# Romantic Relationships and Service-scpaes: Insights from the Yelp dataset

## R Markdown

In this script I will analyse different factors that I have extracted previously from Natural Language Processing. Here I will try different supervised learning models to find the best model and analyse the resaults.I will only focuse on wives and husbands here.

```
# restaurants with more than 50 reviews
rests = read.csv("C:/my files/Romantic/Restaurants_50.csv", header = T,sep = ",")

features = rests[,c(16,18:26,28:30,32,33,35:42,46,49:52,54:58,63)] #Selecting Restaurants' features
features$price = features$expensive+features$pricey
features = scale(features[,c(-10,-13)])
head(features,3)
```

```
##         liquor      dinner      lunch  breakfast     brunch         busy
## [1,]  1.417226  0.08110468 -0.5508192 -0.3958811 -0.3043288 -0.772868722
## [2,] -0.431313 -1.43011296  0.9100620  4.9924507  0.8331276 -0.180823491
## [3,] -0.431313 -0.74337116 -0.7356572 -0.3958811 -0.3043288 -0.001553425
##         décor.x      dance      cheap   ambiance       loud    outdoor
## [1,] -0.8360444 -0.1799082 -0.1157938 -0.2611123  1.3724558  1.3845786
## [2,] -0.7648285 -0.1799082 -0.1251639 -0.9735335 -0.2330688 -0.4418551
## [3,]  1.5648425 -0.1799082  0.2008118 -0.4555412 -0.1855114 -0.4418551
##         authent     modern       cozy       quiet     classy    upscale
## [1,] -0.5044206 -0.3988852  0.8094512 -0.06232542  2.3482305 -0.5517144
## [2,] -0.6502609 -0.3988852 -0.5546617 -0.57137390 -0.4735163 -0.5517144
## [3,] -0.2261065  0.7281055 -0.5546617 -0.57137390 -0.4735163 -0.5517144
##          trendy   creative    tourist     tradit      sport    amb_pos
## [1,] -0.3780857  1.4306850 -0.3688589 -0.9204187 -0.2502969 -0.2503328
## [2,]  2.2478626  0.4205913 -0.3688589  2.0784908 -0.2502969 -0.2045306
## [3,] -0.3780857 -0.3681407 -0.3688589  1.7317929 -0.2502969 -0.8710443
##       neigh_pos  neigh_neg    music.y   food_pos   food_neg   serv_pos
## [1,] -0.4260918 -0.1205859 -0.3242974 -0.7384232  0.6874665 -0.8509367
## [2,]  0.7313909 -0.1205859 -0.3242974  0.9071519 -0.4869481  2.9770938
## [3,]  0.8534938 -0.1205859 -0.3242974 -1.2532849 -0.4262442 -0.2043208
##        serv_neg      stars      price
## [1,]  0.6809805 -0.2821377  1.4402791
## [2,] -0.8057611  0.7348333 -0.5294355
## [3,] -1.0983262 -0.2821377 -0.7768064
```

We will next separate the variables for our analyses.

```
# Romantic variables
wif.hus = rests$spouse
family = rests$family
BF.GF = rests$BFGF
date = rests$date
```

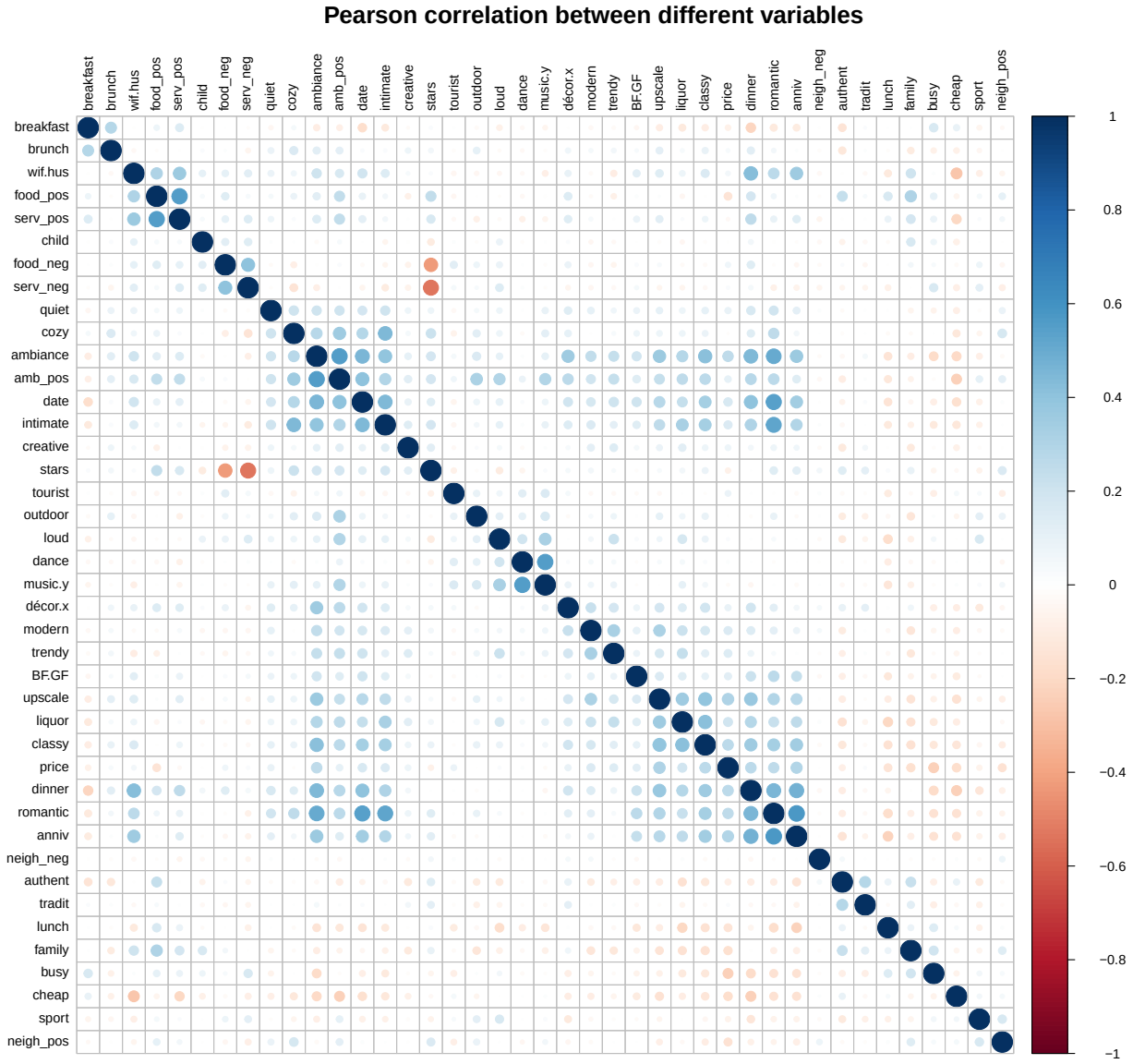
```r
# other relevant romantic variables
romantic = rests$romantic
intimate = rests$intimate
child = rests$children
anniv = rests$anniversary


#Making a dataframe for all the variables that we want
all = data.frame(wif.hus,family,BF.GF,date,romantic,intimate,child,anniv,features)
names(all)
```

```
##  [1] "wif.hus"   "family"    "BF.GF"     "date"      "romantic"
##  [6] "intimate"  "child"     "anniv"     "liquor"    "dinner"
## [11] "lunch"     "breakfast" "brunch"    "busy"      "décor.x"
## [16] "dance"     "cheap"     "ambiance"  "loud"      "outdoor"
## [21] "authent"   "modern"    "cozy"      "quiet"     "classy"
## [26] "upscale"   "trendy"    "creative"  "tourist"   "tradit"
## [31] "sport"     "amb_pos"   "neigh_pos" "neigh_neg" "music.y"
## [36] "food_pos"  "food_neg"  "serv_pos"  "serv_neg"  "stars"
## [41] "price"
```

