KERNEL METHODS AND NEURAL NETWORKS

Rakhshanda Jabeen

18/04/2020

Contents

1. Kernel Methods (Wheather Forecast)

In this task we are implementing the kernel method to forecast wheather for a location of Linköping, Sweden. We will forecast weather for different seasons at time 4:00:00 to 24:00:00 in an interval of 2 hours.

1.1. The Guassian Kernel

The Guassian kernel has the form,

$$k(x, x') = \exp\left(\frac{-||x - x'||^2}{\sigma^2}\right)$$

In this context it is not interpreted as probability density, that is the reason the normalization coefficient is omitted. In this case σ^2 act as the bandwidhth for the kernal density estimate usually denoted by h.

1.1.1. Location Distance Kernel

In this task we are using Haversine to calculate distance between two location instread of Euclidean distance. We are calculating distance kernal between two location using distHaversine function of R package geosphere. This function take two parameters longitude and latitude of the location and calculate the difeerence between them.

$$d = 2r * sin^{-1}\left(\sqrt{sin^2\big(\frac{|\phi_2 - \phi_1|}{2}\big) + cos(\phi_1)\ cos(\phi_2)\ sin^2\big(\frac{|\lambda_2 - \lambda_1|}{2}\big)}\right)$$

Where,

- ϕ_1, ϕ_2 is the latitude of point 1 and 2 respectively.
- λ_1, λ_2 is the longitude of point 1 and 2 respectively.
- r is the radius of the sphere.

1.1.2. Date Distance Kernel

We are calculating date distance by taking minimum of the absolute difference of all the dates in data and the desired estimation day.

$$d = |date_{observed} - date_{prediction}|$$

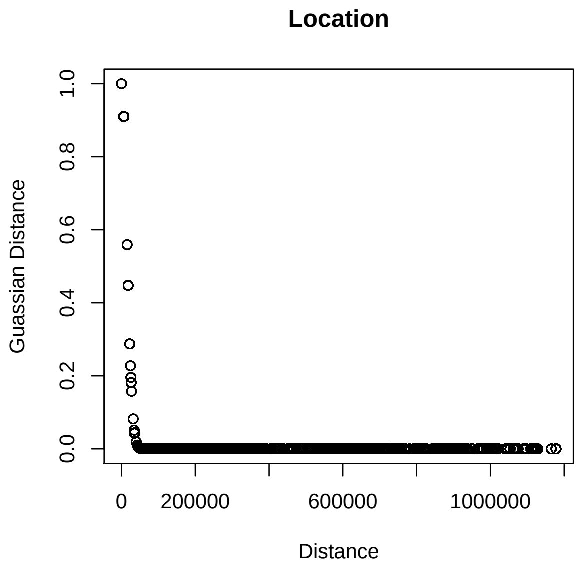
1.1.3. Time Distance Kernel

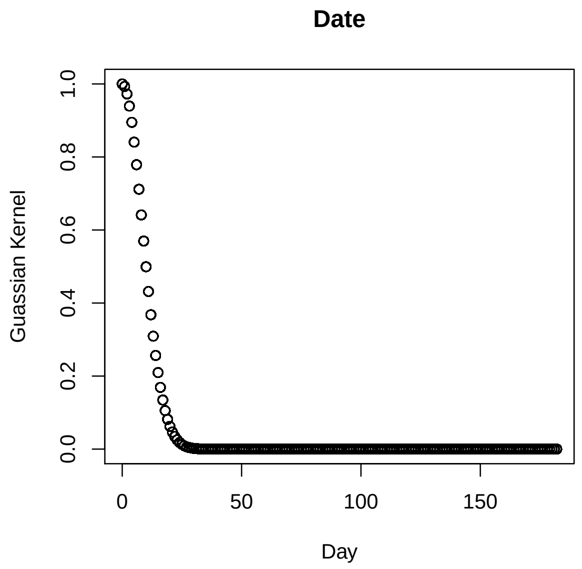
The time distance between the observed data time and desired approximation date is in hours as unit.

