Ml Workshop

Himank Jain

02/12/2019

Assumptions of Parametric Regression:

- Continuous Response Variable with mean(μ) and variance(σ^2).
- Mean of error terms = 0 with variance(σ^2)
- Linearity in Coefficients ($\beta_0, \beta_1, ...\beta_n$)
- No Multicollinearity (in Multiple Linear Regression)
- No autocorrelation
- Homoscedasticity of errors terms.

Autocorrelaion:

Autocorrelation refers to the degree of correlation between the values of the same variables across different observations in the data. For example, one might expect the air temperature on the 1st day of the month to be more similar to the temperature on the 2nd day compared to the 31st day.

Points to ponder:

- Equality of means of two groups can be tested using two group **t-test**.
- Equality of variances of two groups can be tested using **F-test**

Mean can be postitive or negative. Hence we use t-test since t distribution is symmetric.

Variance can only be positive. Hence we use F-test since it assumes a positively skewed distribution.

$F = \frac{S_X^2}{S_Y^2}$ where S_X, S_Y are sample variances of two groups.

$t = \frac{x_1 - x_2}{\sqrt{\frac{s1^2}{n1} + \frac{s2^2}{n2}}}$ where x_1, x_2 are means and $s1^2, s2^2$ are variances of two groups

Problems with assumption violations:

Error mean & variance:

- Since $\hat{y} = \beta_0 + \beta_1 * x$ the error is:
  $\epsilon = \hat{y} - y = (\beta_0 + \beta_1 * x + \epsilon) - (\beta_0 + \beta_1 * x)$

Variance(ϵ)=variance(y)

- mean of errrors needs to be zero and variance needs to be σ^2.

Multicollinearity Problem:

Multicollinearity is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy i.e. Independent variables are highly Correlated.

β= $(X^tX)^{-1}X^{-1}Y$ Let A=(X^tX) Let A= $\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$ let the two columns of matrix A be two variables. We can see $Col_1 + 3 = Col_2$ hence, they are correlated. then determinant of matrix A $|A|$=0 Hence $A^{-1} = \frac{1}{|A|}adj(A)$ is very large. Therefore even the regression coefficient are very large. This is the problem of multicollinearity.

Heteroscedasticity Problem:

Homoscedasticity is a situation in which the variance of error terms is the same across all values of the independent variables. Heteroscedasticity is quite the opposite. i.e. non constant variance of error terms.

Earlier we deduced that $V(\epsilon) = V(y)$ Hence if variance of errors is not constant neither is the variance of response variable. This violates our first assumption.

Tests for testing normality and it's assumptions:

Tests for checking Normality:

```r
library(datarium)
head(marketing)
```

```
##   youtube facebook newspaper sales
## 1  276.12    45.36     83.04 26.52
## 2   53.40    47.16     54.12 12.48
## 3   20.64    55.08     83.16 11.16
## 4  181.80    49.56     70.20 22.20
## 5  216.96    12.96     70.08 15.48
## 6   10.44    58.68     90.00  8.64
```

Anderson-Darling Test:

```r
library(nortest)
ad.test(marketing$sales)
```

```
##
##  Anderson-Darling normality test
##
## data:  marketing$sales
## A = 1.7373, p-value = 0.0001831
```

p-value is significant. Hence we reject the H_0 that sales is normally distributed.

```r
shapiro.test(marketing$sales)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  marketing$sales
## W = 0.97603, p-value = 0.001683
```

p-value is significant. Hence we reject the H_0 that sales is normally distributed.

- Lilliefors test (Kolmogorov–Smirnov test):

```r
lillie.test(marketing$sales)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  marketing$sales
## D = 0.095172, p-value = 0.000147
```

p-value is significant. Hence we reject the H_0 that sales is normally distributed.

Tests for Multicollinearity:

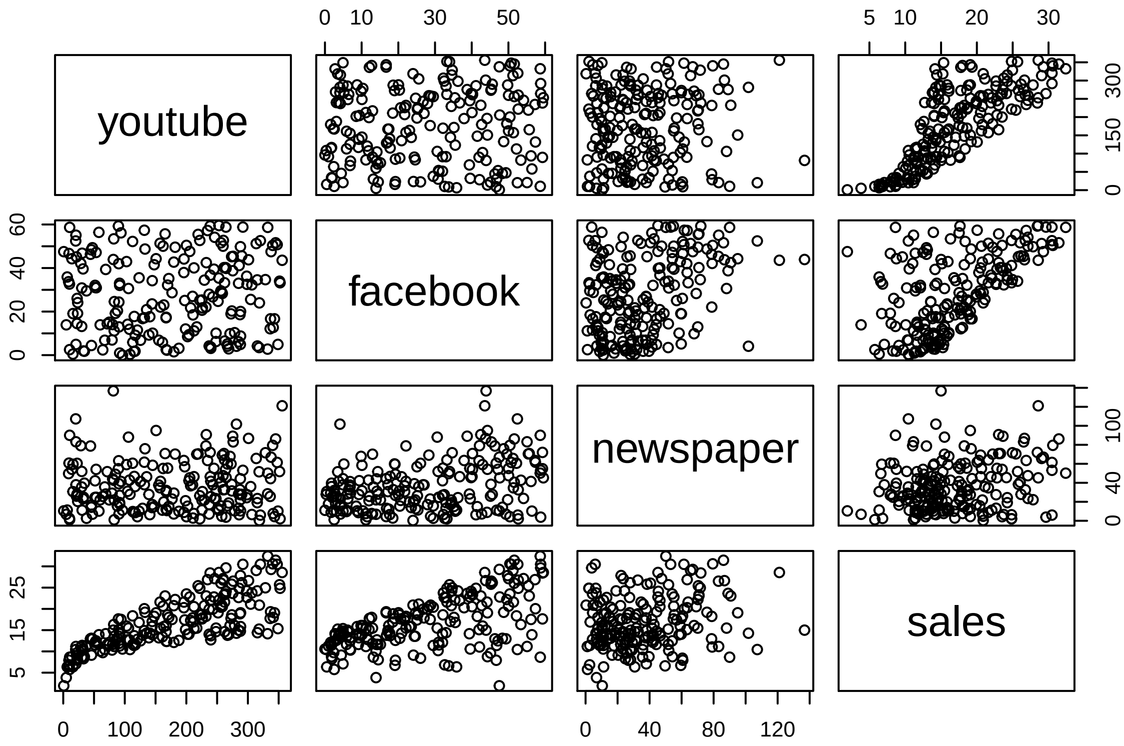
- Eigen values can be used to detect multicollineatiy
- t-test of correlation: $t = \frac{r-0}{S.E.(r)}$ where $S.E.(r) = \sqrt{\frac{1-r^2}{n-2}}$

if t statistic is more than critical value we reject the H_0 that there is no multicollinearity between two variables.