To check how these variables correlate we make a correlation matrix and plot.

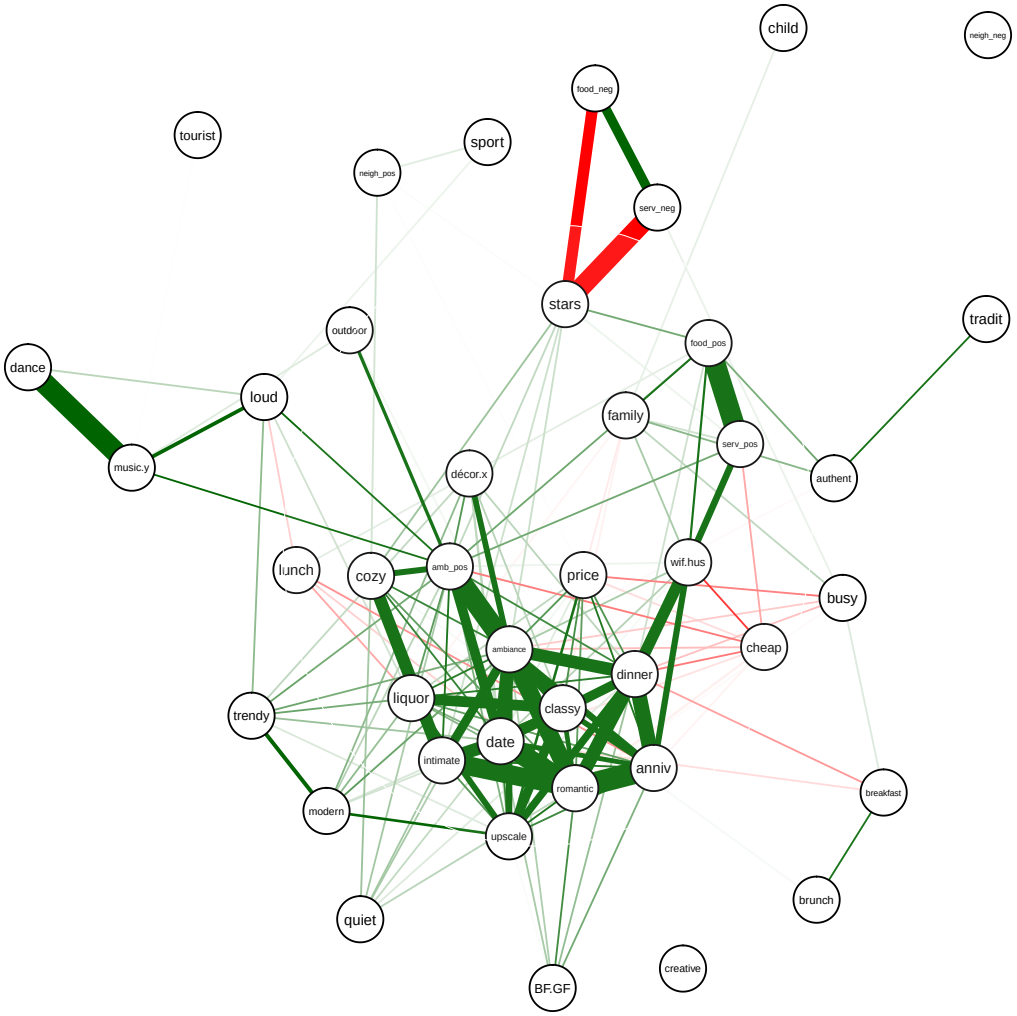
```r
corr = cor(all, use="complete.obs", method="pearson")
corrplot::corrplot(corr, method="circle",order ="hclust",tl.cex=0.85, tl.col = "black")
title("Pearson correlation between different variables ", cex.main= 1.5)
```




A few correlations can be seen that are interesting. we can visualize the correlations in a network as well.

```
qgraph(corr, minimum = 0.15,color = "white", posCol = "darkgreen", negCol = "red",cut = 0.3, vsize = 3.5
title("Pearson corrletaion between different variables ", cex.main= 2.5)
```

**Pearson corrletaion between different variables**




In the next steps we will look into different methods to find the factors that directly affect each romantic variables (i.e. Wif.Hus, BF.GF, date, family, romantic etc.). We intend to find the best model with the least MSE using different learning methods.

```
set.seed(1)
train = sample(nrow(all),1500, replace = F)
test = -train
all.x = all[,c(-1:-8)]
all.y = all[,c(1:8)]
all.x.train = all.x[train,]
all.y.train = all.y[train,]
all.x.test = all.x[test,]
all.y.test = all.y[test,]
all.train = all[train,]
```

```
all.test = all[test,]
```

