Data Mining

Assignment 4

Code shyam:

install.packages("caret")

library(caret)

library(forecast)

getwd()

setwd('C:/Users/Meghashyam/Desktop')

data <- read.csv("abc.csv", na.strings=c("","NA"), header = FALSE)

data1 <- data[2]

head(data1)

timeseries <- ts(data1, frequency=365)

plot.ts(timeseries)

timeseries

time(timeseries)

help(ts)

quantile(timeseries)

plot(decompose(timeseries))

plot(diff(timeseries))

ggseasonplot(timeseries)

adf.test(timeseries)

acf(timeseries, lag.max = 20)

res = auto.arima(timeseries, stepwise = F, approximation = F)

plot(forecast(res, h= 3))

Code:

# If you are on linux you can uncomment the following lines to run caret on multiple cores

# library(doMC)

# registerDoMC(4)

library(caret)

setwd("C:/Users/Meghashyam/Desktop/")

xy=read.table("daymatrix.txt",sep=' ',header=F)

y=xy[,8]

head(y)

x=xy[,1:7]

# Using pre-sliced data

myCvControl <- trainControl(method = "repeatedCV",

number=10,

repeats = 5)

# Linear regression

glmFitTime <- train(V8 ~ .,

data = xy,

method = "glm",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

glmFitTime

summary(glmFitTime)

y\_hat = predict(glmFitTime, newdata = x)

mean(100\*abs(y\_hat-y)/y)

# Your error with linear regression

# Support Vector Regression

svmFitTime <- train(V8 ~ .,

data = xy,

method = "svmRadial",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

svmFitTime

summary(svmFitTime)

y\_hat = predict(svmFitTime, newdata = x)

mean(100\*abs(y\_hat-y)/y)

# Your error with support vector regression

# Neural Network

nnFitTime <- train(V8 ~ .,

data = xy,

method = "avNNet",

preProc = c("center", "scale"),

trControl = myCvControl,

tuneLength = 10,

linout = T,

trace = F,

MaxNWts = 10 \* (ncol(xy) + 1) + 10 + 1,

maxit = 500)

nnFitTime

summary(nnFitTime)

y\_hat = predict(nnFitTime, newdata = x)

mean(100\*abs(y\_hat-y)/y)

# Your error with neural networks

# You can experiment with other methods, here is where you can find the methods caret supports:

# https://topepo.github.io/caret/available-models.html

# Compare models

resamps <- resamples(list(lm = glmFitTime,

svn = svmFitTime,

nn = nnFitTime))

summary(resamps)

# Now working with the time-series modeling

t=read.csv("abc.csv",header=F)

head(t)

t1 <- t[2]

tSeries = ts(t1, frequency = 7)

head(tSeries)

plot.ts(tSeries)

plot(decompose(tSeries))

ggseasonplot(tSeries)

library(tseries)

adf.test(tSeries)

acf(tSeries, lag.max = 20)

library(forecast)

hw = ets(tSeries,model="MAM")

mean(100\*abs(fitted(hw) - tSeries)/tSeries)

# Your Holt-Winters error

res = auto.arima(tSeries, stepwise = F, approximation = F)

res

ar <- Arima(tSeries, order=c(1,0,2))

mean(100\*abs(fitted(ar) - tSeries)/tSeries)

# Your Arima error

plot(forecast(ar, h= 7))

Output start:

There were 50 or more warnings (use warnings() to see the first 50)

> # If you are on linux you can uncomment the following lines to run caret on multiple cores

> # library(doMC)

> # registerDoMC(4)

> library(caret)

> setwd("C:/Users/Meghashyam/Desktop/")

>

> xy=read.table("daymatrix.txt",sep=' ',header=F)

> y=xy[,8]

> head(y)

[1] 29598.09 30253.59 24296.65 24842.11 24047.85 26076.56

> x=xy[,1:7]

>

> # Using pre-sliced data

> myCvControl <- trainControl(method = "repeatedCV",

+ number=10,

+ repeats = 5)

Warning message:

`repeats` has no meaning for this resampling method.

>

> # Linear regression

> glmFitTime <- train(V8 ~ .,

+ data = xy,

+ method = "glm",

+ preProc = c("center", "scale"),

+ tuneLength = 10,

+ trControl = myCvControl)

> glmFitTime

Generalized Linear Model

182 samples

7 predictor

Pre-processing: centered (7), scaled (7)

Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 163, 165, 165, 164, 164, 165, ...

Resampling results:

RMSE Rsquared MAE

1988.478 0.7171525 1558.176

> summary(glmFitTime)

Call:

NULL

Deviance Residuals:

Min 1Q Median 3Q Max

-5038.7 -1244.4 -41.6 1127.4 6626.6

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 27003.31 144.62 186.717 < 2e-16 \*\*\*

V1 1747.05 200.04 8.734 1.99e-15 \*\*\*

V2 1144.83 233.03 4.913 2.06e-06 \*\*\*

V3 -479.42 243.42 -1.970 0.0505 .