- Variance Inflation Factor(VIF): if $y = \beta_0 + \beta_1 x1 + \beta_2 x2 + ...\beta_n xn$ then Variation inflation factor for jth variable $VIF(x_j)$ is: $VIF(x_j)$=$\frac{1}{1-R_j^2}$ where R_j^2 is coefficient of determination of jth variable if it is regressed on all other independent variables.

i.e. R_1^2=R^2 for $x1 \sim x2 + x3 + x4 + ...xn$ R_2^2=R^2 for $x2 \sim x1 + x3 + x4 + ...xn$ etc.

```r
pairs(marketing)
```




```
library(car)
```

```
## Loading required package: carData
```

```
model=lm(data=marketing,sales~.)
vif(model)
```

```
##   youtube  facebook newspaper
##  1.004611  1.144952  1.145187
```

Tests for Heteroscedasticity:

- Breusch pagan test: After building Model regress residual squares on independent variables and get it's R^2.

$\epsilon^2 \sim \beta_0 + \beta_1 x1 + ...\beta_n xn$ then the test statstic LM is: $LM = NR^2$ if $LM > \chi^2_{df=1}$ then statistic and significant and we reject H_0 that errors are Homoscedastic. "'{r}refors are homoscedastic.

Tests for Autocorrelation:

- Durbin Watson test: DW= $\frac{\sum(\epsilon_t - \epsilon_{t-1})^2}{\sum \epsilon_t^2}$ if DW==2 : No Autocorrelation if DW>2 : -ve Autocorrelation if DW<2 : +ve Autocorrelation

```r
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
dwtest(model)
```

```
##
##  Durbin-Watson test
##
## data:  model
## DW = 2.0836, p-value = 0.7236
## alternative hypothesis: true autocorrelation is greater than 0
```

p-value and test statistic suggest that we fail to reject null hypthesis at 5% level of significant. Therefore no problem of autocorrelation.

How to make response variable normal and satisfy nomral assumptions?

BoxCoxTransforms or other tranformations. The Box-Cox transformation of the variable x is defined as: $\frac{X^{\lambda}-1}{\lambda}$

When normaility isn't satisfied even after transformations it's not feasible to use parametric method for regression like linear regression.

Extra points:

- if n is number of observations and k is number of features(independent variables) then degree of freedom for regression is $df = n - k$
- for SLR: $df = n - k = n - 1$ and F statistic is equal to t statistic. i.e. $F = t^2$
- For regression, t-statistic tells if a regressor is significant and F statistic tells if a groups of regressors are together significant.
- Skewness measures how positively or negatively skewed the distribution is and kurtosis measures how thick or thin tail ends are.

Cross Validation:

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(mlbench)
data("BostonHousing")
bost<-BostonHousing
partition=createDataPartition(y=bost$medv,p = 0.7,list=F)
trainbost<- bost[partition,]
testbost<- bost[-partition,]
trcon<-trainControl(method='repeatedcv',number=10,repeats = 3)
lmodel<-train(data=trainbost,medv~.,method='lm',trControl=trcon)
lmodel
```

```
## Linear Regression
##
## 356 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 321, 321, 320, 321, 321, 320, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   5.084731  0.7087095  3.660959
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
R2(testbost$medv,predict(lmodel,testbost))
```

```
## [1] 0.7683103
```

```r
RMSE(testbost$medv,predict(lmodel,testbost))
```

```
## [1] 4.25246
```

PCA:

PCA is used for dimension reduction and removing Multicollinearity.

```r
library(car)
part=createDataPartition(y=mtcars$mpg,p=0.7,list=FALSE)
traincars=mtcars[part,]
testcars=mtcars[-part,]
model=lm(data=traincars,mpg~.)
vif(model)
```

```
##       cyl      disp        hp      drat        wt      qsec        vs        am
## 14.298250 23.940084 13.590619  3.666815 14.916759  8.114786  4.454477  4.239519
##      gear      carb
##  5.947773 10.671677
```

We can observe large multicollinearity due to coefficient inflation.

```r
pc<-prcomp(traincars[,-1],center=T,scale.=T)
summary(pc)
```

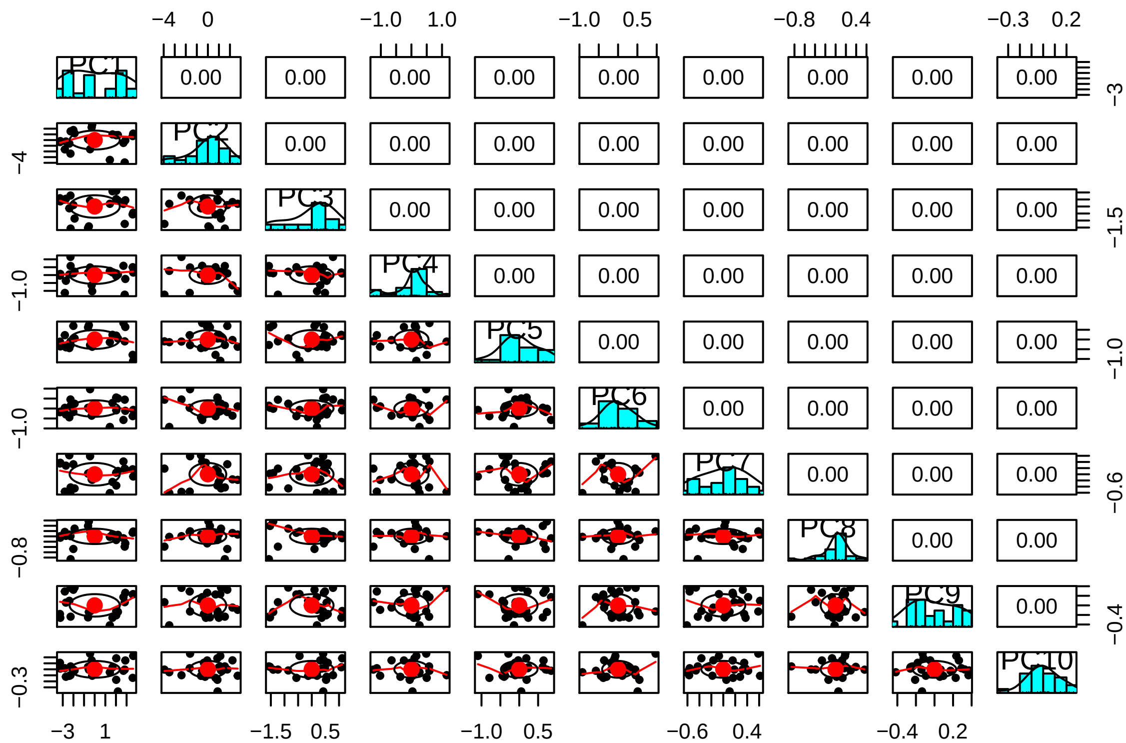
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.3609 1.6651 0.79978 0.56000 0.48457 0.41408 0.36810
## Proportion of Variance 0.5574 0.2772 0.06396 0.03136 0.02348 0.01715 0.01355
## Cumulative Proportion  0.5574 0.8346 0.89860 0.92996 0.95345 0.97059 0.98414
##                            PC8     PC9    PC10
## Standard deviation     0.28953 0.23486 0.14000
## Proportion of Variance 0.00838 0.00552 0.00196
## Cumulative Proportion  0.99252 0.99804 1.00000
```