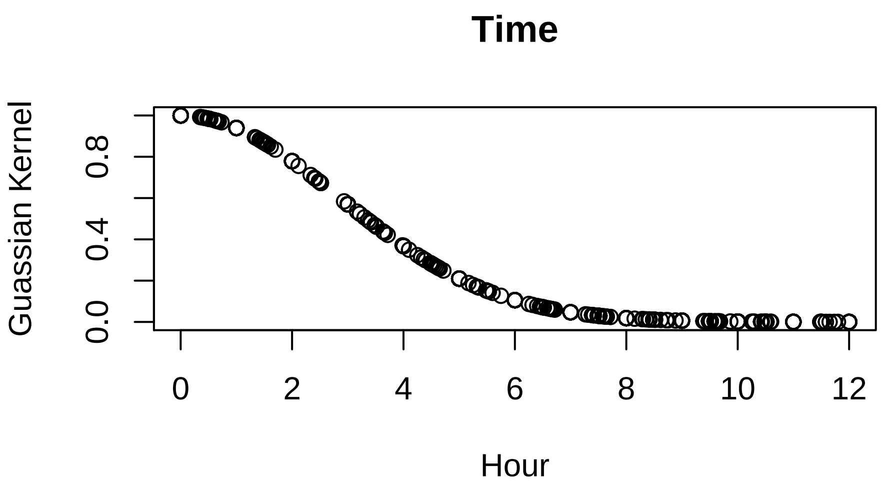
$$d = |time_{observed} - time_{prediction}|$$

1.2. Smoothing Coefficients / Width of the Kernel

- $h_{distance} = 20000$: By chosing this width we are putting more weight on the stations which are in the 20km range of the desired location for which we are forecasting weather.
- $h_{date} = 12$: By chosing this width we are putting more weight to the 12 days around the desired prediction date.
- $h_{time} = 4$: Since we are predicting the temperature of whole day at different time. Thus by chosing this width we are putting more weight on the 4 hours around the desired prediction time.







Sum of Kernels

The first mixture kernel is derived by the sum of all three kernels.

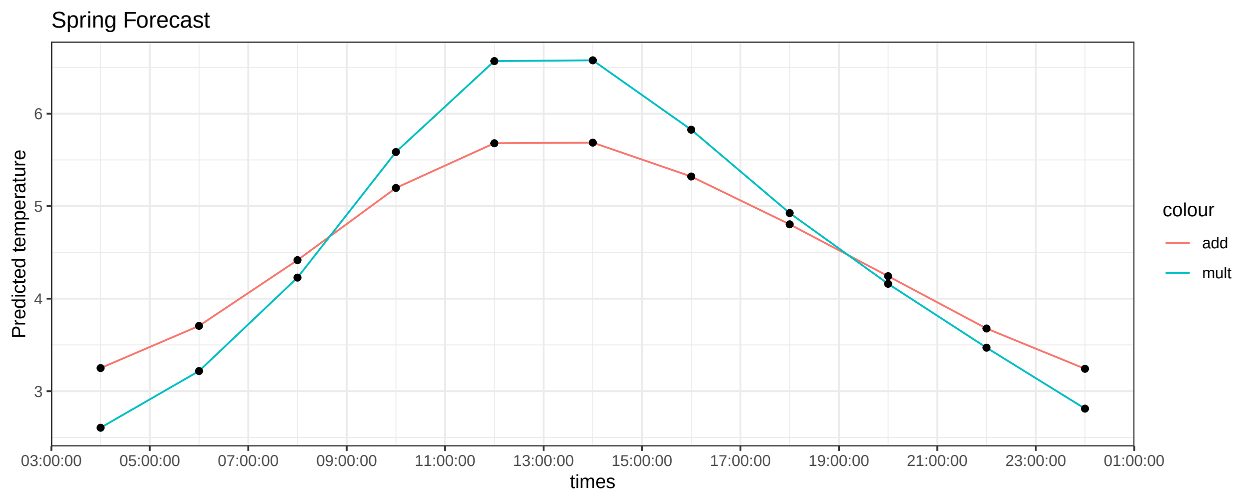
$$\hat{T} = \frac{\sum T * \Big(k(x,x')_{distance} + k(x,x')_{date} + k(x,x')_{time}\Big)}{\sum k(x,x')_{distance} + k(x,x')_{date} + k(x,x')_{time}}$$