First we begin with simple methods, the least square method:

```
model.ls1.wifhus <- lm(wif.hus ~ . -BF.GF-family-date-romantic-intimate-child-anniv,all.train)
pred.ls1.wifhus <- predict(model.ls1.wifhus, newdata = all.test) # validation predictions
mse.ls1.wifhus <- mean((all.test$wif.hus - pred.ls1.wifhus)^2) # mean prediction error
mse.ls1.wifhus
```

```
## [1] 0.007472527
```

```
model.best.wifhus = regsubsets(wif.hus ~ . -BF.GF-family-date-romantic-intimate-child-anniv, data =all.
reg.summary = summary(model.best.wifhus)
reg.summary$rss
```

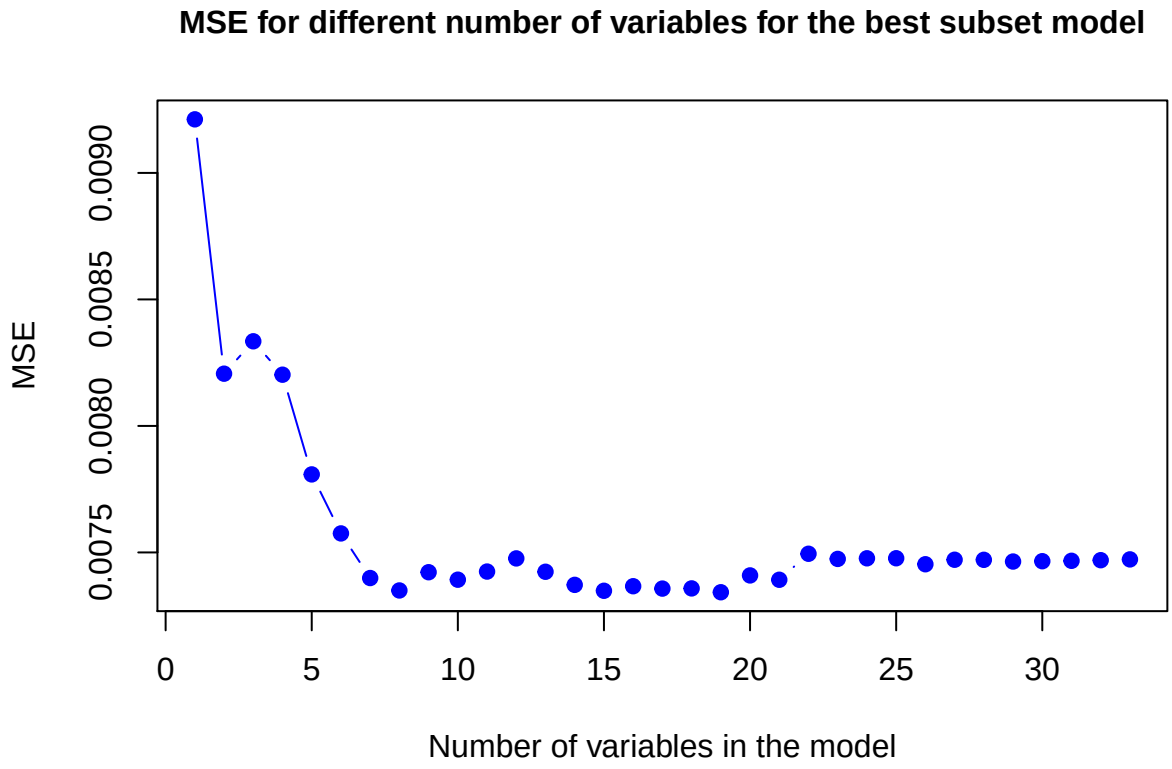
```
##  [1] 15.22697 14.00610 13.60123 13.24824 12.95425 12.69977 12.62329
##  [8] 12.54333 12.46537 12.39300 12.34449 12.29572 12.24420 12.20226
## [15] 12.16116 12.12758 12.10470 12.09082 12.07815 12.06720 12.05438
## [22] 12.04260 12.03097 12.02337 12.01745 12.01335 12.01080 12.00971
## [29] 12.00872 12.00804 12.00743 12.00733 12.00724
```

```
test.mat.wifhus = model.matrix(wif.hus ~ . -BF.GF-family-date-romantic-intimate-child-anniv, data = all
val.errors.wifhus = rep(NA,33)
for (i in 1:33) {
  coefi = coef(model.best.wifhus, id = i)
  pred = test.mat.wifhus[, names(coefi)]%*%coefi
  val.errors.wifhus[i] = mean((all.test$wif.hus- pred)^2)
}

print (val.errors.wifhus)
```

```
##  [1] 0.009211724 0.008206514 0.008334490 0.008202476 0.007808411
##  [6] 0.007575177 0.007398808 0.007349715 0.007421749 0.007392233
## [11] 0.007424401 0.007476350 0.007423623 0.007371790 0.007348422
## [16] 0.007366472 0.007357110 0.007357826 0.007342660 0.007409082
## [21] 0.007391989 0.007494762 0.007474359 0.007476814 0.007477003
## [26] 0.007453258 0.007471059 0.007470764 0.007464009 0.007465324
## [31] 0.007466877 0.007469303 0.007472527
```

```
plot(val.errors.wifhus, ylab = "MSE",xlab = "Number of variables in the model", pch = 19, type = "b", c
title("MSE for different number of variables for the best subset model", cex.main = 1)
```




```
 # we select 8 variables because after that MSE doesn't seem to improve
coef(model.best.wifhus,19)
```

```
##  (Intercept)       dinner        lunch    breakfast       brunch
##  0.180107998  0.039490803 -0.018401123  0.007395415 -0.011560327
##        cheap         loud       modern        quiet       trendy
## -0.018523105 -0.003554148 -0.006880164  0.002806836 -0.009960718
##     creative      tourist       tradit        sport    neigh_pos
##  0.006256615 -0.007492208 -0.006096799 -0.005877035 -0.004621491
##      music.y     food_pos     serv_pos     serv_neg        price
## -0.004320451  0.020273294  0.013945290  0.005582620 -0.005949358
```

```
mse.best.wifhus <- val.errors.wifhus[19]
mse.best.wifhus
```

```
## [1] 0.00734266
```