First 5 PC's explain more than 95% variance.

```r
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

## The following object is masked from 'package:car':
##
##     logit
```

```r
pairs.panels(pc$x)
```



```r
pctrain<-predict(pc,newdata=traincars)
pctrain<-as.data.frame(pctrain)
pctrain$mpg=traincars$mpg
lmodelmpg<-lm(mpg~PC1+PC2+PC3+PC4+PC5,data=pctrain)
lmodelmpg
```

```
##
## Call:
## lm(formula = mpg ~ PC1 + PC2 + PC3 + PC4 + PC5, data = pctrain)
##
## Coefficients:
## (Intercept)          PC1          PC2          PC3          PC4          PC5
##     19.9000      -2.0770      -0.2750       1.0940      -0.5357      -0.3774
```

```r
vif(lmodelmpg)
```

```
## PC1 PC2 PC3 PC4 PC5
##   1   1   1   1   1
```

Hence no multicollinearity and no variance inflation in model.

```r
trainpred=predict(lmodelmpg,newdata=pctrain)
RMSE(trainpred,pctrain$mpg)
```

```
## [1] 2.132845
```

```r
pctestdata=predict(pc,newdata = testcars)
testpred=predict(lmodelmpg,data.frame(pctestdata))
RMSE(testpred,testcars$mpg)
```

```
## [1] 2.845215
```

```r
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
get_eigenvalue(pc)
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1  5.57395573       55.7395573                    55.73956
## Dim.2  2.77244953       27.7244953                    83.46405
## Dim.3  0.63964373        6.3964373                    89.86049
## Dim.4  0.31360013        3.1360013                    92.99649
## Dim.5  0.23480837        2.3480837                    95.34457
## Dim.6  0.17146147        1.7146147                    97.05919
## Dim.7  0.13549428        1.3549428                    98.41413
## Dim.8  0.08382742        0.8382742                    99.25241
## Dim.9  0.05515854        0.5515854                    99.80399
## Dim.10 0.01960081        0.1960081                   100.00000
```

we can also keep only eigen values more than 1 still doing a good job with out model.

Preventing Overfitting

Increase Bias and reduce Variance of model. Introduces penalty term for large coefficients. Three types: * Ridge regression $\epsilon^2 + \lambda \sum \beta_i^2$ * Lasso regression $\epsilon^2 + \lambda \sum |\beta_i|$ * Elastic regression $\epsilon^2 + \lambda((1-\alpha)\sum \beta_i^2 + \alpha \sum |\beta_i|)$

```r
# ridge
rlm<-train(mpg~.,data=traincars,method='glmnet',trControl=trcon,tuneGrid=expand.grid(alpha=0,lambda=seq
```

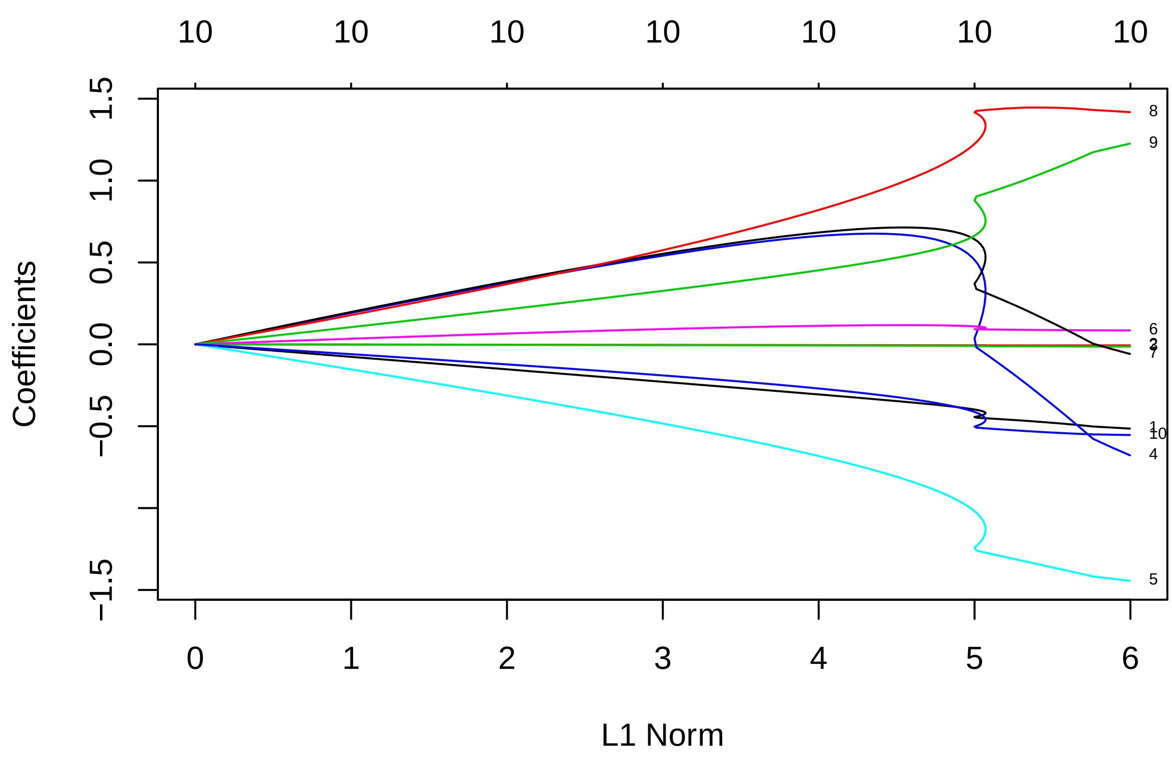
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
rlm
```

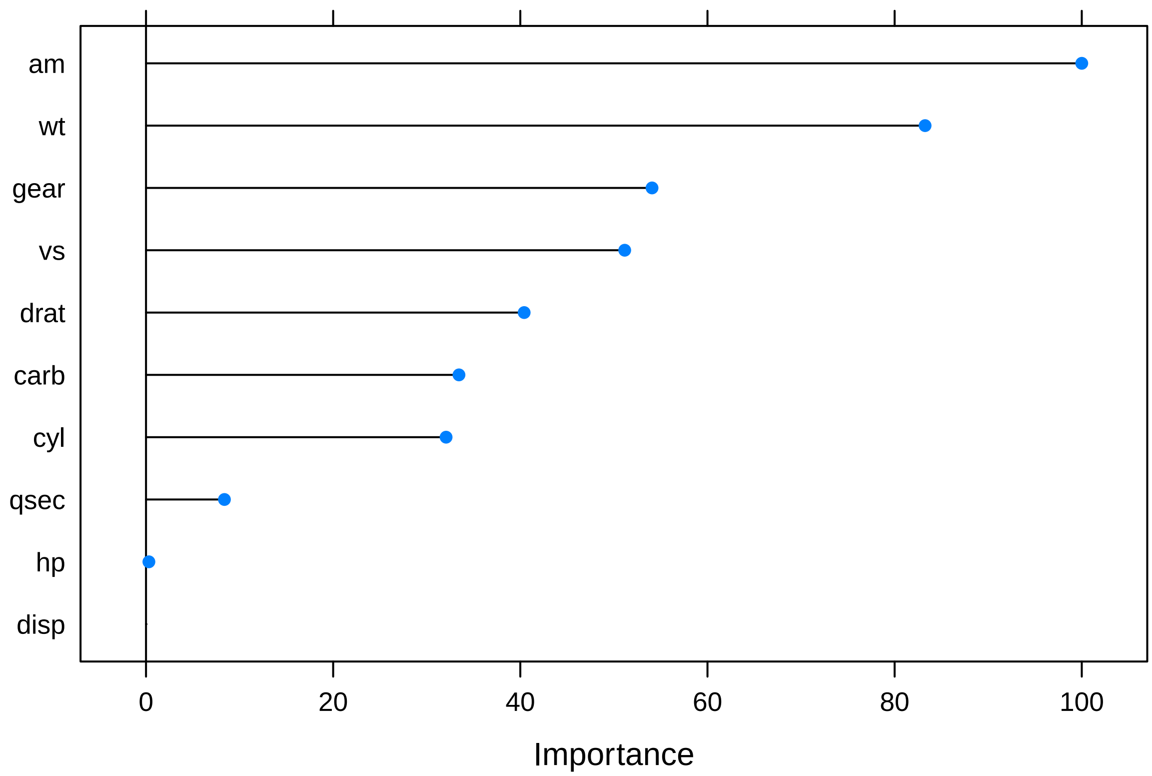
```
## glmnet
##
## 24 samples
## 10 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 22, 21, 22, 21, 22, 22, ...
```

