HW9

Marley Akonnor

9/19/2021

Install Packages

```
library(kernlab)

library(ggplot2)
```

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

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

```
library('e1071')

library(gridExtra)
```

Step 1: Load the data

```
quality_air <-  airquality
```

Remove the NAs

```
quality_air <- na.omit(quality_air)
```

Step 2: Create train and test data sets first create a randomized index

```
randIndex <- sample(1:dim(quality_air)[1])
```

Training data

```
cut_point_2_3 <- floor(2 * dim(quality_air)[1]/3)
train_data <- quality_air[randIndex[1:cut_point_2_3], ]
```

Test data

```
test_data <- quality_air[randIndex[(cut_point_2_3+1):dim(quality_air)[1]],]
```

Step 3: Build a Model using KSVM & visualize the results

```
rmse <- function(error)
{
  sqrt(mean(error^2))
}
```

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 helpful.

```
svm_output <- ksvm(Ozone ~., data=train_data, kernel = "rbfdot", kpar="automatic", C=5, cross=3, prob.mo
```

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

```
svm_pred <- predict(svm_output, test_data, type = "votes")
```

Error

```
ozone_error <- test_data$Ozone - svm_pred
rmse(ozone_error)
```

```
## [1] 19.0898
```

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

```
plot_1 <- ggplot(test_data, aes(x = Temp, y = Wind)) + geom_point(aes(size = ozone_error, color = ozone_
```

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

```
svm_2 <- svm(Ozone ~., data=train_data)
```

Prediction

```
pred_svm_2 <- predict(svm_2, test_data)
pred_svm_2
```

```
##        68        41        86       105        40       129       146        87
## 83.462914 36.707936 68.291881 38.959290 38.641958 19.113165 30.583680 43.317350
##       123       113       149       136        76        38         2       140
## 83.891949 25.914274 26.801240 48.011941 18.497327 24.283076 17.467048 12.501448
##         9        30        13        12       138       127        20       145
## 16.586282 42.932067 16.311190 13.294565  9.033221 83.192441 10.371924 13.798333
##        78       133        64         1        70        19       153        69
## 42.957031 24.643019 40.128944 22.534228 89.573382 17.430271 18.699876 88.258779
##       116        85       130       134       122
## 42.384139 61.636455 33.638074 23.391597 85.391979
```

Error

```
svm_2_error <- test_data$Ozone - pred_svm_2
rmse(svm_2_error)
```

```
## [1] 17.76558
```

Plot

```
plot_2 <- ggplot(test_data, aes(x = Temp, y = Wind)) + geom_point(aes(size = svm_2_error, color = svm_2_
```

Linear Model

```
ozone_lm <- lm(Ozone ~ Solar.R + Wind + Temp, data = train_data)
pred_lm <- predict(ozone_lm, test_data, type = "response")
lm_error <- test_data$Ozone - pred_lm
```

