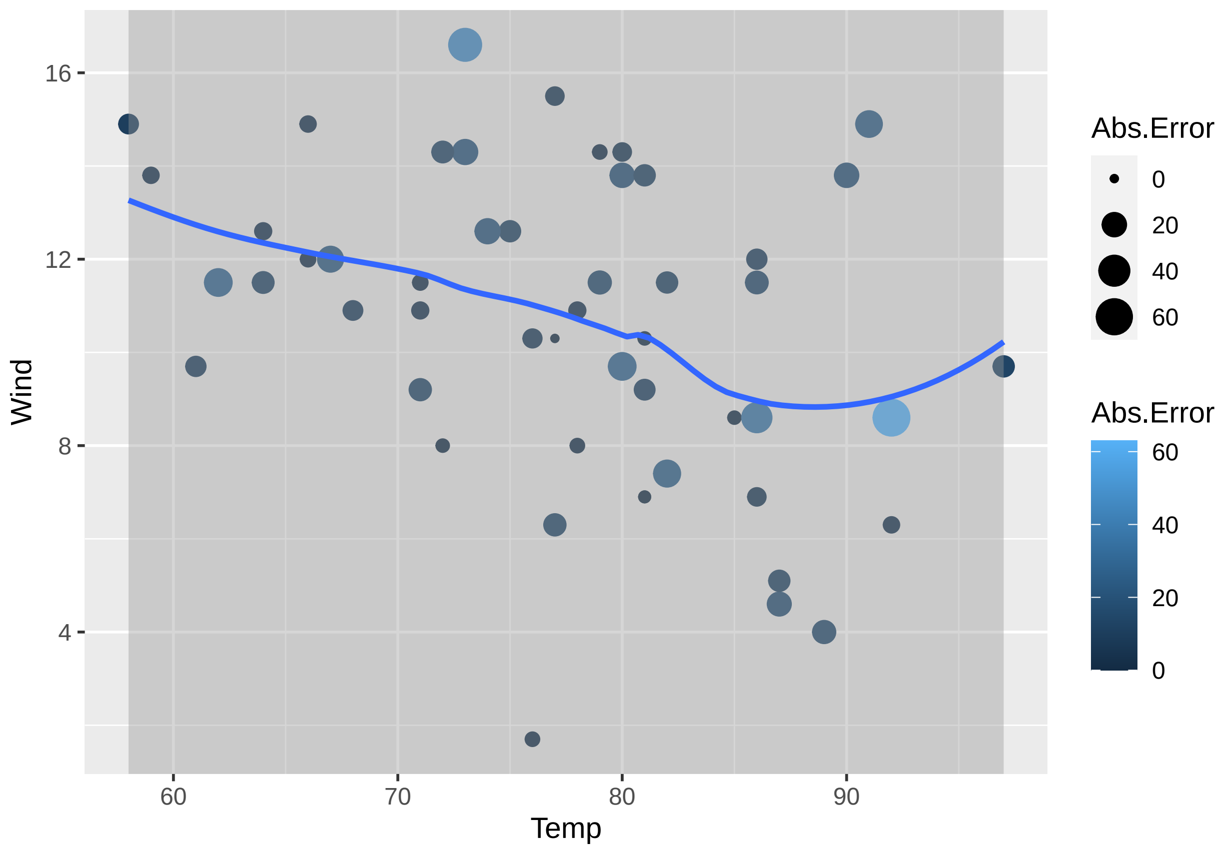
```
#create a dataframe
plot.df <- data.frame(airQ.test$Temp,airQ.test$Wind,abs.Error)
colnames(plot.df) <- c("Temp","Wind","Abs.Error")

ggscatterKSVM <- ggplot(plot.df,aes(x= Temp,y=Wind))
ggscatterKSVM <- ggscatterKSVM + geom_point(aes(size=Abs.Error,color=Abs.Error)) + geom_smooth(meth

ggscatterKSVM + labs(title="Ozone Value Predictions", caption = "Data: Based on KSVM Model")+theme
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggscatterKSVM
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
    ksvm_plot <- ggscatterKSVM+theme(legend.position="none") + ggtitle("ksvm model") +theme
    #------------------------------------------------------------------------------------------

# 4) Compute models and plot the results for 'svm' (in the e1071 package) and 'lm'.
# Generate similar charts for each model

    #predict Ozone values from all the variables in the train dataset using SVM from e1071 package

    modelSVM <- svm(Ozone~.,data=airQ.train,prob.model=TRUE)
    modelSVM
```

```
##
## Call:
## svm(formula = Ozone ~ ., data = airQ.train, prob.model = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  83
```

```r
predSVM <- predict(modelSVM,airQ.test,type="votes")
str(predSVM)
```

```
##  Named num [1:51] 53.7 21.8 78.5 76.9 86.7 ...
##  - attr(*, "names")= chr [1:51] "92" "107" "102" "98" ...
```

```r
# create a dataframe to have actual and preicted Ozone values
modelSVM.df <- data.frame(airQ.test$Ozone,predSVM,stringsAsFactors=FALSE)
colnames(modelSVM.df) <- c("Test.Ozone","Pred.Ozone")
head(modelSVM.df)
```

```
##     Test.Ozone Pred.Ozone
## 92          59   53.69259
## 107         42   21.80037
## 102         42   78.50024
## 98          66   76.90278
## 99         122   86.72047
## 115         42   28.59487
```

```r
#compute the Root Mean Squared Error

rmseSVM <- sqrt(mean((modelSVM.df$Test.Ozone-modelSVM.df$Pred.Ozone)^2))
cat("Root Mean Squared Error from SVM model is :",rmseSVM)
```