```
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##    1      2.575042  0.9170491  2.201784
##    2      2.490878  0.9175331  2.158537
##    3      2.463089  0.9173511  2.156429
##    4      2.455830  0.9171074  2.159900
##    5      2.459598  0.9168966  2.169089
##    6      2.470486  0.9167302  2.181038
##    7      2.486560  0.9166027  2.193411
##    8      2.506747  0.9165058  2.206187
##    9      2.529971  0.9164368  2.221688
##   10      2.555653  0.9163893  2.245974
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 4.
```

```r
plot(rlm$finalModel,label=T)
```



```r
plot(varImp(rlm))
```




```r
# Lasso
llm<-train(mpg~.,data=traincars,method='glmnet',trControl=trcon,tuneGrid=expand.grid(alpha=1,lambda=seq
```

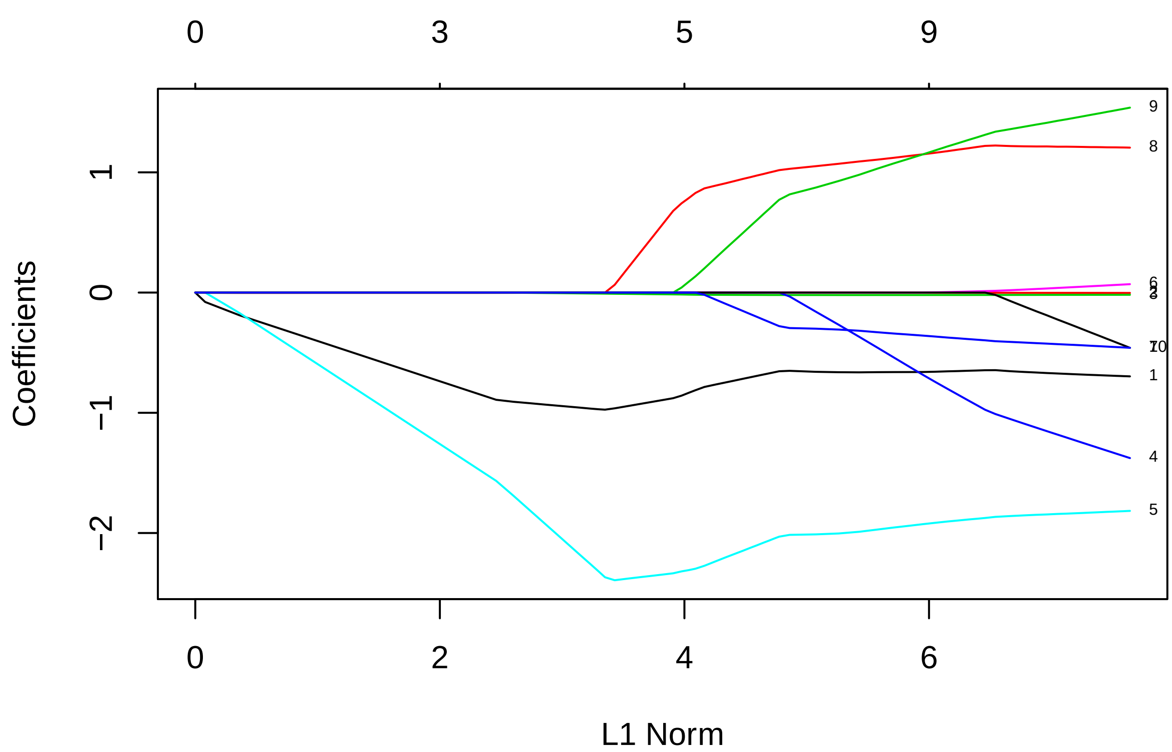
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
llm
```