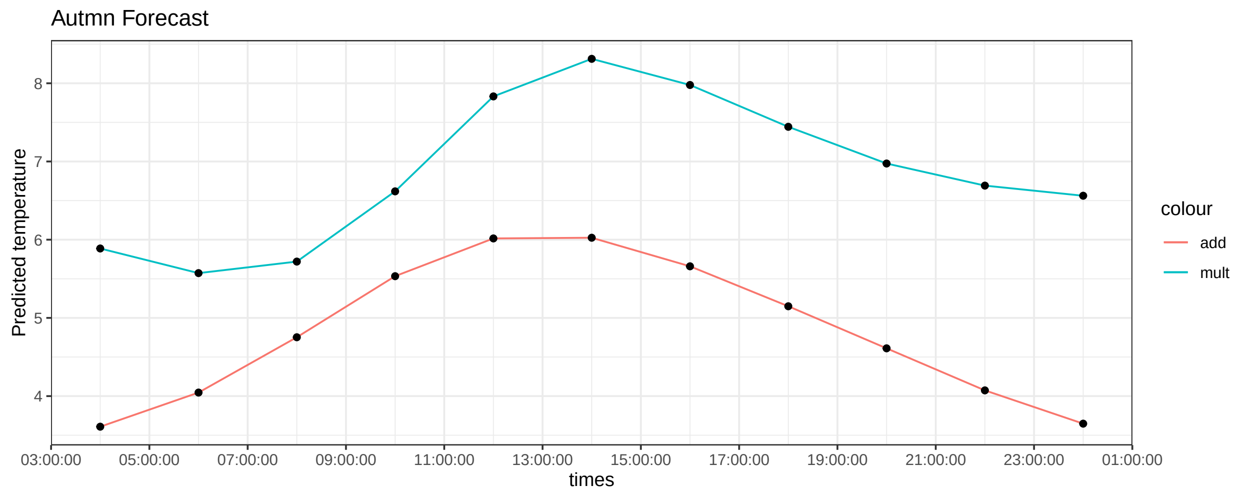
Product of Kernels

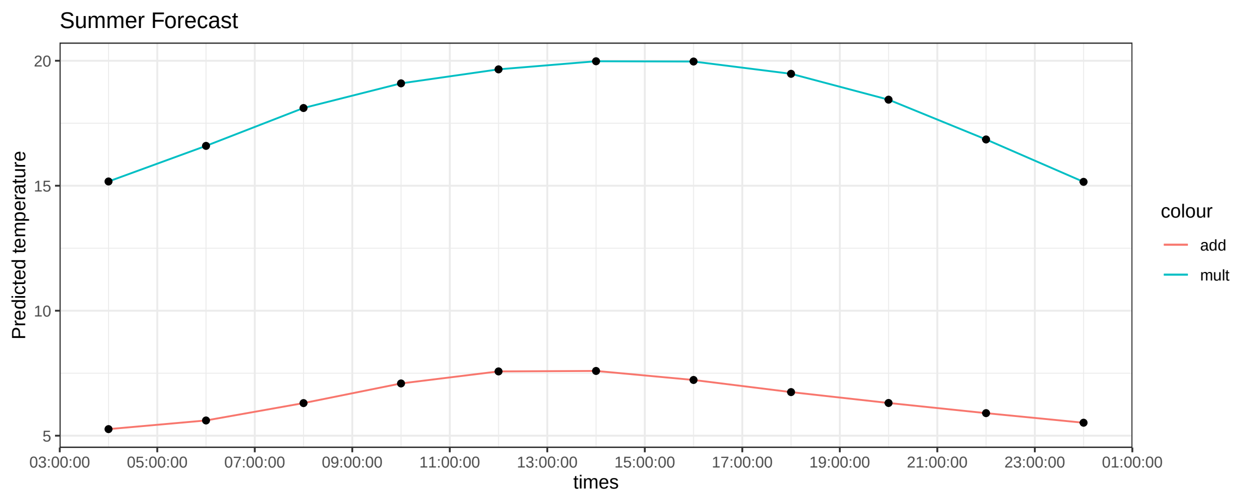
The second mixture kernel is derived by the product of all three kernels.