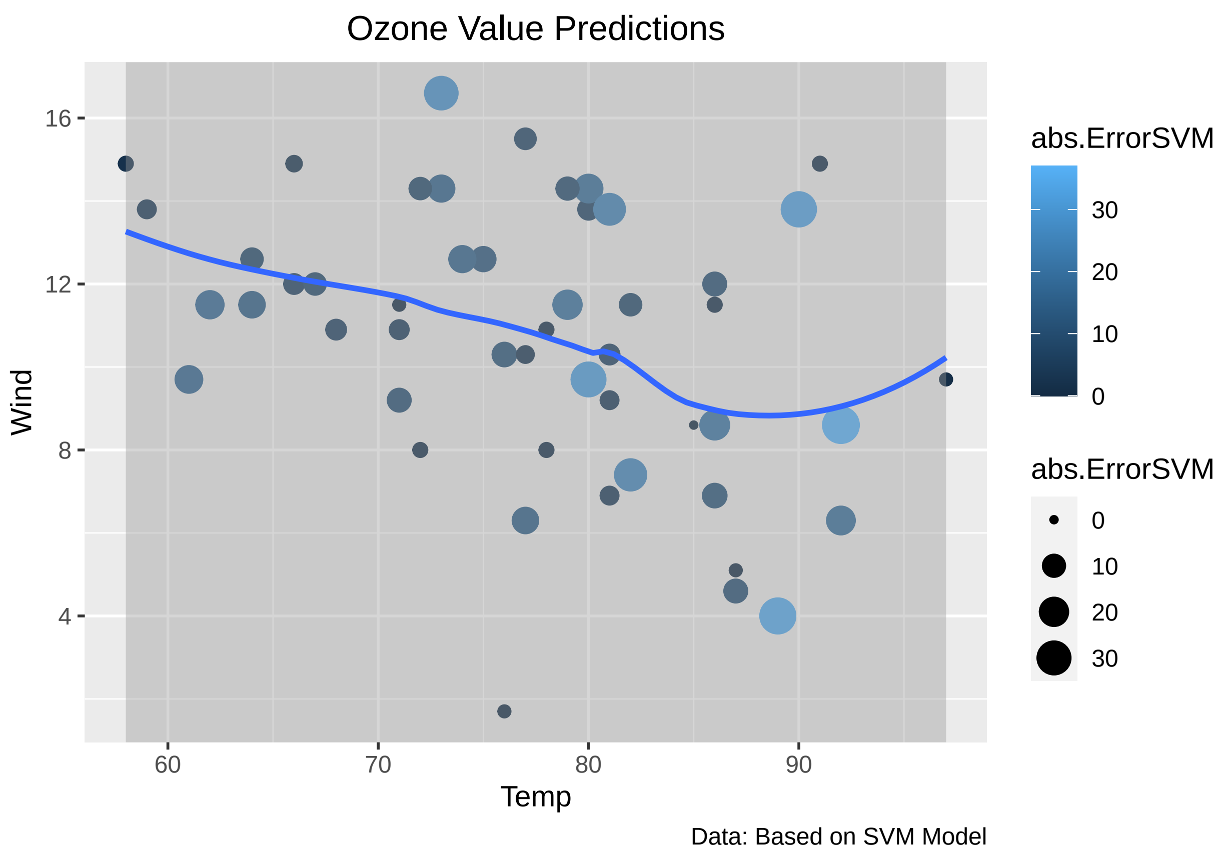
```
## Root Mean Squared Error from SVM model is : 15.15924
```

```r
abs.ErrorSVM <- round(abs(modelSVM.df$Test.Ozone-modelSVM.df$Pred.Ozone),0)
abs.ErrorSVM
```

```
##  [1]  5 20 37 11 35 13  3 15  8 32 17 11  9  9 16  0 19 10 12  8 21  9 18  2 15
## [26] 25  1 16  4  7  1  7  2  2  7 33  1  1 29  5 26  9 11  6  4  2  5 12  2 19
## [51]  2
```

```r
#create a dataframe
plot.dfSVM <- data.frame(airQ.test$Temp,airQ.test$Wind,abs.ErrorSVM)
colnames(plot.dfSVM) <- c("Temp","Wind","Abs.Error")

ggscatterSVM <- ggplot(plot.dfSVM,aes(x= Temp,y=Wind))
ggscatterSVM <- ggscatterSVM + geom_point(aes(size=abs.ErrorSVM,color=abs.ErrorSVM)) + geom_smooth(

ggscatterSVM + labs(backgroundColor="white",title="Ozone Value Predictions", caption = "Data: Based
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
svm_plot <- ggscatterSVM + theme(legend.position="none") + ggtitle("svm model") +theme

#-----------------------------------------------------------------------------------------------

#predict Ozone values from all the variables in the train dataset using LM

modelLM <- lm(formula = Ozone~., data=airQ.train)
modelLM
```

```
##
## Call:
## lm(formula = Ozone ~ ., data = airQ.train)
##
## Coefficients:
## (Intercept)      Solar.R         Wind         Temp        Month          Day
##   -38.93335      0.04221     -2.47095      1.37140     -1.61146      0.21673
```

```r
predLM <- predict(modelLM,airQ.test)
str(predLM)
```

```
##  Named num [1:51] 55.6 34.1 64.6 65.3 72.6 ...
##  - attr(*, "names")= chr [1:51] "92" "107" "102" "98" ...
```

```r
# create a dataframe to have actual and preicted Ozone values
modelLM.df <- data.frame(airQ.test$Ozone,predLM,stringsAsFactors=FALSE)
colnames(modelLM.df) <- c("Test.Ozone","Pred.Ozone")
head(modelLM.df)
```

```
##     Test.Ozone Pred.Ozone
## 92          59   55.57585
## 107         42   34.05177
## 102         42   64.63055
## 98          66   65.27105
## 99         122   72.62532
## 115         42   35.64315
```

```r
#compute the Root Mean Squared Error

rmseLM <- sqrt(mean((modelLM.df$Test.Ozone-modelLM.df$Pred.Ozone)^2))
cat("Root Mean Squared Error from LM model is :",rmseLM)
```

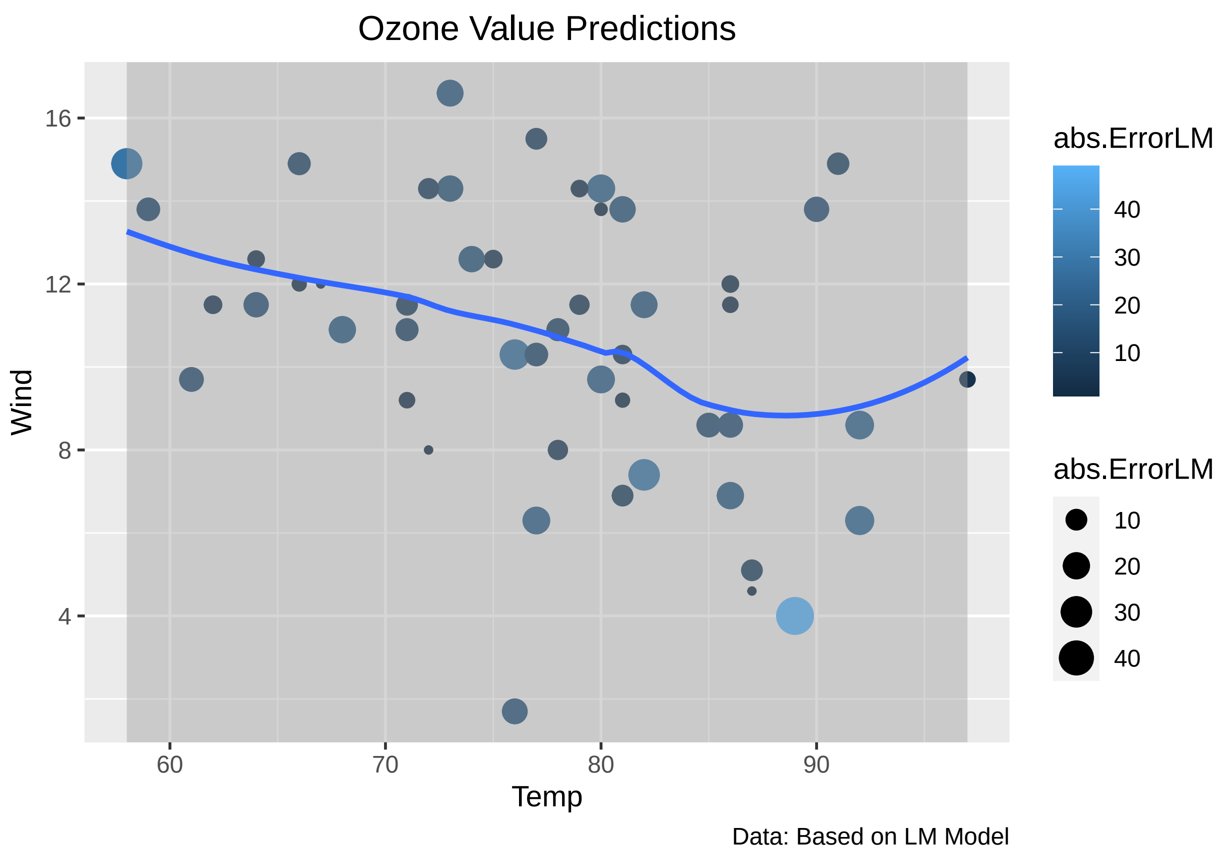
```
## Root Mean Squared Error from LM model is : 16.13633
```

```r
abs.ErrorLM <- round(abs(modelLM.df$Test.Ozone-modelLM.df$Pred.Ozone),0)
abs.ErrorLM
```

```
##  [1]  3  8 23  1 49  6 12 16  2 21 15  4  5 19 18 15 22  5 27 10 16  9  6 12 21
## [26] 18 10 18  7  7 10  3  1  4 20 16 17  4 19 13 30  1  5 12 13  8 10 20 11 24
## [51] 29
```