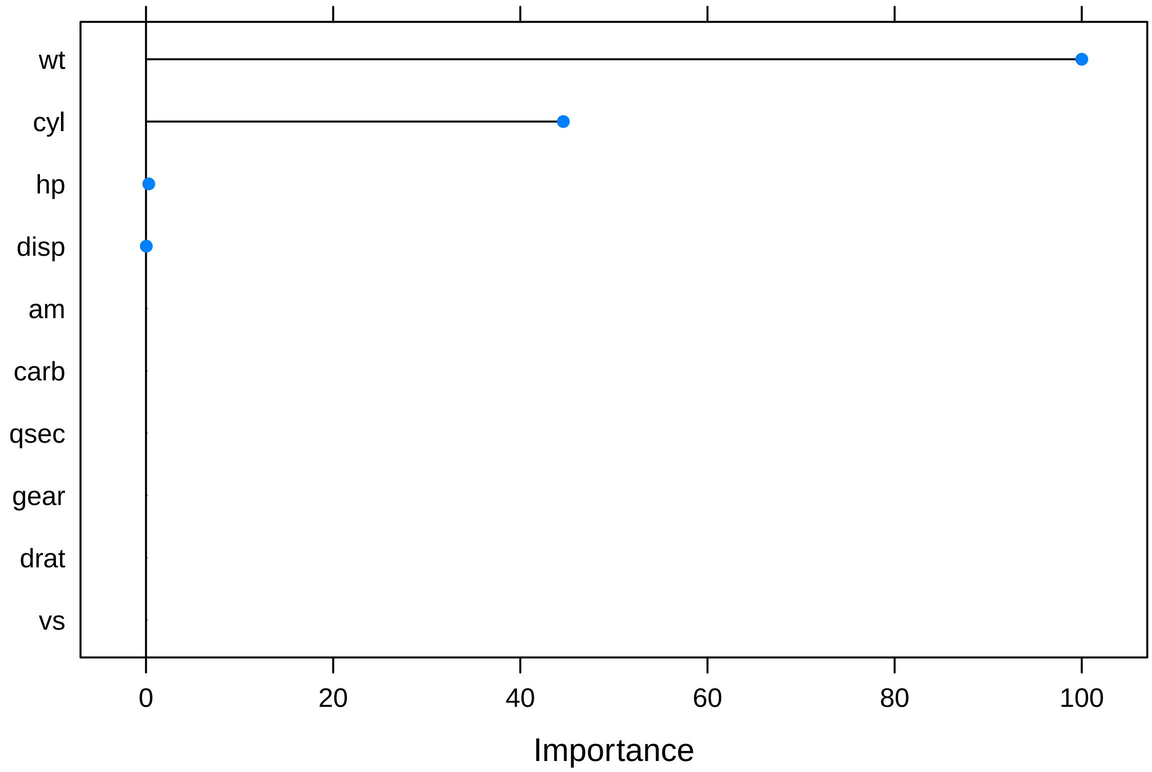
```
## glmnet
##
## 24 samples
## 10 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 22, 22, 21, 21, 22, 22, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##    1      2.571357  0.9468892  2.288085
##    2      3.073220  0.9451063  2.657090
##    3      3.753801  0.9324425  3.268163
```

```
##    4      4.550115  0.9264674  4.028383
##    5      5.025528        NaN  4.496484
##    6      5.025528        NaN  4.496484
##    7      5.025528        NaN  4.496484
##    8      5.025528        NaN  4.496484
##    9      5.025528        NaN  4.496484
##   10      5.025528        NaN  4.496484
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 1.
```

```r
plot(llm$finalModel,label=T)
```




```r
plot(varImp(llm))
```




```r
# Elastic net
elasticlm<-train(mpg~.,data=traincars,method='glmnet',trControl=trcon,tuneGrid=expand.grid(alpha=seq(0,
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

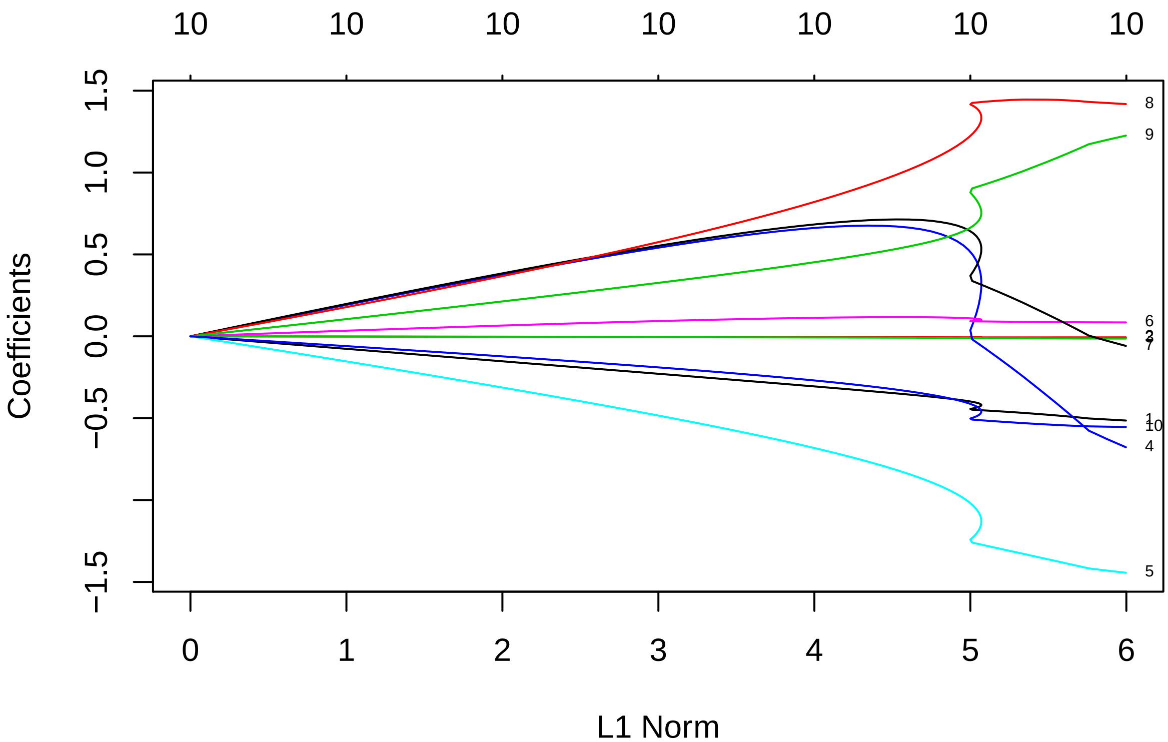
```r
elasticlm
```