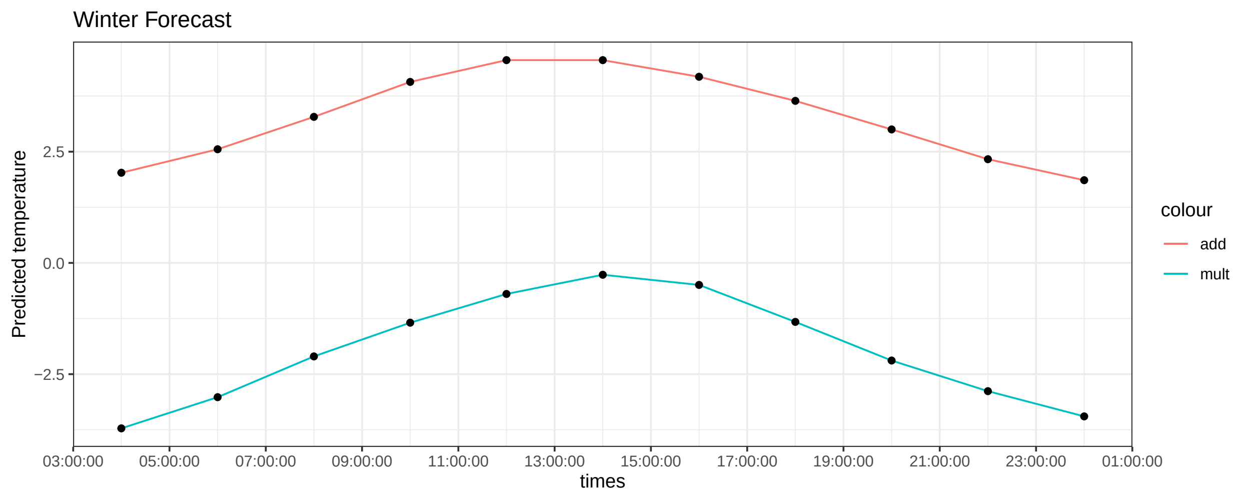
$$\hat{T} = \frac{\sum T * \Big(k(x,x')_{distance} * k(x,x')_{date} * k(x,x')_{time}\Big)}{\sum k(x,x')_{distance} * k(x,x')_{date} * k(x,x')_{time}}$$

1.3. Linköping Weather Forecast









Conclusion

By looking at all these weather predictions we can say that Multiplication of the kernels is doing a better prediction of weather as compare to the sum of te kernels. The sum kernel predictions does not vary much in all four seasons.
In sum of kernels, if a station is given a very small weight nearly equal to zero by one of the kernels but a higher weiht is assigned by any of the two other kernels then the effect of this station does not lead to zero and it still has great impact on the prediction. On the other hand in Product of kernels, if a station got very small weight nearly equal to zero by one kernel then after multiplying with other kernels it tends to zero or a very small quantity then this station would not have a significant impact on the prediction. That is why product of kernels is giving better weather predictions whilst sum of kernels is almost predicting the average temperature of the desired station.

2. SUPPORT VECTOR MACHINES

Support Vector Machine is a supervised machine learning algorithm that can be used to solve both classification and regression problems.

In this task, we are using the SVM model on a classification problem where we have to classify spam and non-spam emails using 57 features. We have divided data into train and test in the ratio of 70%-30%.

2.1 Model Selection

We are considering the radial basis function kernel (also known as Gaussian) with a width of 0.05 (radius of the area of influence of the support vectors). For three different values of cost parameter (C). We are dividing our data into three parts training, validation and testing data in the ratio of 50%, 25% and 25% respectively.

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 =  0.05
##
## Number of Support Vectors : 1079
##
## Objective Function Value : -313.6978
## Training error : 0.055628
```

As we can see in this model 1079 data points lie closest to the decision boundary. That means our model is unable to classify 1079 data points accurately. The following table represents the confusion matrix when we use our odel on validation data.

Table 1: Confusion Matrix

	nonspam	spam
nonspam	661	80
spam	21	388

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 =  0.05
##
```

```
## Number of Support Vectors : 1011
##
## Objective Function Value : -468.197
## Training error : 0.039548
```

As we can see in this model 1011 data points lie closest to the decision boundary. That means our model is unable to classify 1011 data points accurately. Trainng error of this model is lesser than the previous model. The following table represents the confusion matrix when we use our odel on validation data.