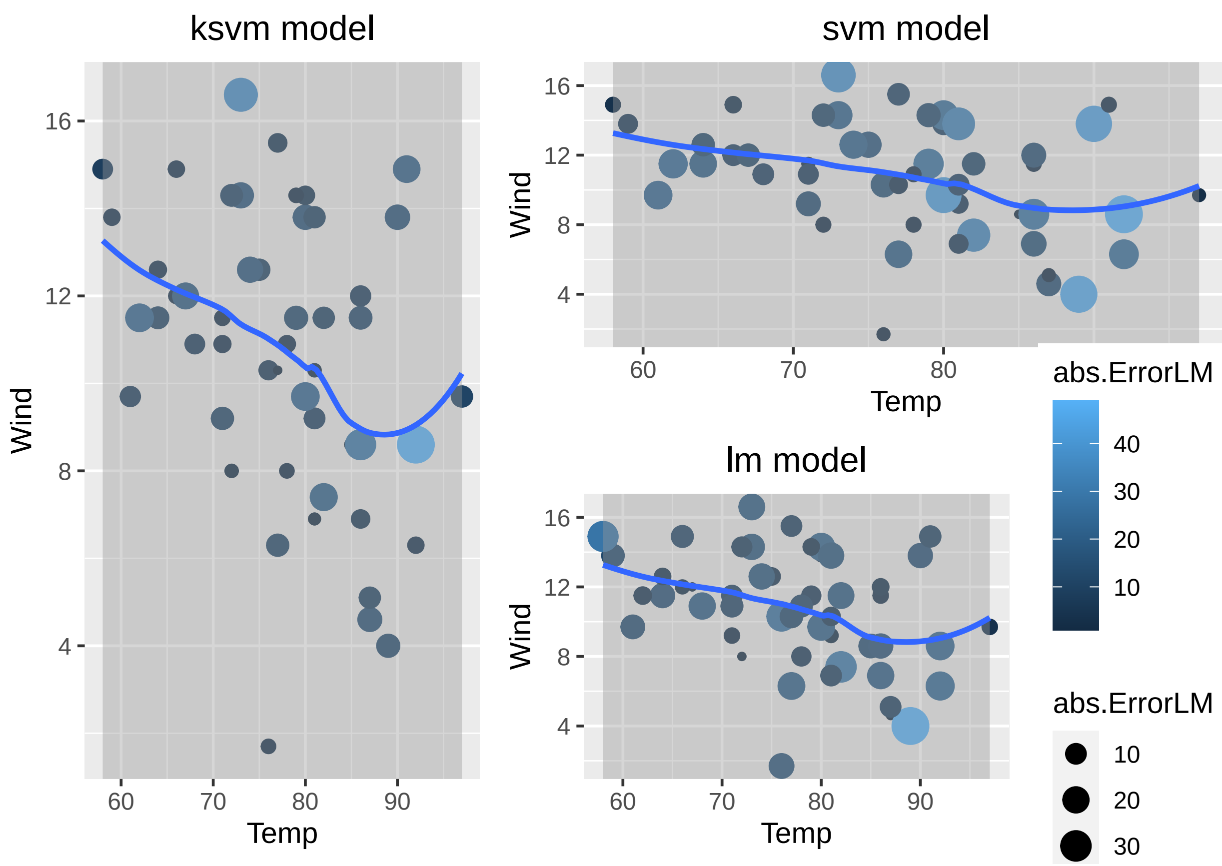
```r
#create a dataframe
plot.dfLM <- data.frame(airQ.test$Temp,airQ.test$Wind,abs.ErrorLM)
colnames(plot.dfLM) <- c("Temp","Wind","abs.Error")

ggscatterLM <- ggplot(plot.dfLM,aes(x= Temp,y=Wind))
ggscatterLM <- ggscatterLM + geom_point(aes(size=abs.ErrorLM,color=abs.ErrorLM)) + geom_smooth(meth

ggscatterLM + labs(backgroundColor="white",title="Ozone Value Predictions", caption = "Data: Based
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
lm_plot <- ggscatterLM + ggtitle("lm model") +theme
#------------------------------------------------------------------------------------------------------------

 # Consolidate all charts together

grid.arrange(ksvm_plot,svm_plot,lm_plot,widths = c(2,2,1), layout_matrix = rbind(c(1,2,2),c(1,3,3)))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# This variable should be either 0 or 1. It should be 0 if the ozone is below the average for all the d

    #compute the mean for Ozone

    ozone.mean <- round(as.numeric(mean(airQ$Ozone, na.rm=TRUE)),0)
    cat("mean value of Ozone variable is :",ozone.mean)
```

```
## mean value of Ozone variable is : 42
```

```
    airQ.train$goodOzone <- ifelse(airQ.train$Ozone < ozone.mean,0,1)  #create goodOzone variable on th
    airQ.test$goodOzone  <- ifelse(airQ.test$Ozone < ozone.mean,0,1) #create goodOzone variable on the

    airQ.train <- airQ.train[,-1]  # remove ozone variable from training dataset
    airQ.test <- airQ.test[,-1] # remove ozone variable from testing dataset

    # review colnames
    colnames(airQ.train)
```

```
## [1] "Solar.R"   "Wind"      "Temp"      "Month"     "Day"       "goodOzone"
```

```
    colnames(airQ.test)
```

```
## [1] "Solar.R"   "Wind"      "Temp"      "Month"     "Day"       "goodOzone"
```

```r
# airQ.train
# airQ.test

# convert to factor
airQ.train$goodOzone <- as.factor(airQ.train$goodOzone)
airQ.test$goodOzone <- as.factor(airQ.test$goodOzone)

# airQ.train
# airQ.test
```

```r
# 1) Build a model (using the 'ksvm' function, trying to predict 'goodOzone').
#    You can use all the possible attributes, or select the attributes that you think would be the most

    #predict Ozone values from all the variables in the train dataset

    goodKSVM <- ksvm(goodOzone~.,data=airQ.train,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.mo
    goodKSVM
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 10
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.191993806011052
##
## Number of Support Vectors : 65
##
## Objective Function Value : -314.0542
## Training error : 0.117647
## Cross validation error : 0.343636
## Probability model included.
```

```r
# 2) Test the model on the testing dataset, and compute the percent of 'goodOzone' that was correctly p

    goodKSVMpred <- predict(goodKSVM,airQ.test)
    str(goodKSVMpred)
```

```
##  Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 1 1 2 ...
```

```r
summary(goodKSVMpred)
```

```
##  0  1
## 25 26
```

```r
# create a dataframe to have actual and preicted Ozone values
goodmodel.df <- data.frame(airQ.test$goodOzone,goodKSVMpred)
colnames(goodmodel.df) <- c("Test.goodOzone","Pred.goodOzone")
head(goodmodel.df)
```

```
##   Test.goodOzone Pred.goodOzone
## 1              1              1
## 2              1              0
## 3              1              1
## 4              1              1
## 5              1              1
## 6              1              1
```

```r
    # Percentage of good predictions

    goodKSVM.Perc <- length(which(goodmodel.df$Test.goodOzone==goodmodel.df$Pred.goodOzone))/dim(goodmod
    cat("Percentage of good predictions from KSVM model is :",goodKSVM.Perc*100)
```

```
## Percentage of good predictions from KSVM model is : 74.5098
```

```r
# 3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis represent 

    goodmodel.df$Accuracy <- ifelse(goodmodel.df$Test.goodOzone==goodmodel.df$Pred.goodOzone,"good","bad
    goodmodel.df
```