```
## glmnet
##
## 24 samples
## 10 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 21, 22, 22, 21, 21, 22, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  RMSE      Rsquared   MAE
##   0.0     1      2.576327  0.9275710  2.234514
##   0.0     2      2.510662  0.9267075  2.194547
##   0.0     3      2.492066  0.9271527  2.190155
##   0.0     4      2.490887  0.9274992  2.191478
##   0.0     5      2.498997  0.9277253  2.195198
##   0.0     6      2.512800  0.9278650  2.206620
```

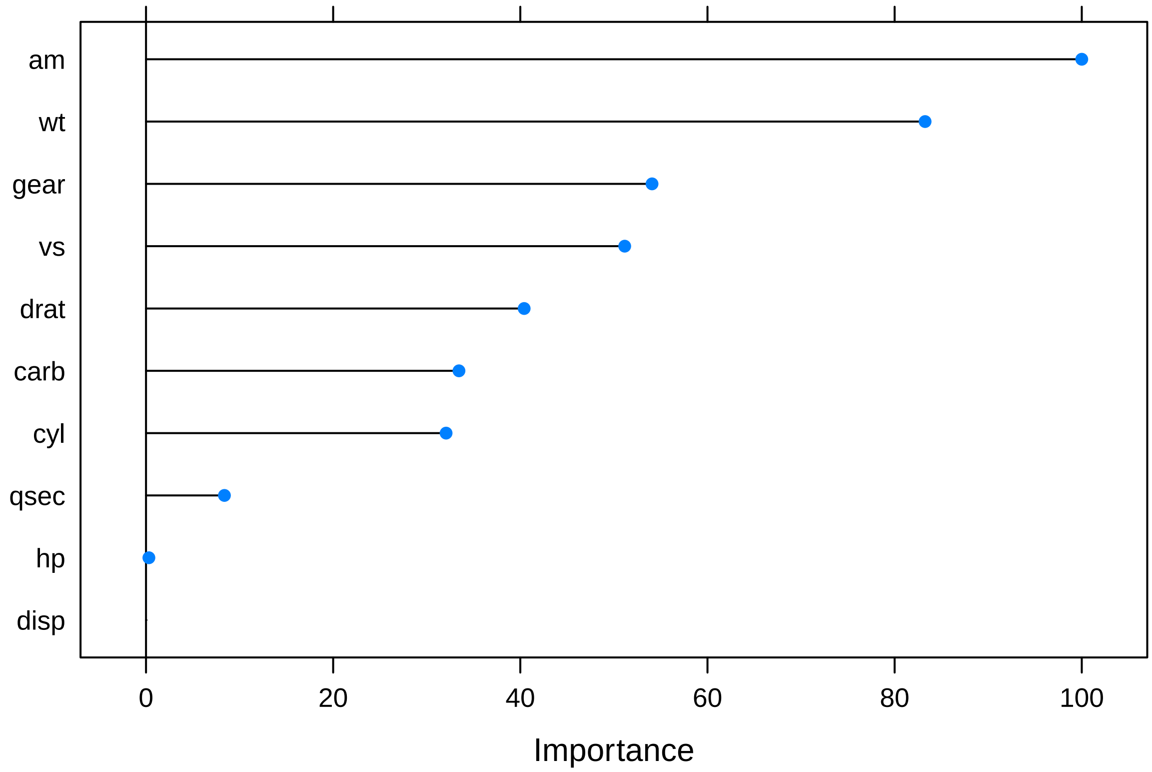
```
## 0.0  7   2.530532 0.9279535 2.221591
## 0.0  8   2.551010 0.9280091 2.238136
## 0.0  9   2.573512 0.9280444 2.255050
## 0.0 10   2.597534 0.9280642 2.273539
## 0.1  1   2.552505 0.9272361 2.222763
## 0.1  2   2.516626 0.9245792 2.217705
## 0.1  3   2.541160 0.9246320 2.247212
## 0.1  4   2.583230 0.9247240 2.293277
## 0.1  5   2.636197 0.9250795 2.342551
## 0.1  6   2.698885 0.9252695 2.394589
## 0.1  7   2.769354 0.9253583 2.445561
## 0.1  8   2.846151 0.9253863 2.496599
## 0.1  9   2.928360 0.9253534 2.556956
## 0.1 10   3.015118 0.9252836 2.630789
## 0.2  1   2.553656 0.9290400 2.235962
## 0.2  2   2.546874 0.9248043 2.250413
## 0.2  3   2.615469 0.9238701 2.328035
## 0.2  4   2.715270 0.9236707 2.419340
## 0.2  5   2.833881 0.9237224 2.512508
## 0.2  6   2.965138 0.9248842 2.605870
## 0.2  7   3.110410 0.9264777 2.711745
## 0.2  8   3.266629 0.9280547 2.852672
## 0.2  9   3.422266 0.9297793 2.989628
## 0.2 10   3.571778 0.9316330 3.115394
## 0.3  1   2.572821 0.9299919 2.262450
## 0.3  2   2.599836 0.9267072 2.305201
## 0.3  3   2.708289 0.9281930 2.413929
## 0.3  4   2.858177 0.9321634 2.532412
## 0.3  5   3.030521 0.9356734 2.644369
## 0.3  6   3.224831 0.9383252 2.806132
## 0.3  7   3.441922 0.9413416 3.004020
## 0.3  8   3.670496 0.9433023 3.199553
## 0.3  9   3.899781 0.9430731 3.392166
## 0.3 10   4.129118 0.9413792 3.611642
## 0.4  1   2.605588 0.9265781 2.301076
## 0.4  2   2.651365 0.9332543 2.362531
## 0.4  3   2.794629 0.9408526 2.478027
## 0.4  4   2.994061 0.9429792 2.617223
## 0.4  5   3.254901 0.9439943 2.832405
## 0.4  6   3.559943 0.9421972 3.108533
## 0.4  7   3.872495 0.9396978 3.371806
## 0.4  8   4.182825 0.9355265 3.661120
## 0.4  9   4.492613 0.9265080 3.969993
## 0.4 10   4.779358 0.9181984 4.252213
## 0.5  1   2.626825 0.9255540 2.326213
## 0.5  2   2.693841 0.9392747 2.400243
## 0.5  3   2.889709 0.9430933 2.543992
## 0.5  4   3.191045 0.9418592 2.769409
## 0.5  5   3.566064 0.9384850 3.115410
## 0.5  6   3.956319 0.9329971 3.445253
## 0.5  7   4.355387 0.9241211 3.829632
## 0.5  8   4.744824 0.9150133 4.216791
## 0.5  9   5.028048 0.9060837 4.496828
## 0.5 10   5.114037       NaN 4.582318
```