V4 143.20 244.31 0.586 0.5585

V5 -91.27 244.72 -0.373 0.7096

V6 -207.80 233.65 -0.889 0.3750

V7 1405.65 199.67 7.040 4.27e-11 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 3806609)

Null deviance: 2392524914 on 181 degrees of freedom

Residual deviance: 662349990 on 174 degrees of freedom

AIC: 3284

Number of Fisher Scoring iterations: 2

> y\_hat = predict(glmFitTime, newdata = x)

> mean(100\*abs(y\_hat-y)/y)

[1] 5.509059

> # Your error with linear regression

>

> # Support Vector Regression

> svmFitTime <- train(V8 ~ .,

+ data = xy,

+ method = "svmRadial",

+ preProc = c("center", "scale"),

+ tuneLength = 10,

+ trControl = myCvControl)

> svmFitTime

Support Vector Machines with Radial Basis Function Kernel

182 samples

7 predictor

Pre-processing: centered (7), scaled (7)

Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 163, 162, 164, 165, 164, 164, ...

Resampling results across tuning parameters:

C RMSE Rsquared MAE

0.25 2095.437 0.7046469 1588.147

0.50 2030.012 0.7095712 1536.152

1.00 1986.803 0.7161020 1498.984

2.00 1945.450 0.7263900 1450.738

4.00 1953.324 0.7228637 1443.400

8.00 1996.925 0.7098797 1468.776

16.00 2050.042 0.6942697 1512.908

32.00 2154.757 0.6666529 1593.484

64.00 2298.342 0.6291648 1703.738

128.00 2304.367 0.6273087 1712.300

Tuning parameter 'sigma' was held constant at a value of 0.2237152

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were sigma = 0.2237152 and C = 2.

> summary(svmFitTime)

Length Class Mode

1 ksvm S4

> y\_hat = predict(svmFitTime, newdata = x)

> mean(100\*abs(y\_hat-y)/y)

[1] 2.414319

> # Your error with support vector regression

>

> # Neural Network

> nnFitTime <- train(V8 ~ .,

+ data = xy,

+ method = "avNNet",

+ preProc = c("center", "scale"),

+ trControl = myCvControl,

+ tuneLength = 10,

+ linout = T,

+ trace = F,

+ MaxNWts = 10 \* (ncol(xy) + 1) + 10 + 1,

+ maxit = 500)

Warning messages:

1: model fit failed for Fold01.Rep1: size=13, decay=0.0000000, bag=FALSE Error in { : task 1 failed - "too many (118) weights"

2: model fit failed for Fold01.Rep1: size=15, decay=0.0000000, bag=FALSE Error in { : task 1 failed - "too many (136) weights"

3: model fit failed for Fold01.Rep1: size=17, decay=0.0000000, bag=FALSE Error in { : task 1 failed - "too many (154) weights"

4: model fit failed for Fold01.Rep1: size=19, decay=0.0000000, bag=FALSE Error in { : task 1 failed - "too many (172) weights"

5: model fit failed for Fold01.Rep1: size=13, decay=0.1000000, bag=FALSE Error in { : task 1 failed - "too many (118) weights"

6: model fit failed for Fold01.Rep1: size=15, decay=0.1000000, bag=FALSE Error in { : task 1 failed - "too many (136) weights"

7: model fit failed for Fold01.Rep1: size=17, decay=0.1000000, bag=FALSE Error in { : task 1 failed - "too many (154) weights"

8: model fit failed for Fold01.Rep1: size=19, decay=0.1000000, bag=FALSE Error in { : task 1 failed - "too many (172) weights"

> resamps <- resamples(list(lm = glmFitTime,

+ svn = svmFitTime,

+ nn = nnFitTime))

Error in resamples(list(lm = glmFitTime, svn = svmFitTime, nn = nnFitTime)) :

object 'nnFitTime' not found

> summary(resamps)

Error in summary(resamps) : object 'resamps' not found

>

>

> # Now working with the time-series modeling

> t=read.csv("abc.csv",header=F)

> head(t)

V1 V2

1 ï»¿01/01/14 24090.91

2 01/02/14 26559.81

3 01/03/14 26167.46

4 01/04/14 30076.56

5 01/05/14 30234.45

6 01/06/14 23507.18

>

> t1 <- t[2]

> tSeries = ts(t1, frequency = 7)

> head(tSeries)

Time Series:

Start = c(1, 1)

End = c(1, 6)

Frequency = 7

V2

[1,] 24090.91

[2,] 26559.81

[3,] 26167.46

[4,] 30076.56

[5,] 30234.45

[6,] 23507.18

> plot.ts(tSeries)

> plot(decompose(tSeries))

> ggseasonplot(tSeries)

>

>

> library(tseries)

> adf.test(tSeries)

Augmented Dickey-Fuller Test

data: tSeries

Dickey-Fuller = -2.6444, Lag order = 7, p-value = 0.305

alternative hypothesis: stationary

>

> acf(tSeries, lag.max = 20)

>

> library(forecast)

> hw = ets(tSeries,model="MAM")

> mean(100\*abs(fitted(hw) - tSeries)/tSeries)

[1] 5.189084

> # Your Holt-Winters error

>

>

> res = auto.arima(tSeries, stepwise = F, approximation = F)

Output end:

Database Queries:

use dataset04;

select RecordDateTime, Value from MT123electricity

limit 5000000;

select SUBSTRING(RecordDateTime,1,10) as dat, Sum(Value) from MT123electricity

group by dat

limit 5000000;