The model seem to have improved a little through the varibles selection procedure. We can also try a k-fold cross validation (k = 10) on the previous model to see if we can see any improvement.

```
k=10
set.seed (1)
folds=sample (1:k,nrow(all),replace =TRUE)
cv.errors =matrix (NA ,k,33, dimnames =list(NULL , paste (1:33) ))
```

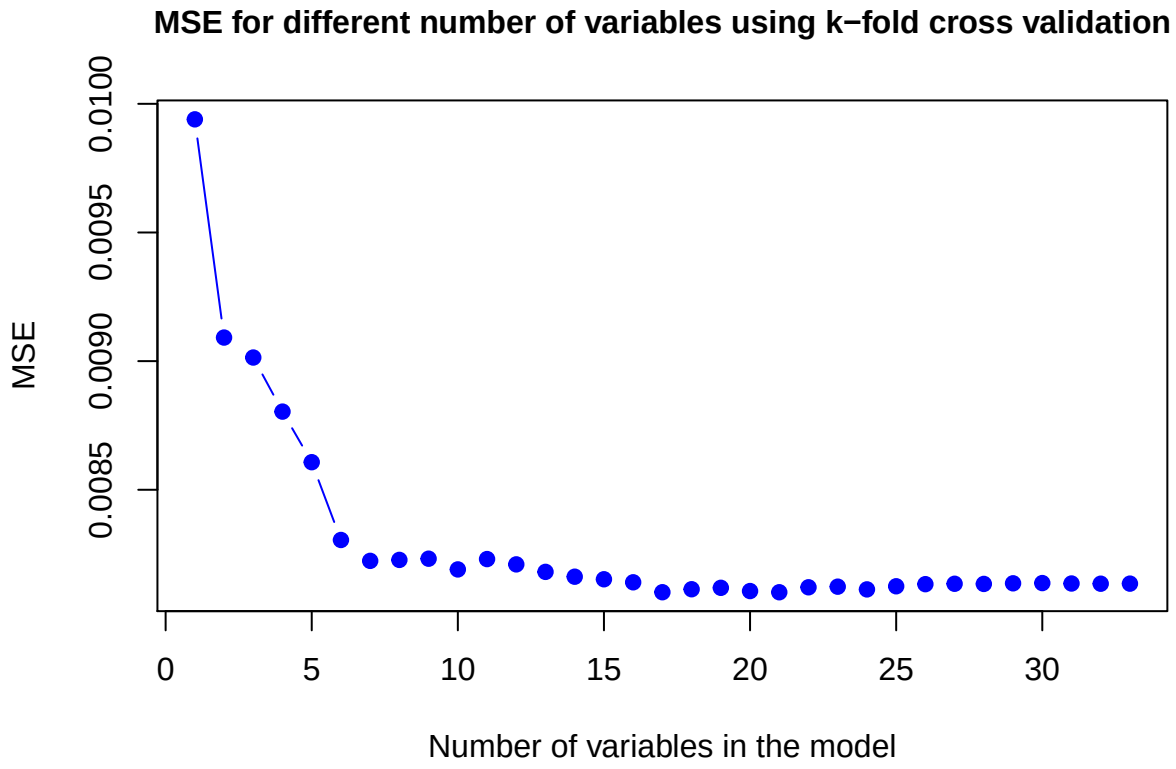
```r
#Define a function to enable us to predict with th ebest subset model
predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}


for(j in 1:k){
   best.fit =regsubsets(wif.hus~. -BF.GF-family-date-romantic-intimate-child-anniv, data = all[folds !=
   for(i in 1:33) {
     pred=predict (best.fit ,all[folds ==j,], id=i)
     cv.errors [j,i]=mean( (all$wif.hus[folds ==j]-pred)^2)
     }
}

mean.cv.errors =apply(cv.errors ,2, mean)
mean.cv.errors
```

```
##           1           2           3           4           5           6
## 0.009939689 0.009091646 0.009013980 0.008803806 0.008607150 0.008304844
##           7           8           9          10          11          12
## 0.008223720 0.008227430 0.008232040 0.008190358 0.008230631 0.008210020
##          13          14          15          16          17          18
## 0.008180823 0.008162081 0.008152176 0.008140311 0.008101785 0.008113396
##          19          20          21          22          23          24
## 0.008118910 0.008106309 0.008101724 0.008121012 0.008123367 0.008112756
##          25          26          27          28          29          30
## 0.008124954 0.008133080 0.008134601 0.008133993 0.008136416 0.008137304
##          31          32          33
## 0.008135736 0.008134864 0.008135203
```