```
##   0.6     1      2.637703  0.9304312  2.348683
##   0.6     2      2.739310  0.9414915  2.427010
##   0.6     3      3.028588  0.9404905  2.645550
##   0.6     4      3.437672  0.9353729  3.000895
##   0.6     5      3.904080  0.9269889  3.404330
##   0.6     6      4.392458  0.9207800  3.863821
##   0.6     7      4.865672  0.9094090  4.335542
##   0.6     8      5.106309  0.8253496  4.574386
##   0.6     9      5.114037        NaN  4.582318
##   0.6    10      5.114037        NaN  4.582318
##   0.7     1      2.634539  0.9367843  2.356640
##   0.7     2      2.810141  0.9413204  2.474281
##   0.7     3      3.195066  0.9342121  2.779200
##   0.7     4      3.705036  0.9259145  3.239999
##   0.7     5      4.273461  0.9197644  3.742110
##   0.7     6      4.852109  0.9036667  4.321162
##   0.7     7      5.113345  0.7127521  4.581615
##   0.7     8      5.114037        NaN  4.582318
##   0.7     9      5.114037        NaN  4.582318
##   0.7    10      5.114037        NaN  4.582318
##   0.8     1      2.634864  0.9401896  2.357018
##   0.8     2      2.910323  0.9367190  2.549700
##   0.8     3      3.373279  0.9278164  2.936674
##   0.8     4      3.998610  0.9208147  3.487718
##   0.8     5      4.680743  0.9004463  4.147285
##   0.8     6      5.105730  0.8092811  4.573841
##   0.8     7      5.114037        NaN  4.582318
##   0.8     8      5.114037        NaN  4.582318
##   0.8     9      5.114037        NaN  4.582318
##   0.8    10      5.114037        NaN  4.582318
##   0.9     1      2.647149  0.9434396  2.362388
##   0.9     2      3.012304  0.9327382  2.627161
##   0.9     3      3.579333  0.9229378  3.128808
##   0.9     4      4.319488  0.9065430  3.779536
##   0.9     5      5.004068  0.8923242  4.472798
##   0.9     6      5.114037        NaN  4.582318
##   0.9     7      5.114037        NaN  4.582318
##   0.9     8      5.114037        NaN  4.582318
##   0.9     9      5.114037        NaN  4.582318
##   0.9    10      5.114037        NaN  4.582318
##   1.0     1      2.668821  0.9451526  2.377377
##   1.0     2      3.113428  0.9290015  2.710460
##   1.0     3      3.808454  0.9181082  3.329073
##   1.0     4      4.656207  0.8819746  4.118425
##   1.0     5      5.114037        NaN  4.582318
##   1.0     6      5.114037        NaN  4.582318
##   1.0     7      5.114037        NaN  4.582318
##   1.0     8      5.114037        NaN  4.582318
##   1.0     9      5.114037        NaN  4.582318
##   1.0    10      5.114037        NaN  4.582318
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 4.
```

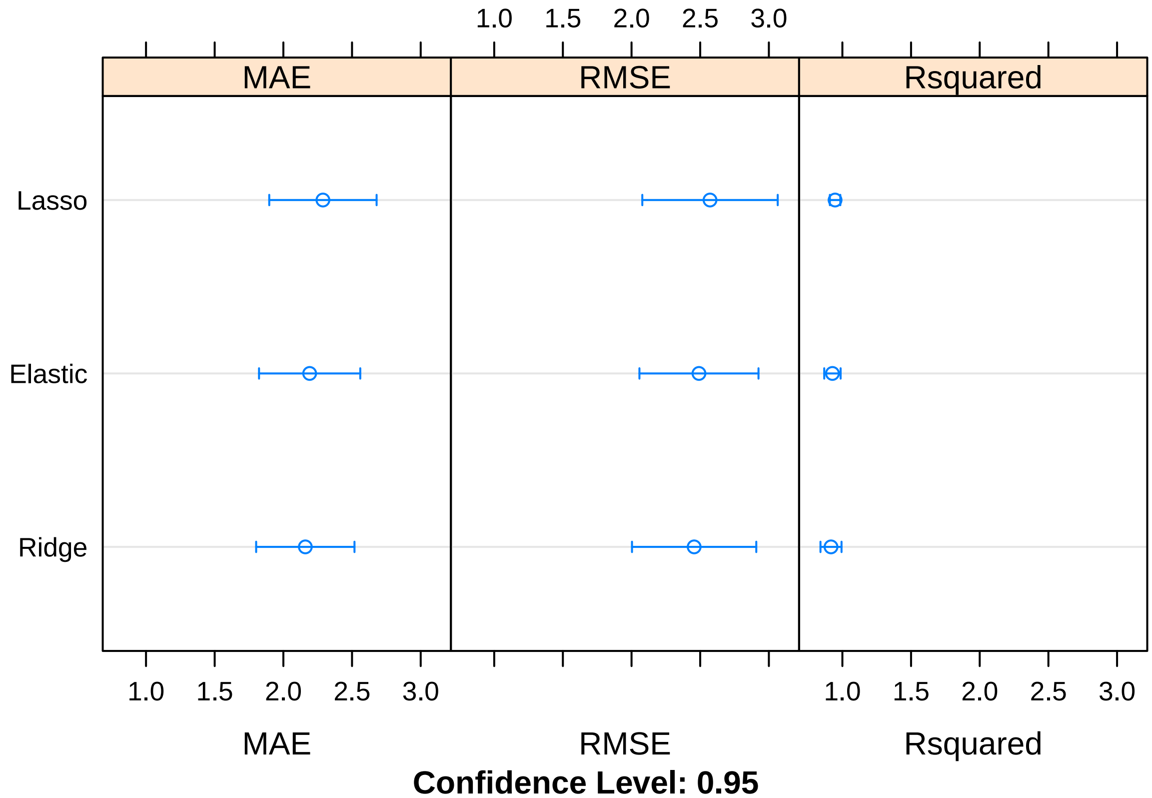
```r
plot(elasticlm$finalModel,label=T)
```



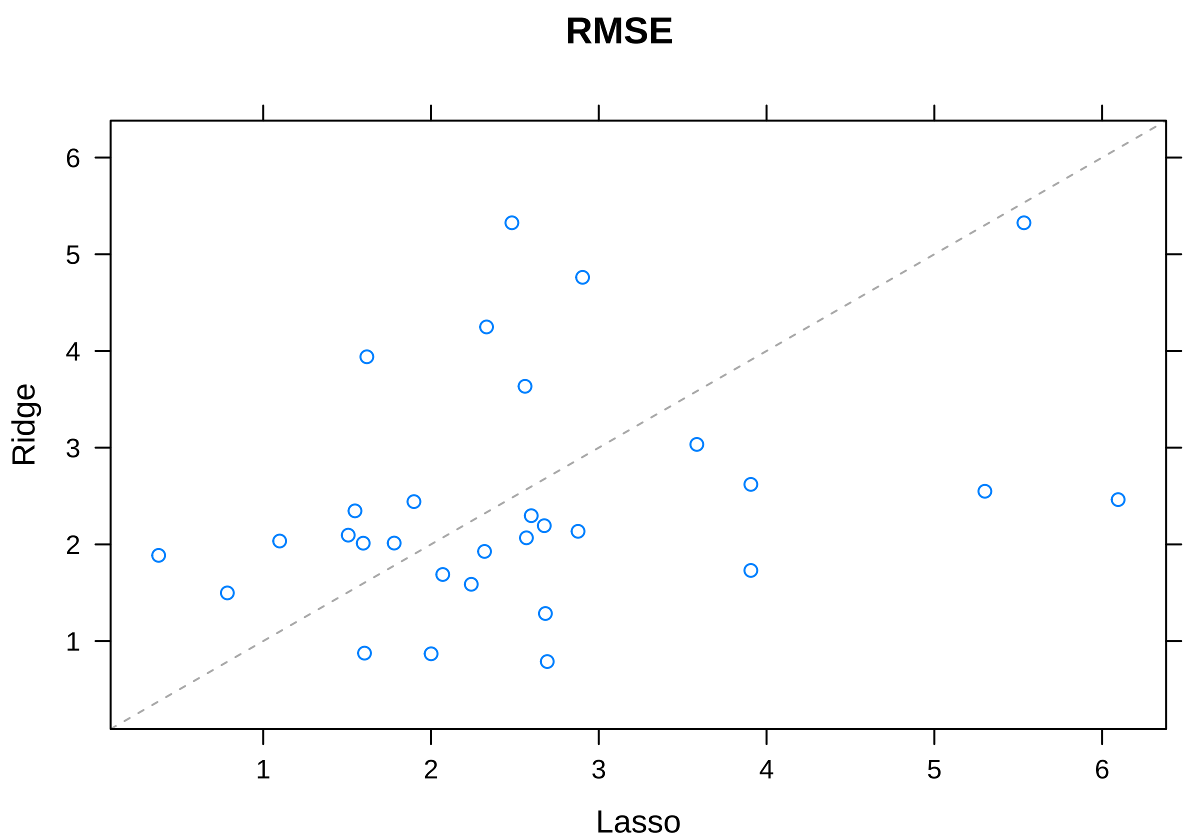

```r
plot(varImp(elasticlm))
```




```
resultlist<-list(Ridge=rlm,Lasso=llm,Elastic=elasticlm)

final<-resamples(resultlist)
dotplot(final)
```




```
xyplot(final,lwd=10,metric="RMSE")
```



therefore ridge works best for least rmse.

Missing values:

```
library(mice)
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
irisdata<-iris
index=sample(seq(1,150,1),15)
irisdata$Sepal.Length[index]=NA
```

```r
index=sample(seq(1,150,1),15)
irisdata$Petal.Length[index]=NA
sum(is.na(irisdata))
```

