Samia Butt

1. KERNEL METHODS

Summary

In this report we will forcast wheather for stockholm location for the date 2016-07-02 and time from 4:00:00 to 24:00:00 with every two hours difference.

Gaussian and Distance Kernels

we are calculating three distances e.g. location, date and time and then calculating gaussian kernel by using distance kernels. After calculating the gaussian kernel we assign them as weight to each prior. Prior with the highest weight means the closest estimation for the desired point.

Gaussian Formula

$$K(d, h) = e^{-(d/h)^2}$$

Distance Kernel

We are using distHaversine function from geosphere package in r. This function takes longitude and latitude of the two locations and then calculates the distance between them.

Date Kernel

$$|desiredDate - predictionDate|$$

Time Kernel

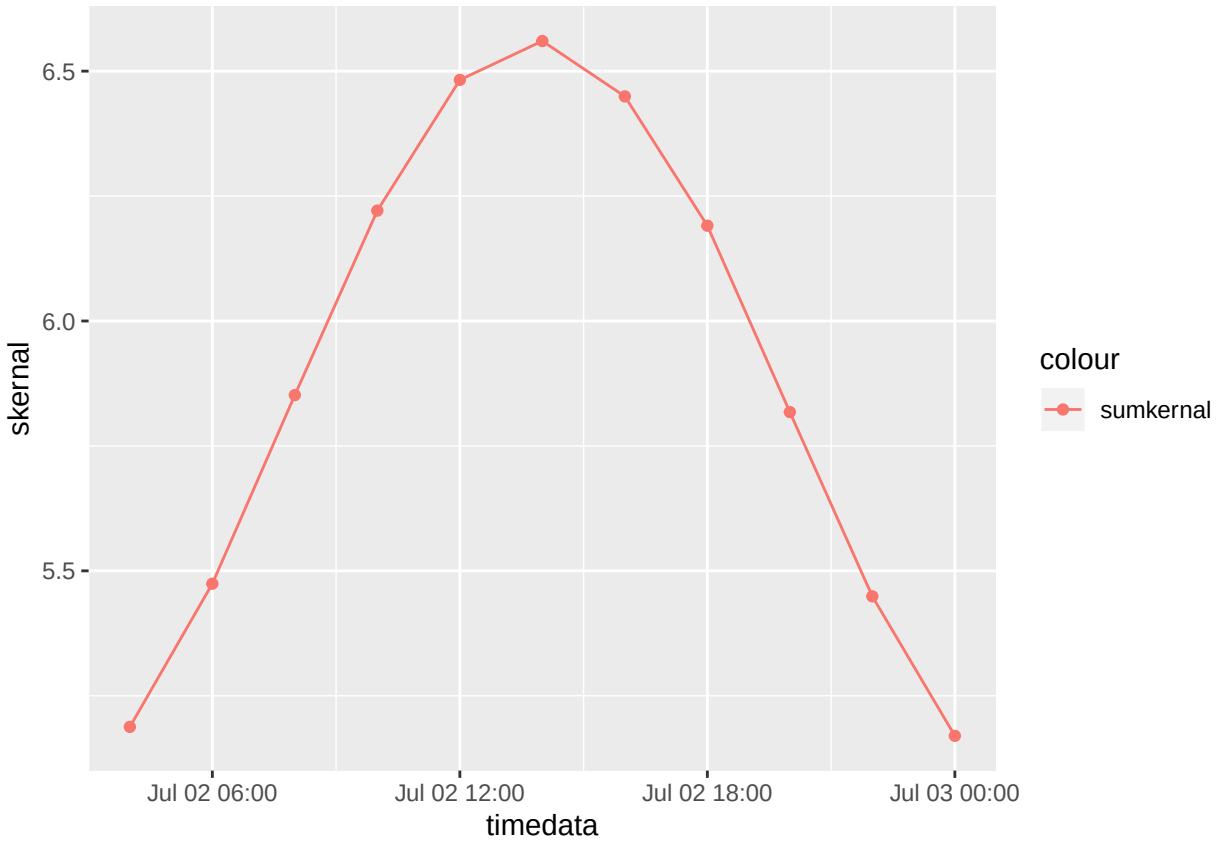
To calculate the time distance we have used difftime function of r.