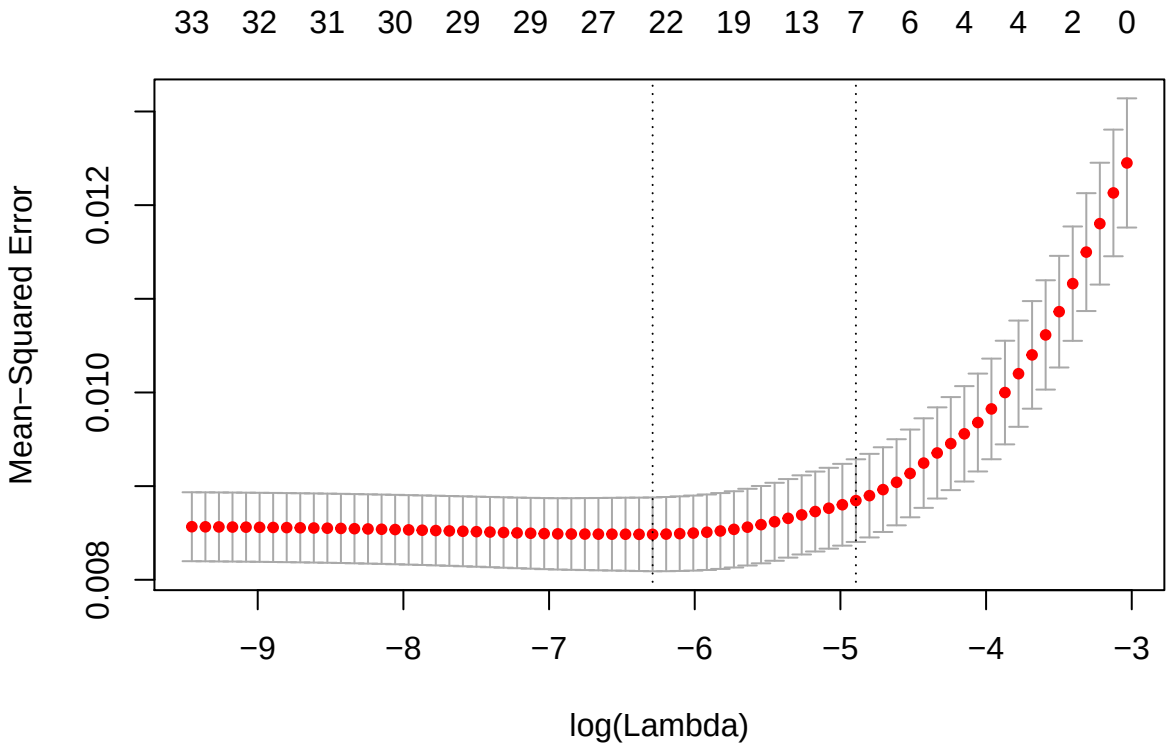
```r
par(mfrow =c(1,1))
plot(mean.cv.errors ,ylab = "MSE",xlab = "Number of variables in the model", pch = 19, type = "b", col =
title("MSE for different number of variables using k-fold cross validation", cex.main = 1)
```




The k-fold CV resulte in higher MSE for this regression. We will now try model shrinkage methods starting with Lasso regression.

```
grid =10^seq (10,-3, length =10000)
lasso.mod.wifhus =glmnet (as.matrix(all.x.train),as.matrix(all.y.train$wif.hus),alpha =1, lambda =grid,

set.seed (1)
cv.out =cv.glmnet (as.matrix(all.x.train),as.matrix(all.y.train$wif.hus),alpha =1)
plot(cv.out)
```




```
bestlam =cv.out$lambda.min #best lambda
bestlam
```

```
## [1] 0.001856342
```

```
lasso.pred.wifhus=predict(lasso.mod.wifhus ,s=bestlam ,newx=as.matrix(all.x.test))
mse.lasso.wifhus<- mean((lasso.pred.wifhus-all.y.test$wif.hus)^2)
mse.lasso.wifhus
```

```
## [1] 0.007296516
```

So far the resulting MSE from the LAsso regression is the smalles (i.e. 0.00729). We will now try the PCR model.

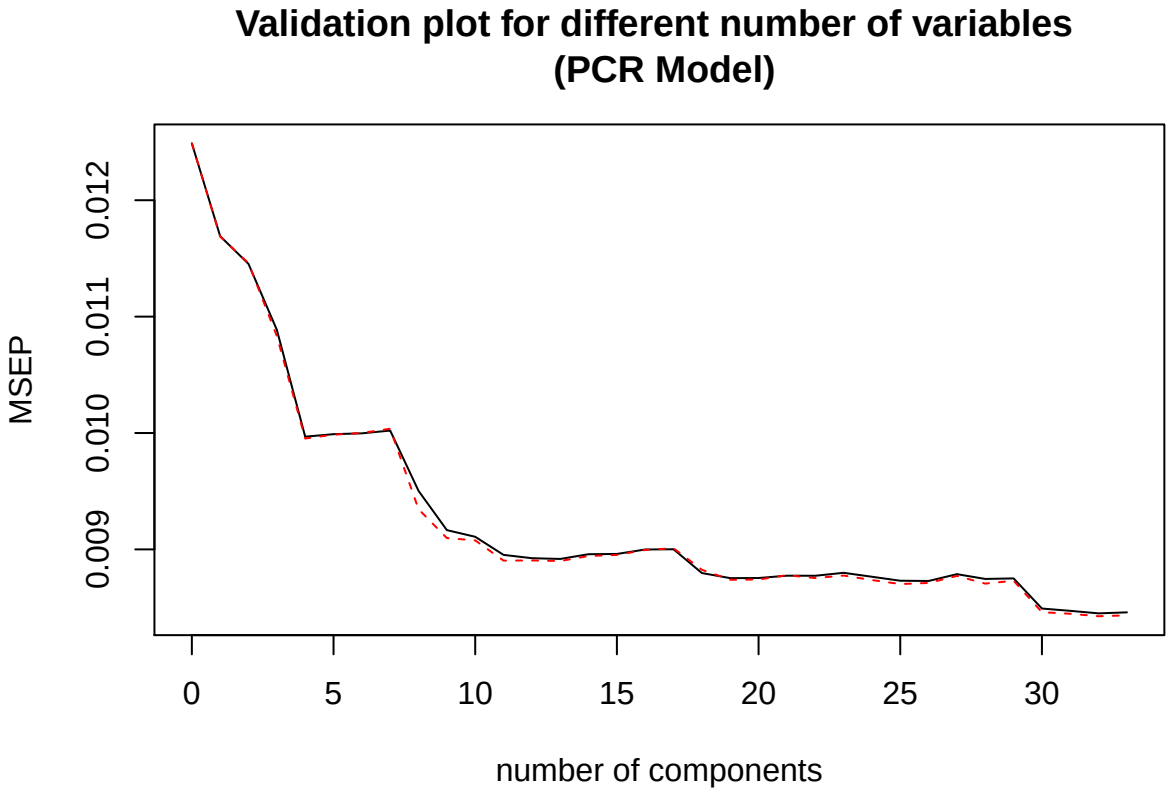
```
set.seed (1)
pcr_fit=pcr(wif.hus~.-BF.GF-family-date-romantic-intimate-child-anniv,
            data= all.train,scale=FALSE,validation ="CV")

summary (pcr_fit)
```

```
## Data:    X dimension: 1500 33
##  Y dimension: 1500 1
## Fit method: svdpc
## Number of components considered: 33
```

```
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.1118   0.1081    0.107   0.1043  0.09984  0.09995  0.09999
## adjCV       0.1118   0.1081    0.107   0.1041  0.09976  0.09993  0.10000
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.1001  0.09748  0.09574   0.09544   0.09462   0.09447   0.09444
## adjCV   0.1002  0.09667  0.09538   0.09528   0.09436   0.09437   0.09434
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV      0.09465   0.09466   0.09486   0.09487   0.09379   0.09356
## adjCV   0.09457   0.09462   0.09484   0.09492   0.09394   0.09348
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV      0.09356   0.09367   0.09367   0.09380   0.09362   0.09344
## adjCV   0.09349   0.09369   0.09356   0.09368   0.09347   0.09329
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV      0.09343   0.09374   0.09352   0.09355   0.09215   0.09205
## adjCV   0.09334   0.09366   0.09331   0.09344   0.09198   0.09191
##        32 comps  33 comps
## CV      0.09193   0.09197
## adjCV   0.09179   0.09183
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         12.319    19.70    26.10    32.38    37.36    41.55    45.44
## wif.hus    6.462     8.52    14.39    20.14    20.75    20.91    20.94
##          8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X          49.11    52.63     55.95     59.09     62.09     64.92
## wif.hus    25.69    27.88     28.38     29.57     29.69     29.78
##          14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X           67.58     70.18     72.58     74.91     77.12     79.24
## wif.hus     29.81     29.82     29.91     30.11     30.98     31.62
##          20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X           81.28     83.21     85.11     86.89     88.57     90.20
## wif.hus     31.96     31.98     32.27     32.35     32.68     32.87
##          26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## X           91.77     93.31     94.68     95.98     97.08     98.11
## wif.hus     32.89     32.90     33.81     33.84     35.49     35.53
##          32 comps  33 comps
## X           99.10    100.00
## wif.hus     35.73     35.82
```

```
validationplot(pcr_fit,val.type="MSEP",main = "Validation plot for different number of variables \n (PC
```




```
pcr_valid=predict(pcr_fit ,all.test, ncomp =25)
mse.pcr.wifhus<- mean((pcr_valid-all.test$wif.hus)^2)
mse.pcr.wifhus
```

```
## [1] 0.007222871
```