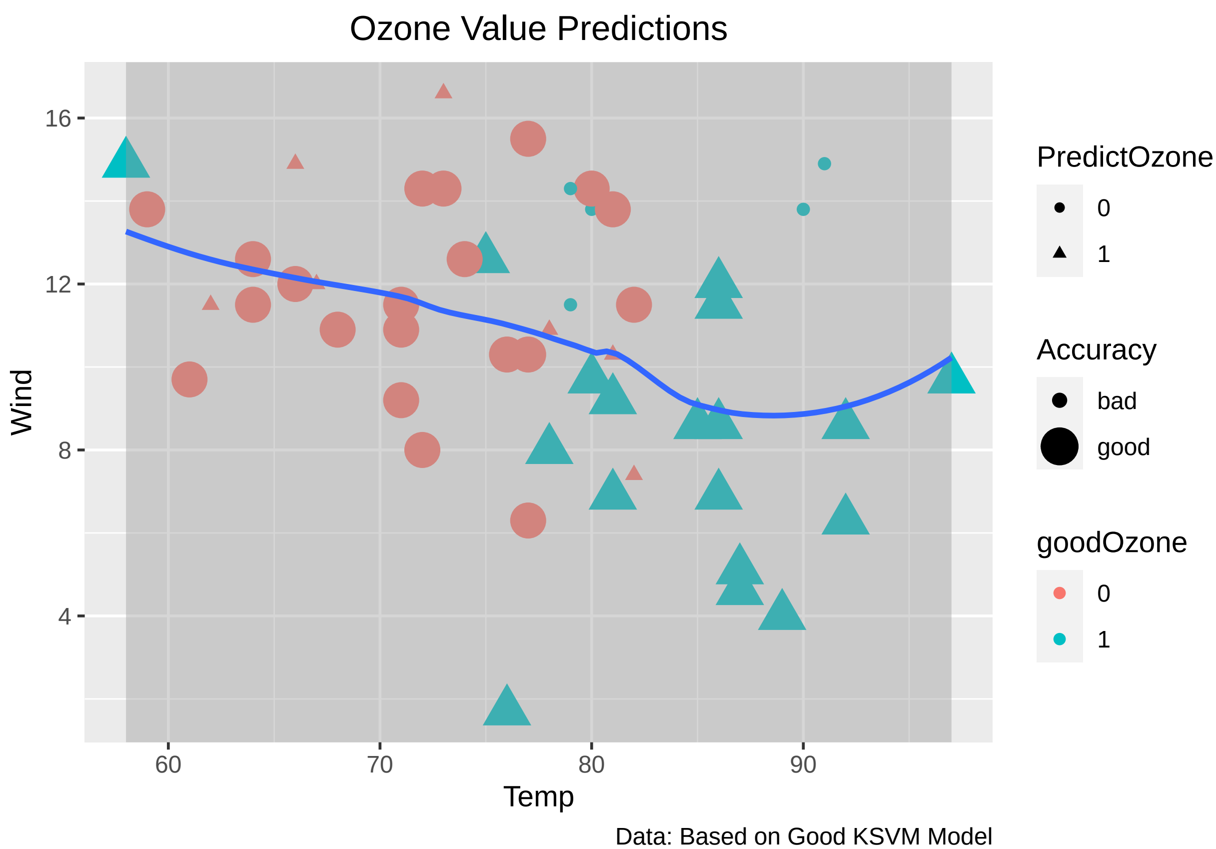
```
##    Test.goodOzone Pred.goodOzone Accuracy
## 1               1              1     good
## 2               1              0      bad
## 3               1              1     good
## 4               1              1     good
## 5               1              1     good
## 6               1              1     good
## 7               0              1      bad
## 8               0              0     good
## 9               1              0      bad
## 10              1              1     good
## 11              0              0     good
## 12              0              0     good
## 13              0              0     good
## 14              0              0     good
## 15              0              0     good
## 16              1              1     good
## 17              0              0     good
## 18              1              0      bad
## 19              0              0     good
## 20              0              0     good
## 21              1              1     good
## 22              0              0     good
## 23              0              1      bad
## 24              0              1      bad
## 25              0              0     good
## 26              0              0     good
## 27              1              1     good
## 28              0              0     good
## 29              0              1      bad
## 30              0              1      bad
## 31              0              0     good
## 32              0              0     good
```

```
## 33              0              0     good
## 34              1              1     good
## 35              0              0     good
## 36              1              0      bad
## 37              1              1     good
## 38              1              1     good
## 39              0              1      bad
## 40              0              0     good
## 41              0              1      bad
## 42              0              1      bad
## 43              1              1     good
## 44              0              0     good
## 45              0              0     good
## 46              1              1     good
## 47              1              1     good
## 48              1              1     good
## 49              1              0      bad
## 50              1              1     good
## 51              1              1     good
```

```r
#create a dataframe
goodplot.df <- data.frame(airQ.test$Temp,airQ.test$Wind,airQ.test$goodOzone,goodmodel.df$Pred.goodO
colnames(goodplot.df) <- c("Temp","Wind","goodOzone","PredictOzone","Accuracy")

goodKSVMplot <- ggplot(goodplot.df,aes(x= Temp,y=Wind))
goodKSVMplot <- goodKSVMplot + geom_point(aes(size=Accuracy,color=goodOzone,shape=PredictOzone)) + g

goodKSVMplot + labs(title="Ozone Value Predictions", caption = "Data: Based on Good KSVM Model")+th
```