Smoothing Parameters Selection

For the kernal's Smoothing Parameter selction we have to choose those values which have high weights.Our prediction date 2016-07-02 is very close the date (2016-07-01) which has highest points. In our case We have set h_distance to 20000, h_date to 14 and h_time to 7. We have selected these widths because the kernals have more weight with these widths.

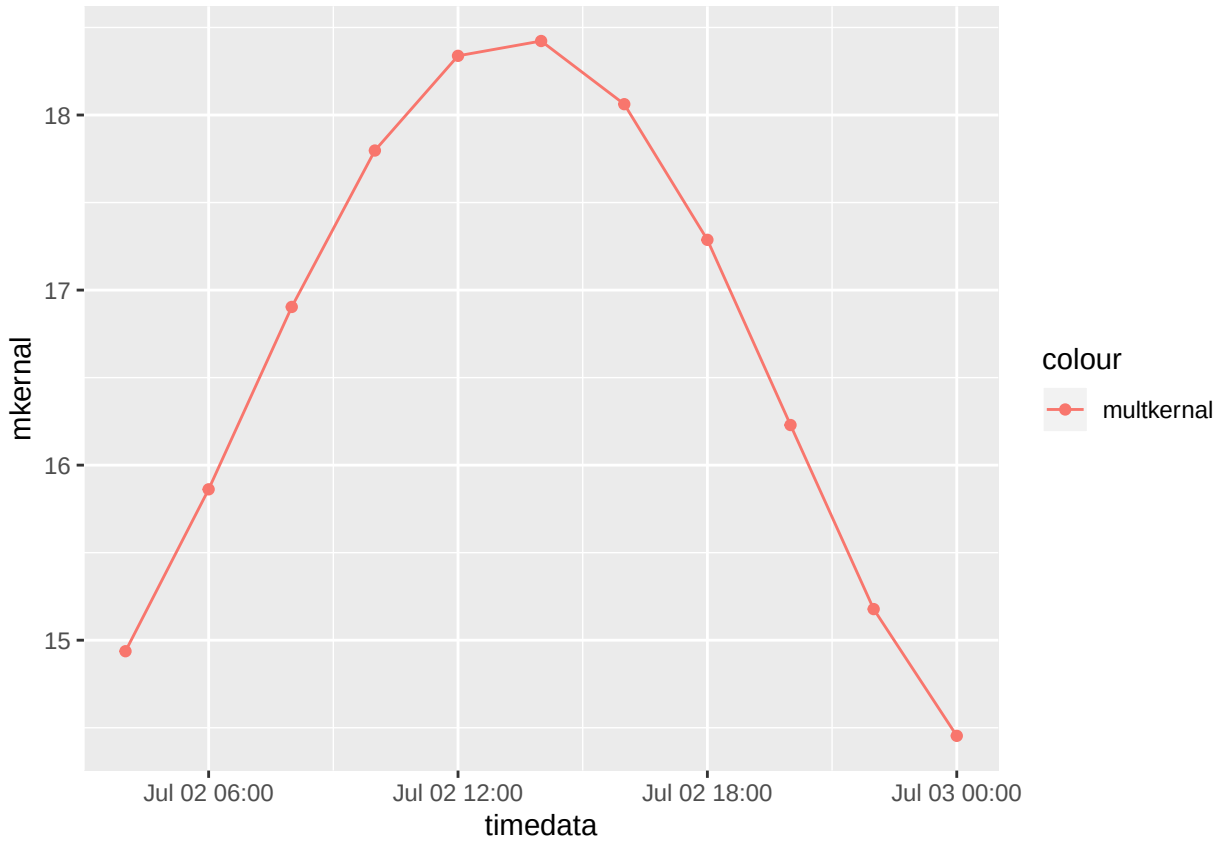
1.1 Summing up kernals

We have calculated weights by summing up location and date kernal with each time from 4 to 24 with 2 hours difference and then consdering that weight as our prediction for the desired date.