Now we run a PLS model to see if it will make any difference if we project the response variable as well.

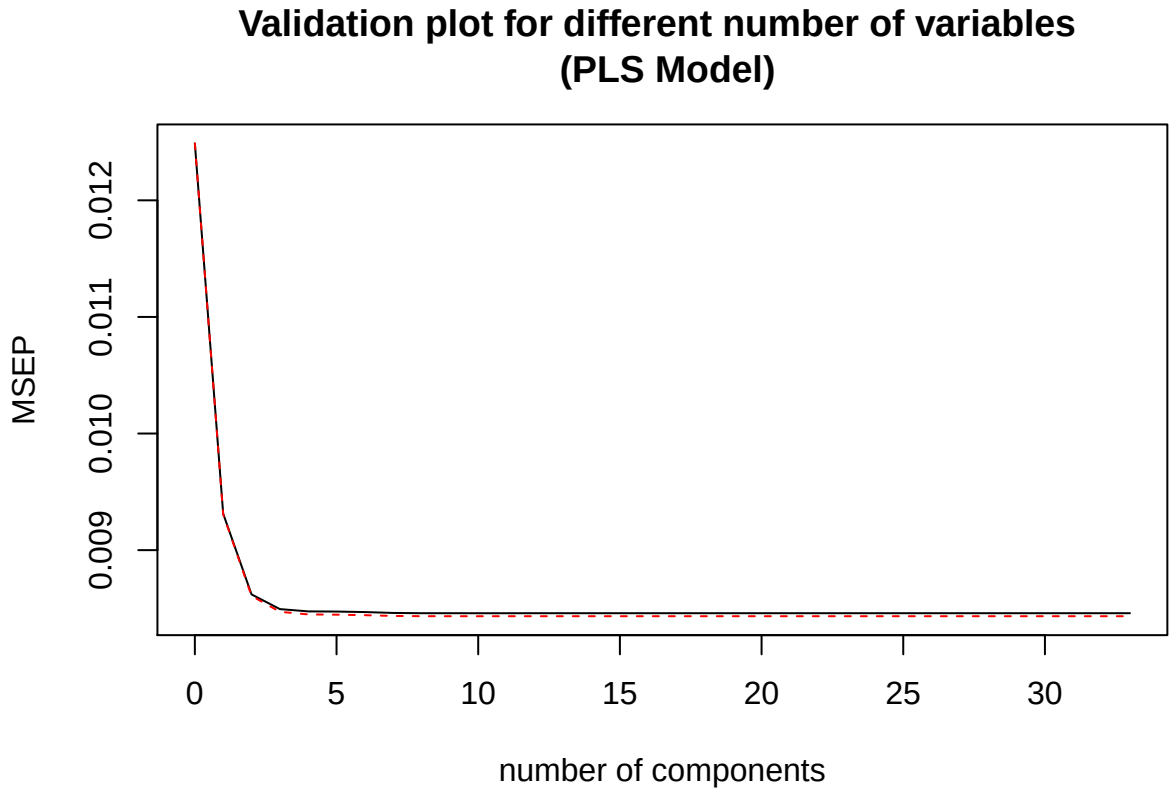
```
set.seed (1)
plsr_fit=plsr(wif.hus~.-BF.GF-family-date-romantic-intimate-child-anniv,
              data= all.train,scale=FALSE,validation ="CV")

summary (plsr_fit)
```

```
## Data:    X dimension: 1500 33
##  Y dimension: 1500 1
## Fit method: kernelpls
## Number of components considered: 33
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.1118  0.09651  0.09285  0.09217  0.09206  0.09205  0.09203
## adjCV       0.1118  0.09647  0.09278  0.09204  0.09192  0.09191  0.09188
```

```
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV     0.09199  0.09197  0.09197   0.09197   0.09197   0.09197   0.09197
## adjCV  0.09185  0.09183  0.09183   0.09183   0.09183   0.09183   0.09183
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV      0.09197   0.09197   0.09197   0.09197   0.09197   0.09197
## adjCV   0.09183   0.09183   0.09183   0.09183   0.09183   0.09183
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV      0.09197   0.09197   0.09197   0.09197   0.09197   0.09197
## adjCV   0.09183   0.09183   0.09183   0.09183   0.09183   0.09183
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV      0.09197   0.09197   0.09197   0.09197   0.09197   0.09197
## adjCV   0.09183   0.09183   0.09183   0.09183   0.09183   0.09183
##        32 comps  33 comps
## CV      0.09197   0.09197
## adjCV   0.09183   0.09183
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X          9.604    17.43    21.80    25.61    30.72    33.66    37.64
## wif.hus   26.665    33.09    35.12    35.65    35.78    35.81    35.82
##          8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X          41.16    45.43     48.64     51.09     53.97     56.29
## wif.hus    35.82    35.82     35.82     35.82     35.82     35.82
##          14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X           58.94     61.31     63.57     65.96     68.73     71.27
## wif.hus     35.82     35.82     35.82     35.82     35.82     35.82
##          20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X           73.84     75.86     77.94     80.00     82.00     84.12
## wif.hus     35.82     35.82     35.82     35.82     35.82     35.82
##          26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## X           86.12     88.12     90.18     92.32     94.44     96.49
## wif.hus     35.82     35.82     35.82     35.82     35.82     35.82
##          32 comps  33 comps
## X           98.12    100.00
## wif.hus     35.82     35.82
```

```
validationplot(plsr_fit,val.type="MSEP", main = "Validation plot for different number of variables \n (
```




```
plsr_valid=predict(plsr_fit ,all.test, ncomp =5)
mse.pls.wifhus<- mean((plsr_valid-all.test$wif.hus)^2)
mse.pls.wifhus
```

```
## [1] 0.007461713
```