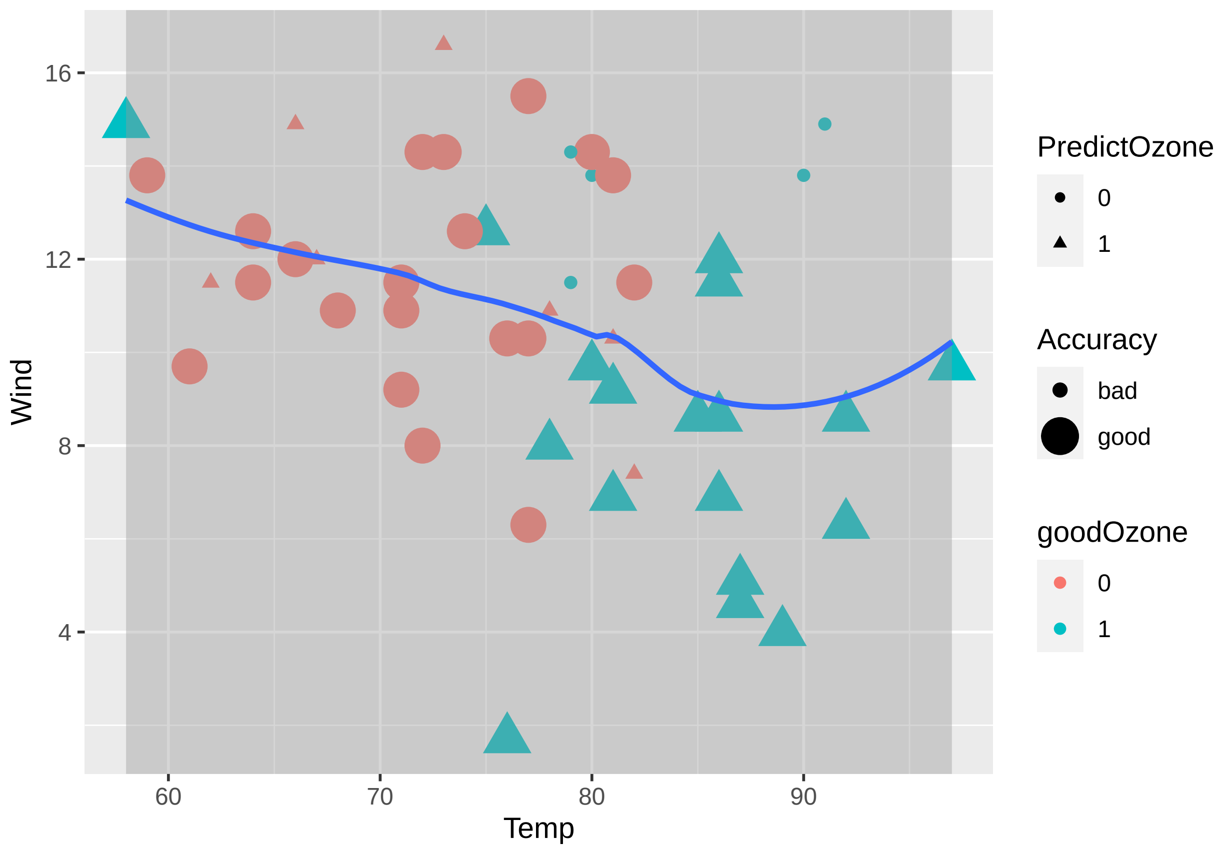
```
## Warning: Using size for a discrete variable is not advised.
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
goodKSVMplot
```

```
## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
    goodksvm_plot <- goodKSVMplot+theme(legend.position="none") + ggtitle("good ksvm model") +theme
    #------------------------------------------------------------------------------------------------

# 4) Compute models and plot the results for 'svm' (in the e1071 package) and 'nb' (Naive Bayes, also i

    #predict Ozone values from all the variables in the train dataset using SVM from e1071 package

    goodmodelSVM <- svm(goodOzone~.,data=airQ.train)
    goodmodelSVM
```

```
##
## Call:
## svm(formula = goodOzone ~ ., data = airQ.train)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  75
```

```
    goodpredSVM <- predict(goodmodelSVM,airQ.test)
    str(goodpredSVM)
```

```
##  Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 1 2 ...
##  - attr(*, "names")= chr [1:51] "92" "107" "102" "98" ...
```

```r
# create a dataframe to have actual and preicted Ozone values
goodmodelSVM.df <- data.frame(airQ.test$goodOzone,goodpredSVM)
colnames(goodmodelSVM.df) <- c("Test.goodOzone","Pred.goodOzone")
head(goodmodelSVM.df)
```

```
##     Test.goodOzone Pred.goodOzone
## 92               1              1
## 107              1              0
## 102              1              1
## 98               1              1
## 99               1              1
## 115              1              0
```

```r
# Percentage of good predictions

goodSVM.Perc <- length(which(goodmodelSVM.df$Test.goodOzone==goodmodelSVM.df$Pred.goodOzone))/dim(g
cat("Percentage of good predictions from SVM model is :",goodSVM.Perc*100)
```

```
## Percentage of good predictions from SVM model is : 80.39216
```

```r
goodmodelSVM.df$Accuracy <- ifelse(goodmodelSVM.df$Test.goodOzone==goodmodelSVM.df$Pred.goodOzone,"
goodmodelSVM.df
```

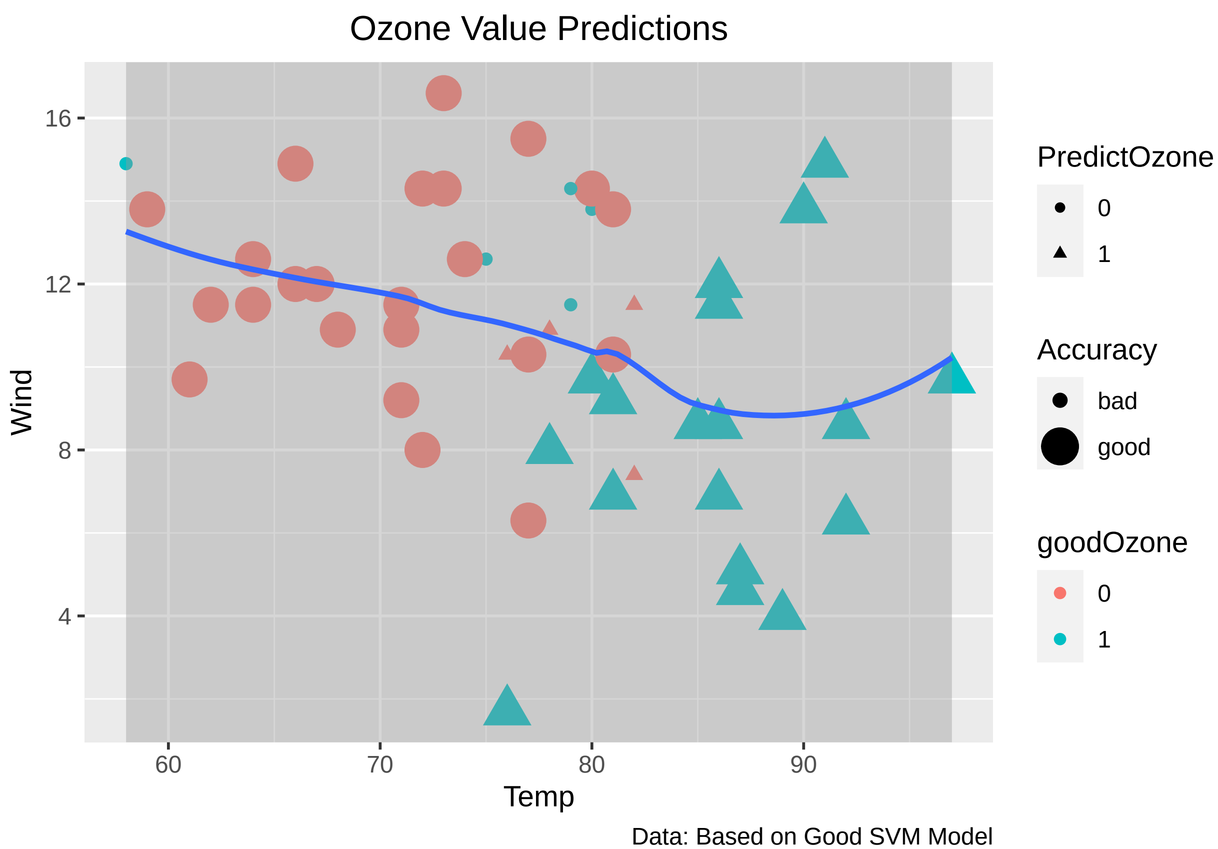
```
##     Test.goodOzone Pred.goodOzone Accuracy
## 92               1              1     good
## 107              1              0      bad
## 102              1              1     good
## 98               1              1     good
## 99               1              1     good
## 115              1              0      bad
## 6                0              0     good
## 16               0              0     good
## 45               1              0      bad
## 106              1              1     good
## 23               0              0     good
## 145              0              0     good
## 144              0              0     good
## 105              0              1      bad
## 73               0              0     good
## 36               1              1     good
## 76               0              0     good
## 37               1              0      bad
## 51               0              1      bad
## 113              0              0     good
## 85               1              1     good
## 114              0              0     good
## 4                0              0     good
## 111              0              1      bad
## 136              0              0     good
```

```
## 94           0           0    good
## 80           1           1    good
## 3            0           0    good
## 93           0           1     bad
## 146          0           0    good
## 138          0           0    good
## 17           0           0    good
## 2            0           0    good
## 103          1           1    good
## 14           0           0    good
## 40           1           1    good
## 53           1           1    good
## 120          1           1    good
## 22           0           0    good
## 8            0           0    good
## 95           0           1     bad
## 28           0           0    good
## 88           1           1    good
## 137          0           0    good
## 108          0           0    good
## 57           1           1    good
## 77           1           1    good
## 96           1           1    good
## 75           1           1    good
## 69           1           1    good
## 26           1           0     bad
```

```
#create a dataframe
goodplotSVM.df <- data.frame(airQ.test$Temp,airQ.test$Wind,airQ.test$goodOzone,goodmodelSVM.df$Pred
colnames(goodplotSVM.df) <- c("Temp","Wind","goodOzone","PredictOzone","Accuracy")

goodSVMplot <- ggplot(goodplotSVM.df,aes(x= Temp,y=Wind))
goodSVMplot <- goodSVMplot + geom_point(aes(size=Accuracy,color=goodOzone,shape=PredictOzone)) + ge

goodSVMplot + labs(title="Ozone Value Predictions", caption = "Data: Based on Good SVM Model")+them
```