1.2 Multiply Kernals

We have calculated weights by multiplying location and date kernal with each time from 4 to 24 with 2 hours difference and then consdering that weight as our prediction for the desired date.



Conclusion

From the above graphs, we can easily conclude that the multiplicative kernal predicts reasonable temprature in summer as compare to additive kernal.

2. SUPPORT VECTOR MACHINES

2.1 validate SVM for C = 0.5, 1 and 5

We have devided our data into three parts, 60% for training, 20% for validation and 20% for testing. We will train our model on training data and predict using validation data and then we will choose best model and test it using test data.

SVM for C=0.5

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0269037613160921
##
## Number of Support Vectors : 1027
##
## Objective Function Value : -331.2972
## Training error : 0.060949
```

Confusion Matrix of C=0.5

```
##
## ksvpredict1 nonspam spam
##     nonspam     527   55
##     spam         19  330
```

Missclassification rate by using validation data for C=0.5 is 0.0794844

SVM for C=1

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0269735515287254
##
## Number of Support Vectors : 952
##
## Objective Function Value : -532.6789
## Training error : 0.052555
```

Confusion Matrix of C=1

```
##
## ksvpredict2 nonspam spam
##     nonspam     528   46
##     spam         18  339
```

Missclassification rate by using validation data for C=1 is 0.0687433

SVM for C=5

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0284605129550808
##
## Number of Support Vectors : 902
##
## Objective Function Value : -1547.935
## Training error : 0.031387
```

Confusion Matrix of C=5

```
##
## ksvpredict3 nonspam spam
##     nonspam     525   46
##     spam         21  339
```

Missclassification rate by using validation data for C=5 is 0.0719656

Comparison

Misclassification Comparision Table for Validation data:

c0.5	c1	c5
0.0794844	0.0687433	0.0719656

According to the above analysis by using validation data, C = 1 has the lowest missclassification rate so we conclude that the SVM model is the best model when the value of C is equal to 1.

2.3 Testing SVM for C=1.

As we have found in 2.2 that the misclassfication rate for C=1 is the lowest and is considered as best model so now we will test our model for C=1 by using testing data.

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0266797554397489
##
## Number of Support Vectors : 950
##
## Objective Function Value : -533.0084
## Training error : 0.052555
```

Confusion Matrix

```
##
## ksvpredict nonspam spam
##    nonspam     526   51
##    spam         14  339
```

Misclassification rate for when C=1 by using test data is 0.0698925.

Conclusion

Misclassification Comparision Table for validation and test data:

Test.c1	Validate.c1
0.0698925	0.0687433

Misclassification rate of validation model and test model has very slight difference it means our model works perfect when C=1.

2.4 Model for the customer

Model that should be returned to the customer.

```
ksvmodel2 =  ksvm(type~., data = train, C = 5 , kpca=list(sigma=0.05),type="C-svc")
```

2.5 C Parameter

C is a cost parameter in ksvm which penalize large residuals.Thats why the model with the larger cost has the less classification rate. So in other words we can say that the larger the cost the lower thr missclassification rate is.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

library(geosphere)
library(kernlab)
library(caret)
set.seed(1234567890)
stations <- read.csv("stations.csv",fileEncoding  = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

h_distance <- 25000 # These three values are up to the students
h_date <-14
h_time <- 7
# Latitude and longitude of stockholm
a <- 59.3293 # The point to predict (up to the students)
b <- 18.0686
date <- as.Date(x = "2016-07-02",format="%Y-%m-%d") # The date to predict (up to the students)

times <- c("4:00:00","6:00:00","8:00:00","10:00:00","12:00:00","14:00:00",
           "16:00:00","18:00:00","20:00:00","22:00:00","24:00:00")
temp <- vector(length=length(times))


timeval <- vector()
difference <- sapply(1:nrow(st), function(i){

  distval = distHaversine(c(b, a), c(st$longitude[i], st$latitude[i]))