The model suggested by PLS is simpler but MSE is highe rfor this model. At last, we try regression trees and random forests.

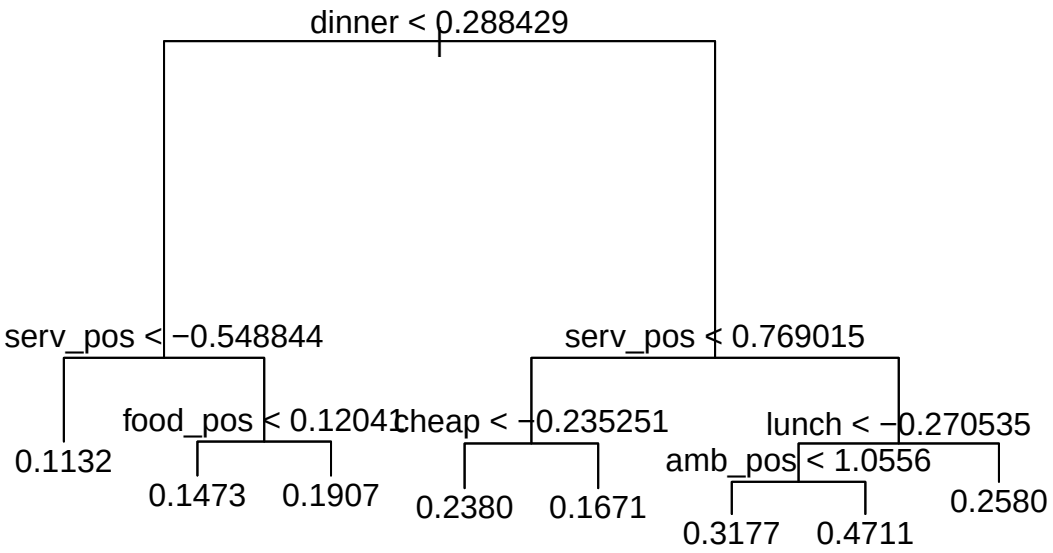
```
tree_wh =tree(wif.hus~.-BF.GF-family-date-romantic-intimate-child-anniv,data= all.train)
summary (tree_wh)
```

```
##
## Regression tree:
## tree(formula = wif.hus ~ . - BF.GF - family - date - romantic -
##     intimate - child - anniv, data = all.train)
## Variables actually used in tree construction:
## [1] "dinner"   "serv_pos" "food_pos" "cheap"    "lunch"    "amb_pos"
## Number of terminal nodes:  8
## Residual mean deviance:  0.008809 = 13.14 / 1492
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.24930 -0.06356 -0.01321  0.00000  0.05203  0.44820
```

```
cv_wh =cv.tree(tree_wh)
yhat=predict(tree_wh ,newdata =all.test)
y.test=all.test$wif.hus
mse.tree.wifhus<-mean((yhat-y.test)^2)
mse.tree.wifhus
```

```
## [1] 0.008984373
```

```
plot(tree_wh)
text(tree_wh,pretty=0)
```




The MSE for the regression tree is high, compared to other methods. We will now try a random forest.