Error

```
rmse(lm_error)
```

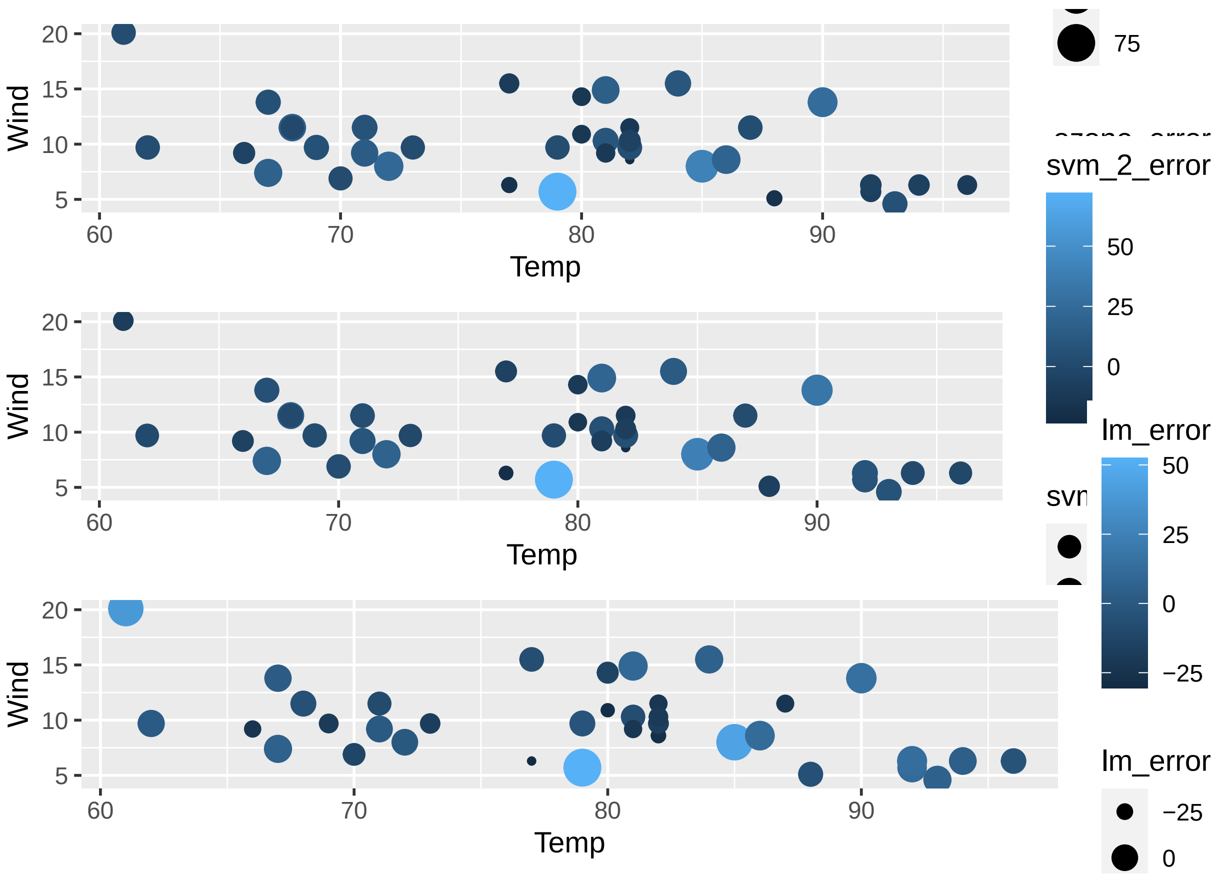
```
## [1] 19.1552
```

Plot linear model

```
plot_3 <- ggplot(test_data, aes(x = Temp, y = Wind)) + geom_point(aes(size = lm_error, color = lm_error)
```

Show all three results (charts) in one window, using the grid.arrange function

```
grid.arrange(plot_1, plot_2, plot_3)
```



Step 4: Create a 'goodOzone' variable This variable should be either 0 or 1. It should be 0 if the ozone is below the average for all the data observations, and 1 if it is equal to or above the average ozone observed.

```
avg_ozone <- mean(quality_air$Ozone)
```

Add new column

```
goodOzone <- c()

for (i in 1:length(quality_air$Ozone)){
  if(quality_air$Ozone[i] < avg_ozone )
  {
    goodOzone <- append(goodOzone, 0)
  }
  else
  {
    goodOzone <- append(goodOzone, 1)
  }
}
```

Create a new dataframe appending goodOzone

```
new_quality_air <- data.frame(quality_air, goodOzone)
```

Step 5: See if we can do a better job predicting 'good' and 'bad' days Create random indexes for training and test data

```
randIndex_2 <- sample(1:dim(new_quality_air)[1])
goodOZ_cutpoint <- floor(2 * dim(new_quality_air)[1]/3)
train_data_2 <- new_quality_air[randIndex_2[1:goodOZ_cutpoint], ]
test_data_2 <- new_quality_air[randIndex_2[(goodOZ_cutpoint+1):dim(new_quality_air)[1]],]
```

Build a model (using the 'ksvm' function, trying to predict 'goodOzone')

```
ksvm_output_goodOz <- ksvm(goodOzone ~., data=train_data_2, kernel = "rbfdot", kpar="automatic", C=5, c
```

Test the ksvm model

```
ksvm_pred_goodOz <- predict(ksvm_output_goodOz, test_data_2)
ksvm_pred_goodOz <- round(ksvm_pred_goodOz)
```

Error

```
goodOz_error <- test_data_2$goodOzone - ksvm_pred_goodOz
rmse(goodOz_error)
```

```
## [1] 0.2847474
```

goodOZ plot

```
plot_4 <- ggplot(test_data_2, aes(x = Temp, y = Wind)) + geom_point(aes(size = goodOz_error, color = go
```

Compute the percent of 'goodOzone' that was correctly predicted.

```
res_table <- table(ksvm_pred_goodOz, test_data_2$goodOzone)
accuracy <- (res_table[1,1]+res_table[2,2])/(res_table[1,1]+res_table[1,2]+res_table[2,1]+res_table[2,2
```

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

```
svm_3 <- svm(goodOzone ~., data=train_data_2)
```

Predict

```
pred_svm_3 <- predict(svm_3, test_data_2)
pred_svm_3 <- round(pred_svm_3)
```

Error

```
goodOz_error_2 <- test_data_2$goodOzone - pred_svm_3
rmse(goodOz_error_2)
```

```
## [1] 0.328798
```