Table 2: Confusion Matrix

	nonspam	spam
nonspam	660	64
spam	22	404

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 =  0.05
##
## Number of Support Vectors : 962
##
## Objective Function Value : -1109.859
## Training error : 0.021295
```

As we can see in this model 962 data points lie closest to the decision boundary. Trainng error of this model is some how close to the previous model. The following table represents the confusion matrix when we use our odel on validation data.

Table 3: Confusion Matrix

	nonspam	spam
nonspam	657	65
spam	25	403

Comparison

Table 4: Accuracy

C = 0.5	C = 1	C = 5
91.22	92.52	92.17

Hyperparameter sigma is not very high in all the models which indicates that the decision boundary tends to be strict and sharp for all the models. The table shown above represents the accuracy of all the models for different values of cost parameter C. Here we will chose the model with cost paramter $C = 1$ as it has maximun accuracy level on validation data.

2.2 Generalization Error Of Model

The generalization error is calculated on the test dataset. Even though all the three models have high accuracy for classifying spam, but we have chosen the model with $C = 1$ because it can control the bias-variance tradeoff properly. Because choosing a very small amount of C can lead us to very soft margins and a large amount of C can result in very hard margins. Error rates of the selected model is 7.48.

The table shown below represents the confusion matrix of our fitted model when tested on the training dataset.

Table 5: Confusion Matrix

	nonspam	spam
nonspam	663	58
spam	32	397

Thus generalize error of model is 7.83.

Thus SVM model for users is shown in the following *R* code.