  st$distgaus[i] <<- exp(-(distval/(h_distance))^2)

  dateval = as.numeric(strftime(date,format = "%j")) - as.numeric(strftime(as.Date(st$date[i]),format =

  dateval = sapply(dateval, function(dat){ return(min(abs(dat),365 - abs(dat))) })

  st$dategaus[i] <<- exp(-(as.numeric(dateval)/(h_date))^2)

  timespam = as.numeric(as.POSIXct(paste(date,times)))
  currentTime = as.numeric(as.POSIXct(paste(date,st$time[i])))

  timeval = timespam-currentTime
  timeval = timeval/3600
  timeval = sapply(timeval, function(t){return( min(abs(t),24-abs(t)))})
  # strptime(paste(date,st$time[i]), format = "%Y-%m-%d%H:%M:%S"),units="hour")
  #
  # timeval = ifelse(is.na(timeval) == TRUE,0,timeval)

  st[i,times] <<-  unlist(exp(-(timeval/(h_time))^2))

})

sumweight = st[times]+st$distgaus+st$dategaus
```

```r
sumkernal = colSums(sumweight*st$air_temperature)/colSums(sumweight)

multweight = st[times]*(st$distgaus*st$dategaus)

multkernal = colSums(multweight*st$air_temperature)/colSums(multweight)

plotdata = data.frame(skernal = sumkernal, mkernal=multkernal,  timedata = times)

plotdata$timedata <- as.POSIXct(strptime(paste(date,plotdata$timedata), format="%Y-%m-%d%H:%M:%S"))

ggplot(plotdata)+geom_line(aes(x=timedata , y = skernal,group = 1,col = "sumkernal"))+
  geom_point(aes(x=timedata , y = skernal,group = 1,col = "sumkernal"))

ggplot(plotdata)+geom_line(aes(x=timedata , y = mkernal,group = 2,col = "multkernal"))+
  geom_point(aes(x=timedata , y = mkernal,group = 2,col = "multkernal"))
data(spam)
data2 = spam

RNGkind(sample.kind = "Rounding")
set.seed(12345)
# n = nrow(data2)
# id=sample(1:n, floor(n*0.7))
# train=data2[id,]
# test=data2[-id,]

idx <- sample(seq(1, 3), size = nrow(data2), replace = TRUE, prob = c(.6, .2, .2))
train <- data2[idx == 1,]
valid <- data2[idx == 2,]
test <- data2[idx == 3,]
ksvmodel1 =  ksvm(type~., data = train, C = 0.5 , kpca=list(sigma=0.05))
ksvpredict1 = predict(ksvmodel1,newdata = valid)
misclassificationRate1 = mean(ksvpredict1 != valid$type)
ksvmodel1
table(ksvpredict1, valid$type)
ksvmodel2 =  ksvm(type~., data = train, C = 1 , kpca=list(sigma=0.05))
ksvpredict2 = predict(ksvmodel2,newdata = valid)
misclassificationRate2 = mean(ksvpredict2 != valid$type)
ksvmodel2
table(ksvpredict2, valid$type)

ksvmodel3 =  ksvm(type~., data = train, C = 5 , kpca=list(sigma=0.05))
ksvpredict3 = predict(ksvmodel3,newdata = valid)
misclassificationRate3 = mean(ksvpredict3 != valid$type)
ksvmodel3

table(ksvpredict3, valid$type)
kableExtra::kable(data.frame(c0.5=misclassificationRate1,c1=misclassificationRate2,c5=misclassificationR
ksvmodel =  ksvm(type~., data = train, C = 1 , kpca=list(sigma=0.05))
ksvpredict = predict(ksvmodel,newdata = test)
misclassificationRate = mean(ksvpredict != test$type)
ksvmodel
table(ksvpredict, test$type)
kableExtra::kable(data.frame(Test.c1=misclassificationRate,Validate.c1=misclassificationRate2))
```

```
ksvmodel2 =  ksvm(type~., data = train, C = 5 , kpca=list(sigma=0.05),type="C-svc")

``
```