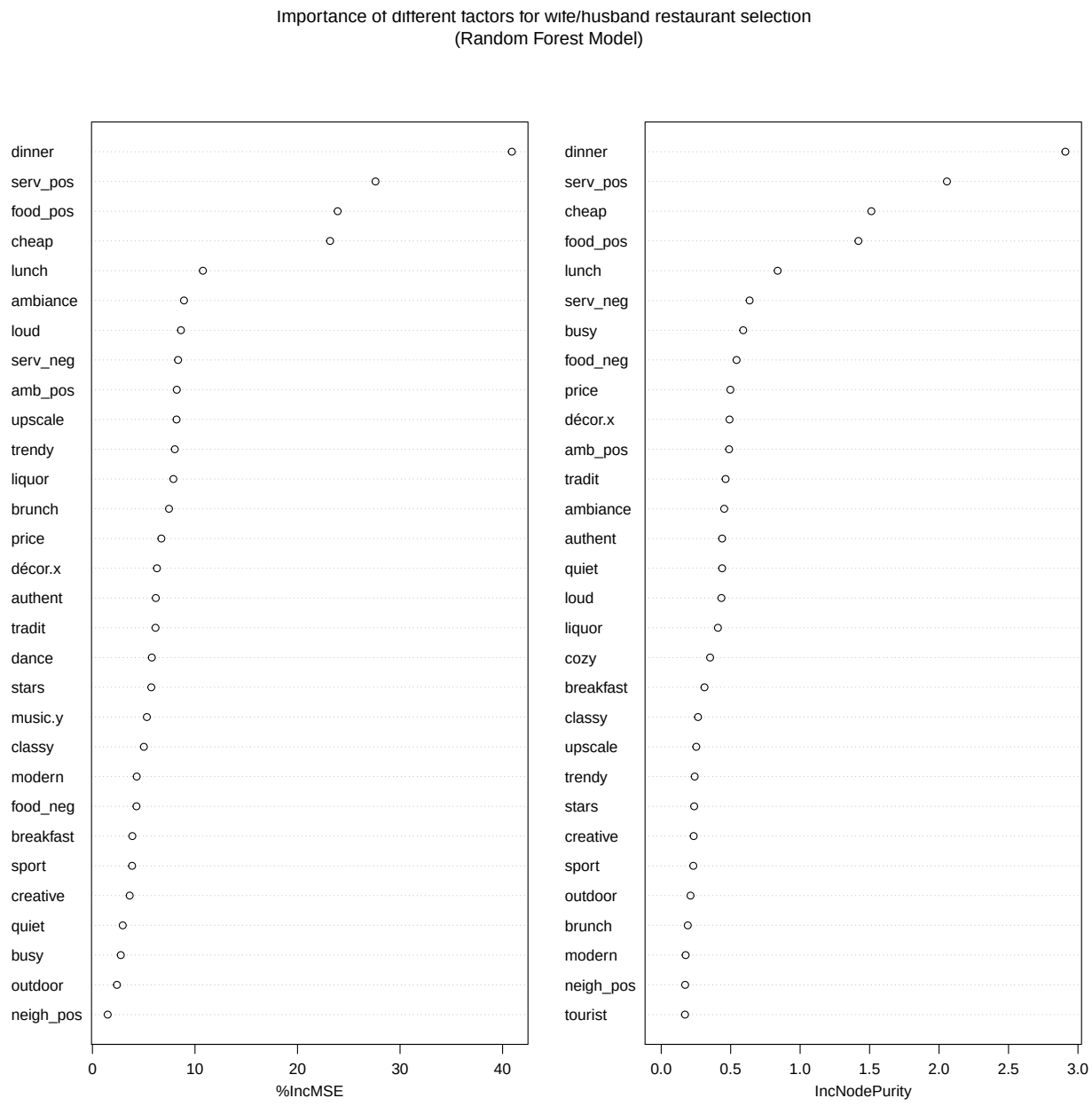
```
set.seed (1)
rand.wh =randomForest(wif.hus~.-BF.GF-family-date-romantic-intimate-child-anniv,data=all ,subset =train
rand.wh
```

```
##
## Call:
##  randomForest(formula = wif.hus ~ . - BF.GF - family - date -      romantic - intimate - child - anni
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 0.008306602
##                     % Var explained: 33.4
```

```
yhat.rand = predict(rand.wh ,newdata =all.test)
mse.forest.wifhus<-mean((yhat.rand - all.test$wif.hus)^2)
mse.forest.wifhus
```

```
## [1] 0.007637067
```

```
#importance(rand.wh)
varImpPlot (rand.wh, main = "Importance of different factors for wife/husband restaurant selection \n (
```




Now let's compare the resulting MSE for different models:

| Model | MSE |
|---|---|
| Linear Least Square | 0.0074725 |

| Model | MSE |
|---|---|
| Best Subset | 0.0073427 |
| Best Subset (CV K=10) | 0.0081403 |
| Lasso | 0.0072965 |
| PCR | 0.0072229 |
| PLS | 0.0074617 |
| Regression Tree | 0.0089844 |
| Random Forest | 0.0076371 |

Although the PCR model gives the smallest MSE, it's difficult to interpret the direct effect of factors in this model, as each component is a combination of many variables. The Lasso regression, although has a MSE slightly higher than the PCR model, but it provides a better model for analytical purposes.

```
lasso.full.wifhus = glmnet(as.matrix(all.x),as.matrix(all.y$wif.hus),alpha =1)
lasso_coef = predict (lasso.full.wifhus ,s=bestlam, type = "coefficient", newx = as.matrix(all.x))
lasso_coef
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  0.1795774886
## liquor       .
## dinner       0.0354531132
## lunch       -0.0148892905
## breakfast    0.0035523277
## brunch      -0.0082421408
## busy         0.0021448124
## décor.x      .
## dance        .
## cheap       -0.0165090011
## ambiance     0.0011745970
## loud        -0.0014987796
## outdoor      .
## authent      .
## modern      -0.0052039584
## cozy         .
## quiet        0.0006888949
## classy       0.0001115154
## upscale      .
## trendy      -0.0092694740
## creative     0.0023663856
## tourist     -0.0031246855
## tradit      -0.0022453545
## sport       -0.0044253225
## amb_pos      .
## neigh_pos   -0.0023630122
## neigh_neg    .
## music.y     -0.0039986956
## food_pos     0.0159088956
## food_neg     .
## serv_pos     0.0170165576
## serv_neg     0.0036192418
## stars        .
## price       -0.0017348549
```

This process was repreated for other romantic variables (i.e. Dating, Boyfriend/girlfriend, family, romantic, intimate, and anniversary). The following table shows the MSE for each variable and model used. The smaller the MSE the more preferable the model for that variable.

A Comparison between different models' performance (MSE) for different romantic variables

Romantic Variables

Linear Least Square

Best Subset

Best Subset (CV K=10)

Lasso

PCR

PLS

Regression Tree

Random Forest

dating

0.0030

0.0030

0.0023

0.0030

0.0030

0.0030

0.0038

0.0029

boyfriend/girlfriend

0.0019

0.0019

0.0018

0.0019

0.0018

0.0019

0.0020

0.0019

wife/husband

0.0074

0.0073

0.0082

0.0072

0.0072

0.0074

0.0089

0.0076

family

0.0137

0.0138

0.0120

0.0136

0.0136

0.0137

0.0187

0.0135

romantic

0.0008

0.0008

0.0006

0.0008

0.0008

0.0008

0.0010

0.0007

intimate

0.0003

0.0002

0.0002

0.0002

0.0003

0.0003

0.0003

0.0002

anniversary

0.00058

0.0005

0.0005

0.0005

0.0005

0.0005

0.0008

0.0004