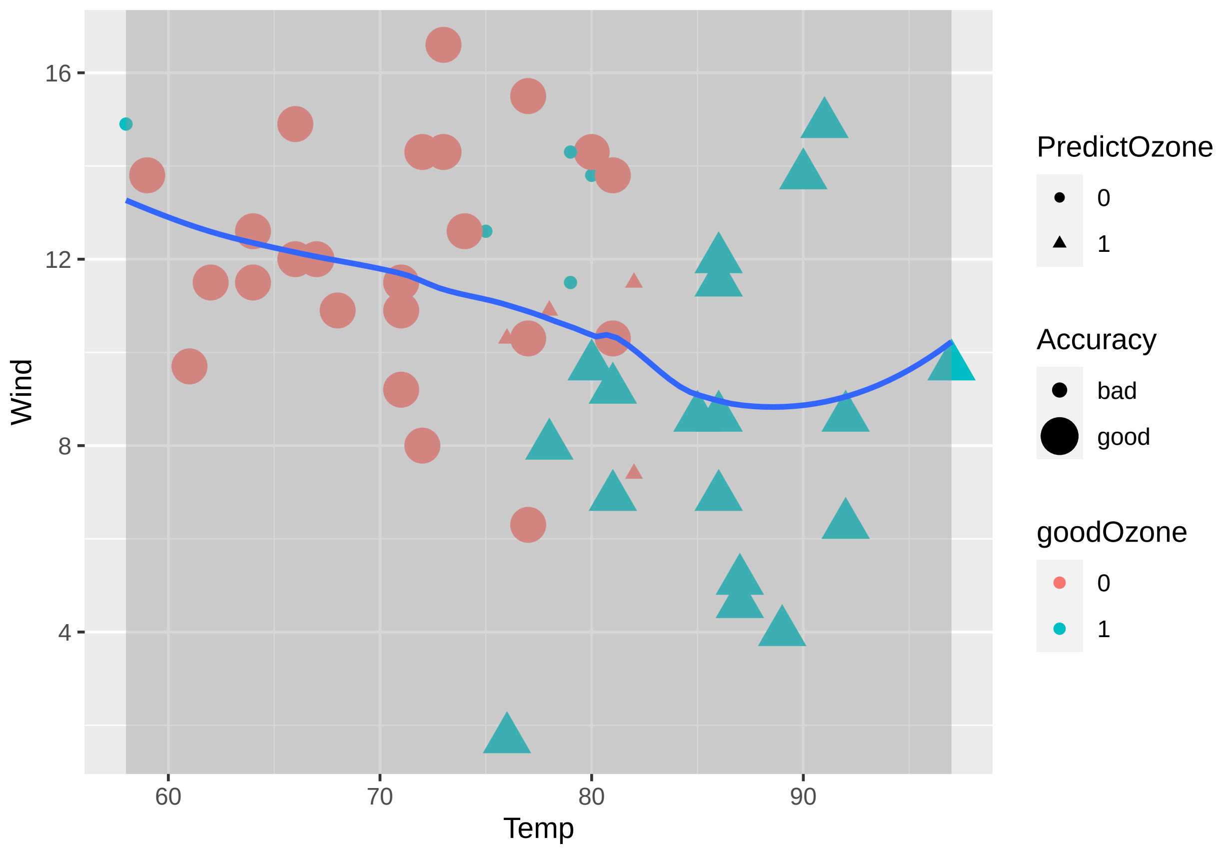
```
## Warning: Using size for a discrete variable is not advised.
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
goodSVMplot
```

```
## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
goodsvm_plot <- goodSVMplot+theme(legend.position="none") + ggtitle("good svm model") +theme

#------------------------------------------------------------------------------------------

#predict Ozone values from all the variables in the train dataset using naiveBayes (nb)

modelNB <- naiveBayes(goodOzone~., data=airQ.train)
modelNB
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.4313725 0.5686275
##
## Conditional probabilities:
##    Solar.R
## Y       [,1]     [,2]
##   0 171.7045 96.88847
##   1 200.2586 67.75046
##
```

```
##    Wind
## Y         [,1]     [,2]
##   0 11.084091 3.479638
##   1  8.531034 3.316160
##
##    Temp
## Y        [,1]     [,2]
##   0 72.84091 8.054879
##   1 82.13793 8.794832
##
##    Month
## Y        [,1]     [,2]
##   0 7.022727 1.704831
##   1 7.000000 1.213954
##
##    Day
## Y        [,1]      [,2]
##   0 15.86364  7.696544
##   1 16.55172 10.319752
```

```r
predNB <- predict(modelNB,airQ.test)
str(predNB)
```

```
##  Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 1 2 ...
```

```r
summary(predNB)
```

```
##  0  1
## 27 24
```

```r
head(predNB)
```

```
## [1] 1 0 1 1 1 0
## Levels: 0 1
```

```r
# create a dataframe to have actual and preicted Ozone values
modelNB.df <- data.frame(airQ.test$goodOzone,predNB)
colnames(modelNB.df) <- c("Test.goodOzone","Pred.goodOzone")
head(modelNB.df)
```

```
##   Test.goodOzone Pred.goodOzone
## 1              1              1
## 2              1              0
## 3              1              1
## 4              1              1
## 5              1              1
## 6              1              0
```

```r
# Percentage of good predictions

goodNB.Perc <- length(which(modelNB.df$Test.goodOzone==modelNB.df$Pred.goodOzone))/dim(modelNB.df)[
cat("Percentage of good predictions from NB model is :",goodNB.Perc*100)
```

```
## Percentage of good predictions from NB model is : 78.43137
```