```
svm_model2= ksvm(type~., data = spam, kernel = rbfdot(sigma=0.05), C = 1)
```

2.3 Cost Parameter (C)

The cost parameter controls the bias-variance tradeoff between decision boundary and classifies training data points correctly. The higher value of C cost would result in high variance and low bias. This means for a very high value of C, the margins will become more narrow to decrease the misclassification rate of the training data and can make model overfit and for a very low value of c, the margins will become very wide and can result in an underfit model.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
RNGversion('3.5.1')
library(readxl)
library(ggplot2)
library(knitr)
library(kernlab)
library(kableExtra)
library(geosphere)
set.seed(123456789)
stations = read.csv("E:/Machine Learning/lab 03 block 1/stations.csv",fileEncoding = "latin1")
temps = read.csv("E:/Machine Learning/lab 03 block 1/temps50k.csv")
```

```r
st = merge(stations,temps,by="station_number")
h_distance = 20000
h_date = 12
h_time = 4

# stataion name = Link?ping(    85250)
a = 15.6333
b = 58.4166

# The date to predict
date <- "2013-07-04"

#______Plots______#
  my_position = c(a,b)
  positions = cbind(st$longitude, st$latitude)
  dist_kernal = vector()
  distDiff = c()
  days_kernal = vector()
  dateDiff = vector()
  for (i in 1:nrow(st))
  {
    distDiff[i] = abs(distHaversine(my_position,positions[i,]))
    dist_kernal[i] = exp(-(distDiff[i]/h_distance)^2)
    temp = as.numeric(strftime(date,format = "%j")) -
      as.numeric(strftime(as.Date(st$date[i]),format = "%j"))
    dateDiff[i] = min(abs(temp),365-abs(temp))
    days_kernal[i] = exp(-(dateDiff[i]/h_date)^2)
  }
  time = "12:00:00"
  timeDiff = (as.numeric(as.POSIXct(paste(date, time))) -
    as.numeric(as.POSIXct(paste(date, st$time))))/3600
  timeDiff = sapply(timeDiff, function(t) {
    min(abs(t), 24 - abs(t))
  })
  time_kernel = sapply(timeDiff / (h_time), function(x) exp(-x ^ 2))
par(mfrow=c(1,2))
  plot(x = distDiff,y = dist_kernal, xlab='Distance',
       ylab='Guassian Distance', main = 'Location')

  plot(x = dateDiff,y = days_kernal, xlab='Day',
       ylab='Guassian Kernel', main = 'Date')
plot(x = timeDiff,y = time_kernel, xlab='Hour',
       ylab='Guassian Kernel', main = 'Time')
kernal = function(a, b, h_distance=2000, h_date=12, h_time=4, date, times){

#_________distance comparsion_________#
    my_position = c(a,b)
    positions = cbind(st$longitude, st$latitude)
    dist_kernal = vector()
    for (i in 1:nrow(positions))
    {
      temp = abs(distHaversine(my_position,positions[i,]))
      dist_kernal[i] = exp(-((temp)^2)/((h_distance^2)))
    }
    #________date comparison________#
    days_kernal = vector()
    for (i in 1:nrow(st))
    {
      temp = as.numeric(strftime(date,format = "%j")) - as.numeric(strftime(as.Date(st$date[i]),format
      temp = min(abs(temp),365-abs(temp))
      days_kernal[i] = exp(-((temp)^2)/((h_date^2)))
    }
    #________Time comparison________#
    fit_time = paste(date, times)
    date_time = paste(date, st$time)
    time_kernal = matrix(ncol = length(times),nrow = length(date_time))
    for (i in 1:length(fit_time))
    {
      for (j in 1:length(date_time))
      {
        temp = as.numeric(as.POSIXct(fit_time[i]))- as.numeric(as.POSIXct(date_time[j]))
        temp = temp/3600
        temp = min(abs(temp),24-abs(temp))
        time_kernal[j,i] = exp(-((temp)^2)/((h_time^2)))
      }
    }
    sum_kernals = time_kernal + dist_kernal + days_kernal
    sum_temprature = colSums(sum_kernals*st$air_temperature)/colSums(sum_kernals)

    prod_kernals = time_kernal * dist_kernal * days_kernal
    prod_temprature = colSums(prod_kernals * st$air_temperature)/ colSums(prod_kernals)

    df = data.frame(kernal_Add = sum_temprature,
                    kernal_Mult = prod_temprature)
    return(df)
  }
# stataion name = Link?ping(    85250)
  a = 15.6333
  b = 58.4166

# Time to predict
times = c("04:00:00", "06:00:00", "08:00:00", "10:00:00",
            "12:00:00", "14:00:00", "16:00:00", "18:00:00",
            "20:00:00", "22:00:00", "24:00:00")
#_________Spring_______#
df1 = kernal(a, b, h_distance = 20000, h_date = 12,h_time = 4,date="2001-04-15" , times)
df1$times = as.POSIXct(strptime(times, format="%H:%M:%S"))
 ggplot(df1)+
    geom_line(aes(x = times, y = kernal_Add, col = "add"))+theme_bw()+
    geom_point(aes(x = times, y = kernal_Add))+
    geom_line(aes(x = times,y=kernal_Mult,col = "mult"))+
    geom_point(aes(x = times,y=kernal_Mult))+
    ylab("Predicted temperature")+ggtitle('Spring Forecast')+
  scale_x_datetime( date_labels = "%H:%M:%S",date_breaks = "2 hours")
include_graphics("autmn.pdf")
include_graphics("summer.pdf")
include_graphics("winter.pdf")
#___________SUPPORT VECTOR MACHINES_________#
library(kernlab)
library(e1071)
library(caret)
#___________SUPPORT VECTOR MACHINES_________#
data(spam)
n = nrow(spam)
suppressWarnings(RNGkind(sample.kind = "Rounding"))
set.seed(12345)
index = sample(1:n, n)
train = spam[index[1:2301],]
valid = spam[index[2302:3451],]
test = spam[index[3452:4601],]
train_data = spam[index[1:3451],]

svm_model1= ksvm(type~., data = train, kernel = rbfdot(sigma=0.05), C = 0.5)
svm_model2= ksvm(type~., data = train, kernel = rbfdot(sigma=0.05), C = 1)
svm_model3= ksvm(type~., data = train, kernel = rbfdot(sigma=0.05), C = 5)

rates = function(model, data)
{
  n = nrow(data)
  prediction = predict(model, data)
  conf_mat = confusionMatrix(prediction, data$type)$table
  error = round((1-sum(diag(conf_mat))/n)*100, 2)
  acc = round((sum(diag(conf_mat))/n)*100, 2)
  lst = list("Confucion Matrix" = conf_mat,
             "Error Rates" = error,
             "Accuracy" = acc)
  return(lst)
}
rate1 = rates(model = svm_model1, data = valid)
rate2 = rates(model = svm_model2, data = valid)
rate3 = rates(model = svm_model3, data = valid)
#_____Comparison_____#
comp = data.frame(rate1[3], rate2[3], rate3[3])
colnames(comp) = c("C = 0.5", "C = 1", "C = 5")
svm_model1
kable(rate1[1],"latex", caption = "Confusion Matrix", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
svm_model2
kable(rate2[1],"latex", caption = "Confusion Matrix", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
svm_model3
kable(rate3[1],"latex", caption = "Confusion Matrix", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
kable(comp,"latex", caption = "Accuracy", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
rate = rates(model = svm_model2, data = test)
kable(rate[1],"latex", caption = "Confusion Matrix", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
svm_model2= ksvm(type~., data = spam, kernel = rbfdot(sigma=0.05), C = 1)
```