Plot

```
plot_5 <- ggplot(test_data_2, aes(x = Temp, y = Wind)) + geom_point(aes(size = goodOz_error_2, color =
```

Accuracy

```
res_table_2 <- table(pred_svm_3, test_data_2$goodOzone)
accuracy <- (res_table_2[1,1]+res_table_2[2,2])/(res_table_2[1,1]+res_table_2[1,2]+res_table_2[2,1]+res
```

Naive Bayes Model

```
goodOz_nb <- naiveBayes(as.factor(goodOzone) ~ ., train_data_2)
pred_NB <- predict(goodOz_nb, test_data_2)
```

Error

```
goodOz_error_3 <- test_data_2$goodOzone - as.numeric(pred_NB)
rmse(goodOz_error_3)
```

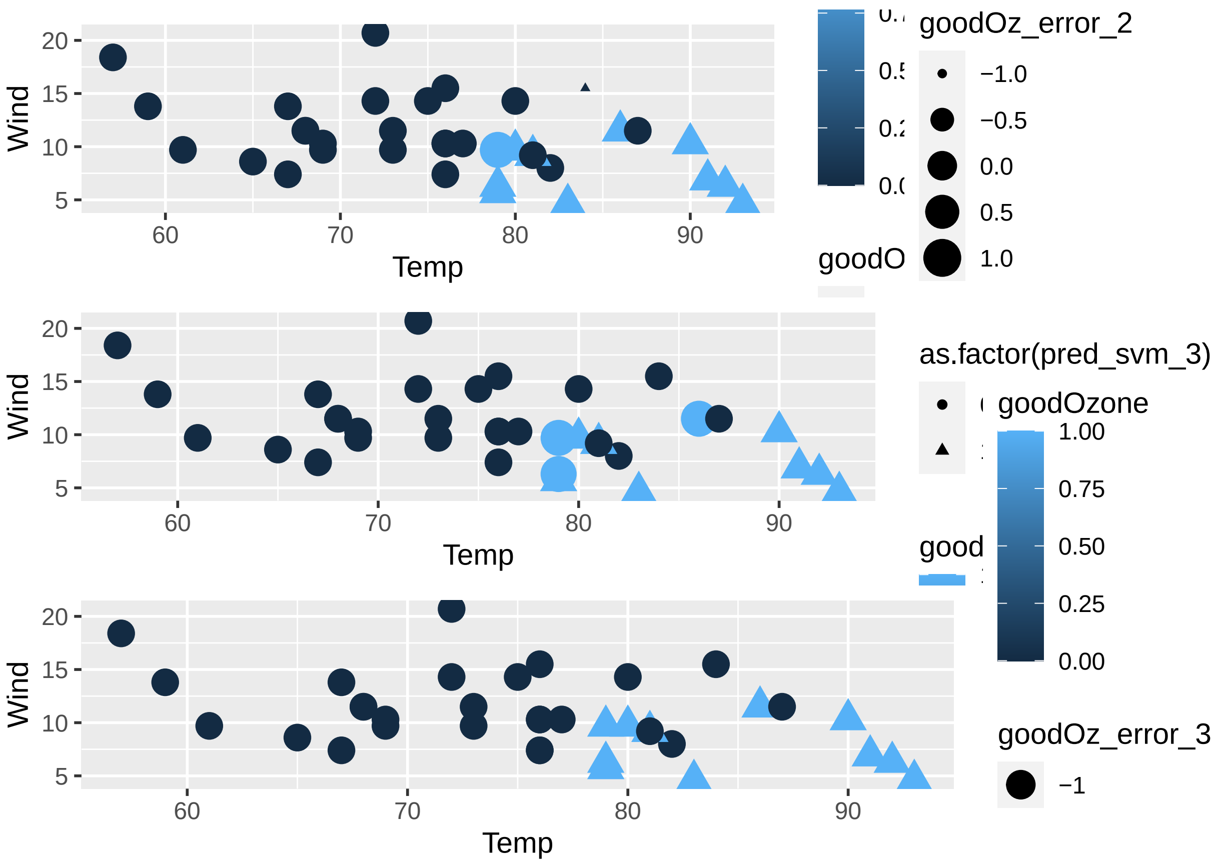
```
## [1] 1
```

NB plot

```
plot_6 = ggplot(data = test_data_2, aes(x = Temp, y = Wind)) + geom_point(aes(size = goodOz_error_3, co
```

All 3 in a grid

```
grid.arrange(plot_4, plot_5, plot_6)
```



Step 6: Which are the best Models for this data? The SVM was the best model for this based on the RSME scores. SVM was the lowest at 0.1