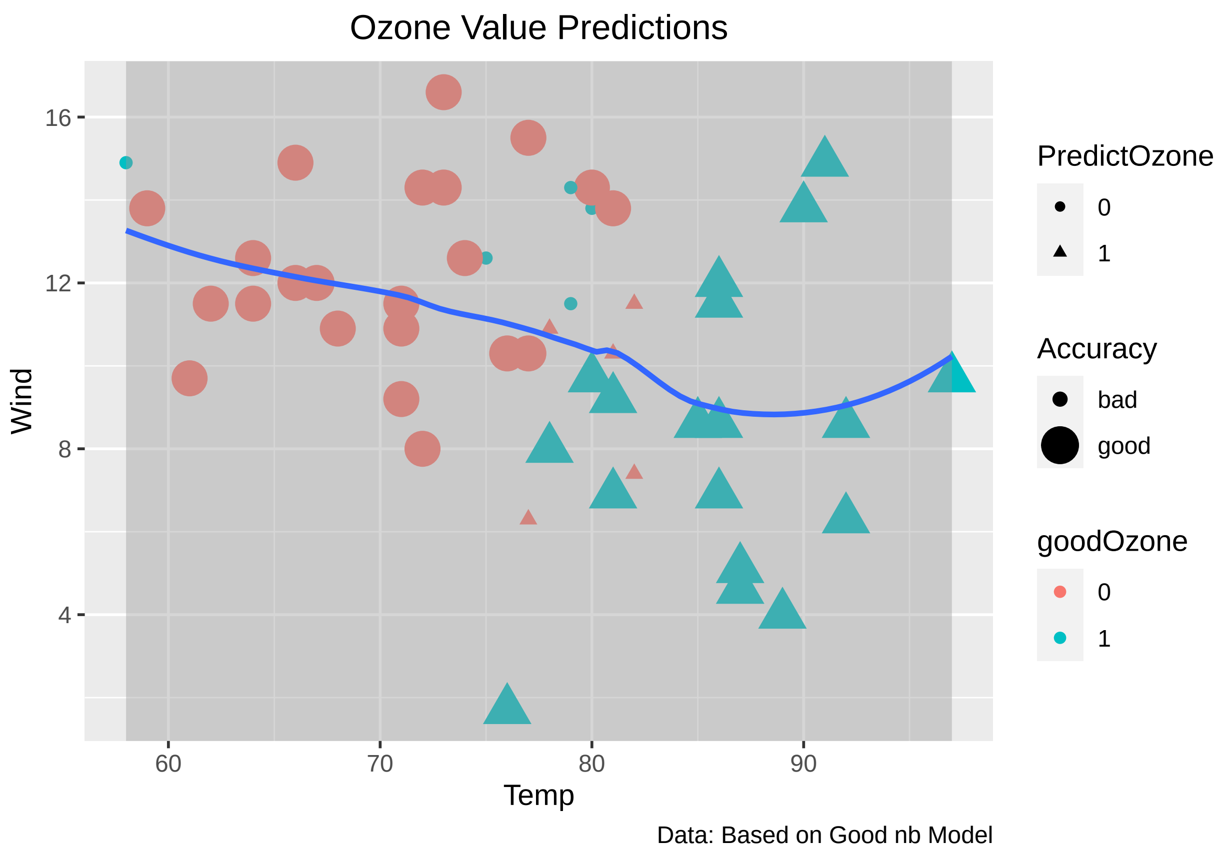
```r
modelNB.df$Accuracy <- ifelse(modelNB.df$Test.goodOzone==modelNB.df$Pred.goodOzone,"good","bad")
modelNB.df
```

```
##    Test.goodOzone Pred.goodOzone Accuracy
## 1               1              1     good
## 2               1              0      bad
## 3               1              1     good
## 4               1              1     good
## 5               1              1     good
## 6               1              0      bad
## 7               0              0     good
## 8               0              0     good
## 9               1              0      bad
## 10              1              1     good
## 11              0              0     good
## 12              0              0     good
## 13              0              0     good
## 14              0              1      bad
## 15              0              0     good
## 16              1              1     good
## 17              0              0     good
## 18              1              0      bad
## 19              0              0     good
## 20              0              0     good
## 21              1              1     good
## 22              0              0     good
## 23              0              0     good
## 24              0              1      bad
## 25              0              1      bad
## 26              0              0     good
## 27              1              1     good
## 28              0              0     good
## 29              0              1      bad
## 30              0              1      bad
## 31              0              0     good
## 32              0              0     good
## 33              0              0     good
## 34              1              1     good
## 35              0              0     good
## 36              1              1     good
## 37              1              1     good
## 38              1              1     good
## 39              0              0     good
## 40              0              0     good
## 41              0              1      bad
## 42              0              0     good
## 43              1              1     good
## 44              0              0     good
## 45              0              0     good
## 46              1              1     good
## 47              1              1     good
## 48              1              1     good
```

```
## 49                1               1      good
## 50                1               1      good
## 51                1               0       bad
```

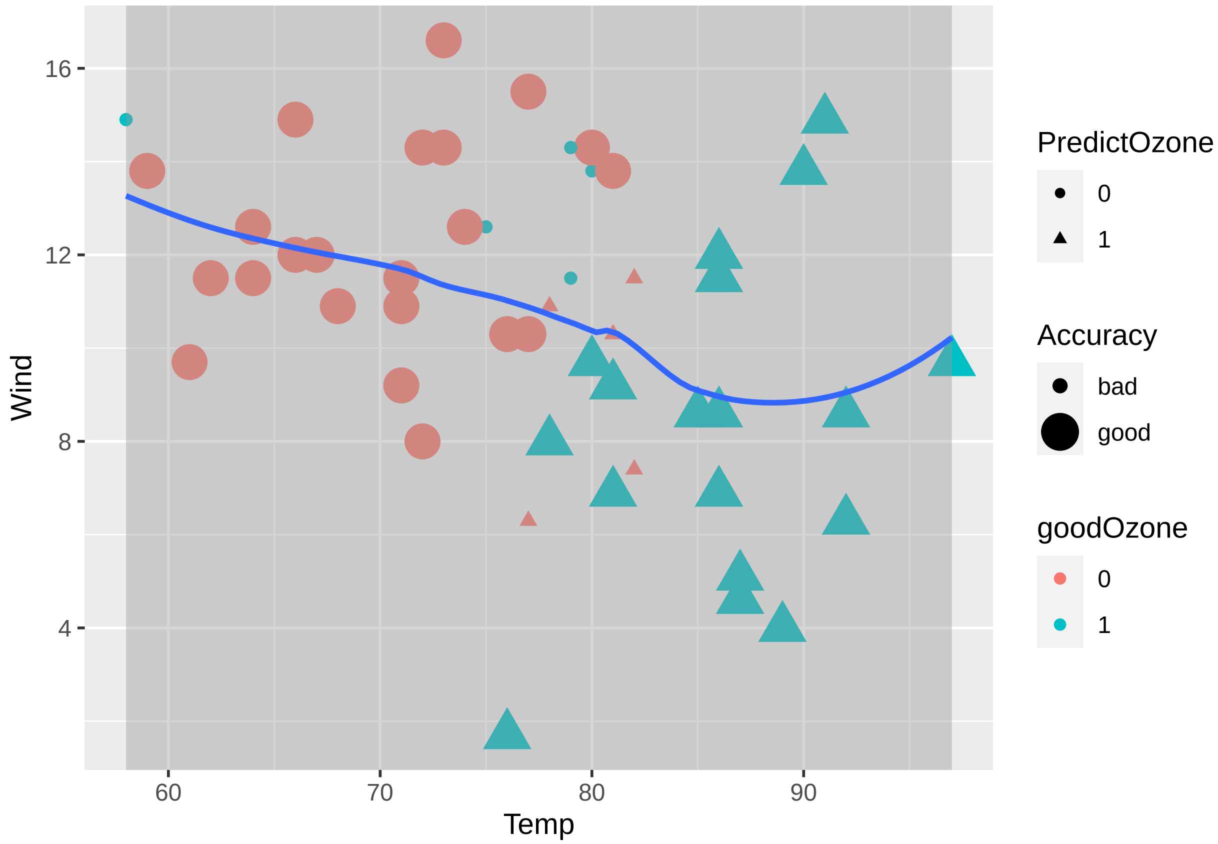
```r
#create a dataframe
goodplotNB.df <- data.frame(airQ.test$Temp,airQ.test$Wind,airQ.test$goodOzone,modelNB.df$Pred.goodOz
colnames(goodplotNB.df) <- c("Temp","Wind","goodOzone","PredictOzone","Accuracy")

goodNBplot <- ggplot(goodplotNB.df,aes(x= Temp,y=Wind))
goodNBplot <- goodNBplot + geom_point(aes(size=Accuracy,color=goodOzone,shape=PredictOzone)) + geom_

goodNBplot + labs(title="Ozone Value Predictions", caption = "Data: Based on Good nb Model")+theme
```

```
## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



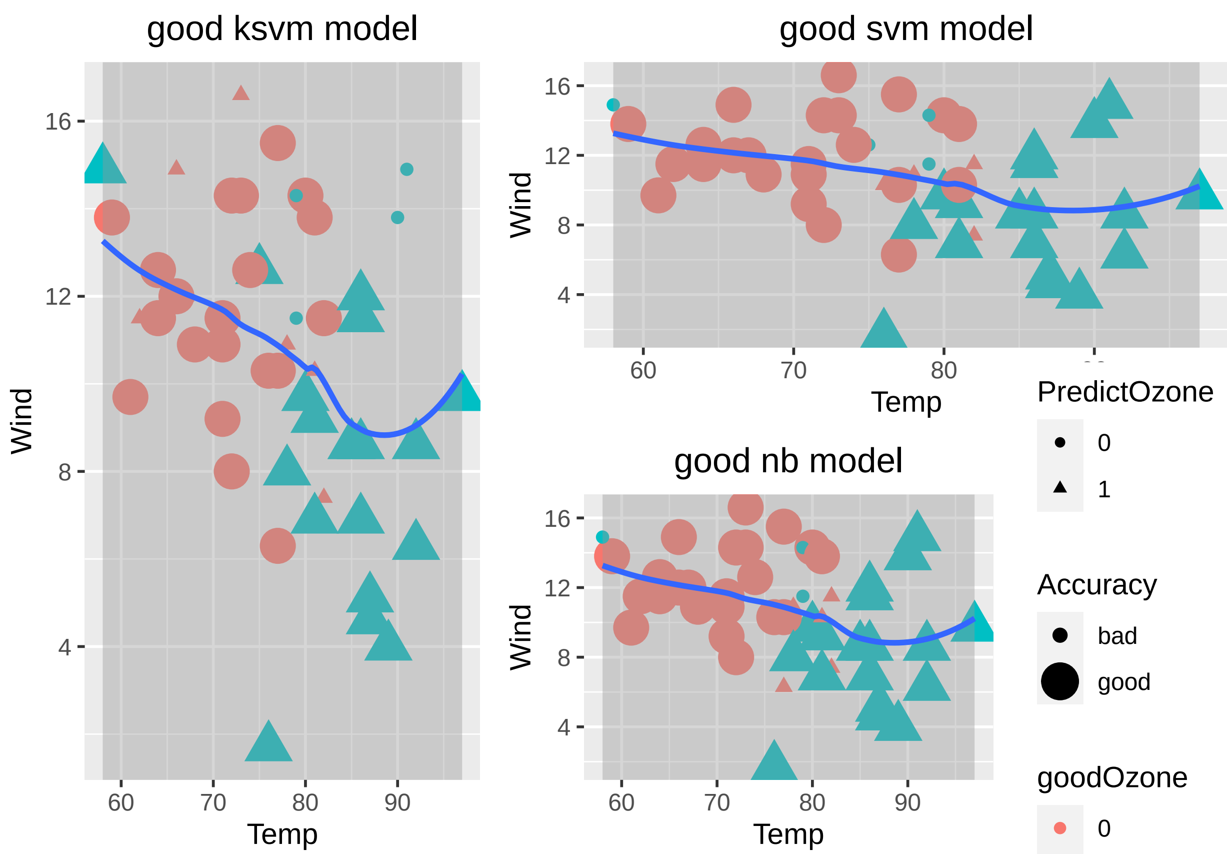

```r
goodNBplot
```

```
## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
goodNBplot <- goodNBplot+ ggtitle("good nb model") +theme#----------------------------------------

 # Consolidate all charts together

grid.arrange(goodksvm_plot,goodsvm_plot,goodNBplot,widths = c(2,2,1), layout_matrix = rbind(c(1,2,2
```

```
## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Using size for a discrete variable is not advised.

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
cat("Root Mean Squared Error from KSVM model is :",rmseKSVM)
```

```
## Root Mean Squared Error from KSVM model is : 18.22862
```

```
cat("\n Root Mean Squared Error from SVM model is :",rmseSVM)
```

```
##
##  Root Mean Squared Error from SVM model is : 15.15924
```

```
cat("\n Root Mean Squared Error from LM model is :",rmseLM)
```

```
##
##  Root Mean Squared Error from LM model is : 16.13633
```

```
cat("\n Percentage of good predictions from KSVM model is :",goodKSVM.Perc*100)
```

```
##
##  Percentage of good predictions from KSVM model is : 74.5098
```

```
cat("\n Percentage of good predictions from SVM model is :",goodSVM.Perc*100)
```

```
##
##  Percentage of good predictions from SVM model is : 80.39216
```

```r
cat("\n Percentage of good predictions from NB model is :",goodNB.Perc*100)
```

```
##
##  Percentage of good predictions from NB model is : 78.43137
```

```r
cat("\n based on RMSE variable - LM model has highest and SVM has the lowest score. So in this case
```

```
##
##  based on RMSE variable - LM model has highest and SVM has the lowest score. So in this case LM is re
```

```r
cat("\n based on Good Ozone variable - KSVM model has 78% accuracy and the naive bayes model has the
```

```
##
##  based on Good Ozone variable - KSVM model has 78% accuracy and the naive bayes model has the least a
```