```
## [1] 30
```

```r
colSums((is.na(irisdata)))
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##           15            0           15            0            0
```

```r
imputed<-mice(irisdata,m=5,method='pmm',maxit=50)
```

```
##
##  iter imp variable
##   1   1  Sepal.Length  Petal.Length
##   1   2  Sepal.Length  Petal.Length
##   1   3  Sepal.Length  Petal.Length
##   1   4  Sepal.Length  Petal.Length
##   1   5  Sepal.Length  Petal.Length
##   2   1  Sepal.Length  Petal.Length
##   2   2  Sepal.Length  Petal.Length
##   2   3  Sepal.Length  Petal.Length
##   2   4  Sepal.Length  Petal.Length
##   2   5  Sepal.Length  Petal.Length
##   3   1  Sepal.Length  Petal.Length
##   3   2  Sepal.Length  Petal.Length
##   3   3  Sepal.Length  Petal.Length
##   3   4  Sepal.Length  Petal.Length
##   3   5  Sepal.Length  Petal.Length
##   4   1  Sepal.Length  Petal.Length
##   4   2  Sepal.Length  Petal.Length
##   4   3  Sepal.Length  Petal.Length
##   4   4  Sepal.Length  Petal.Length
##   4   5  Sepal.Length  Petal.Length
##   5   1  Sepal.Length  Petal.Length
##   5   2  Sepal.Length  Petal.Length
##   5   3  Sepal.Length  Petal.Length
##   5   4  Sepal.Length  Petal.Length
##   5   5  Sepal.Length  Petal.Length
##   6   1  Sepal.Length  Petal.Length
##   6   2  Sepal.Length  Petal.Length
##   6   3  Sepal.Length  Petal.Length
##   6   4  Sepal.Length  Petal.Length
##   6   5  Sepal.Length  Petal.Length
##   7   1  Sepal.Length  Petal.Length
##   7   2  Sepal.Length  Petal.Length
##   7   3  Sepal.Length  Petal.Length
##   7   4  Sepal.Length  Petal.Length
##   7   5  Sepal.Length  Petal.Length
##   8   1  Sepal.Length  Petal.Length
##   8   2  Sepal.Length  Petal.Length
```

```
##   8   3  Sepal.Length  Petal.Length
##   8   4  Sepal.Length  Petal.Length
##   8   5  Sepal.Length  Petal.Length
##   9   1  Sepal.Length  Petal.Length
##   9   2  Sepal.Length  Petal.Length
##   9   3  Sepal.Length  Petal.Length
##   9   4  Sepal.Length  Petal.Length
##   9   5  Sepal.Length  Petal.Length
##   10   1  Sepal.Length  Petal.Length
##   10   2  Sepal.Length  Petal.Length
##   10   3  Sepal.Length  Petal.Length
##   10   4  Sepal.Length  Petal.Length
##   10   5  Sepal.Length  Petal.Length
##   11   1  Sepal.Length  Petal.Length
##   11   2  Sepal.Length  Petal.Length
##   11   3  Sepal.Length  Petal.Length
##   11   4  Sepal.Length  Petal.Length
##   11   5  Sepal.Length  Petal.Length
##   12   1  Sepal.Length  Petal.Length
##   12   2  Sepal.Length  Petal.Length
##   12   3  Sepal.Length  Petal.Length
##   12   4  Sepal.Length  Petal.Length
##   12   5  Sepal.Length  Petal.Length
##   13   1  Sepal.Length  Petal.Length
##   13   2  Sepal.Length  Petal.Length
##   13   3  Sepal.Length  Petal.Length
##   13   4  Sepal.Length  Petal.Length
##   13   5  Sepal.Length  Petal.Length
##   14   1  Sepal.Length  Petal.Length
##   14   2  Sepal.Length  Petal.Length
##   14   3  Sepal.Length  Petal.Length
##   14   4  Sepal.Length  Petal.Length
##   14   5  Sepal.Length  Petal.Length
##   15   1  Sepal.Length  Petal.Length
##   15   2  Sepal.Length  Petal.Length
##   15   3  Sepal.Length  Petal.Length
##   15   4  Sepal.Length  Petal.Length
##   15   5  Sepal.Length  Petal.Length
##   16   1  Sepal.Length  Petal.Length
##   16   2  Sepal.Length  Petal.Length
##   16   3  Sepal.Length  Petal.Length
##   16   4  Sepal.Length  Petal.Length
##   16   5  Sepal.Length  Petal.Length
##   17   1  Sepal.Length  Petal.Length
##   17   2  Sepal.Length  Petal.Length
##   17   3  Sepal.Length  Petal.Length
##   17   4  Sepal.Length  Petal.Length
##   17   5  Sepal.Length  Petal.Length
##   18   1  Sepal.Length  Petal.Length
##   18   2  Sepal.Length  Petal.Length
##   18   3  Sepal.Length  Petal.Length
##   18   4  Sepal.Length  Petal.Length
##   18   5  Sepal.Length  Petal.Length
##   19   1  Sepal.Length  Petal.Length
```

```
## 19 2 Sepal.Length Petal.Length
## 19 3 Sepal.Length Petal.Length
## 19 4 Sepal.Length Petal.Length
## 19 5 Sepal.Length Petal.Length
## 20 1 Sepal.Length Petal.Length
## 20 2 Sepal.Length Petal.Length
## 20 3 Sepal.Length Petal.Length
## 20 4 Sepal.Length Petal.Length
## 20 5 Sepal.Length Petal.Length
## 21 1 Sepal.Length Petal.Length
## 21 2 Sepal.Length Petal.Length
## 21 3 Sepal.Length Petal.Length
## 21 4 Sepal.Length Petal.Length
## 21 5 Sepal.Length Petal.Length
## 22 1 Sepal.Length Petal.Length
## 22 2 Sepal.Length Petal.Length
## 22 3 Sepal.Length Petal.Length
## 22 4 Sepal.Length Petal.Length
## 22 5 Sepal.Length Petal.Length
## 23 1 Sepal.Length Petal.Length
## 23 2 Sepal.Length Petal.Length
## 23 3 Sepal.Length Petal.Length
## 23 4 Sepal.Length Petal.Length
## 23 5 Sepal.Length Petal.Length
## 24 1 Sepal.Length Petal.Length
## 24 2 Sepal.Length Petal.Length
## 24 3 Sepal.Length Petal.Length
## 24 4 Sepal.Length Petal.Length
## 24 5 Sepal.Length Petal.Length
## 25 1 Sepal.Length Petal.Length
## 25 2 Sepal.Length Petal.Length
## 25 3 Sepal.Length Petal.Length
## 25 4 Sepal.Length Petal.Length
## 25 5 Sepal.Length Petal.Length
## 26 1 Sepal.Length Petal.Length
## 26 2 Sepal.Length Petal.Length
## 26 3 Sepal.Length Petal.Length
## 26 4 Sepal.Length Petal.Length
## 26 5 Sepal.Length Petal.Length
## 27 1 Sepal.Length Petal.Length
## 27 2 Sepal.Length Petal.Length
## 27 3 Sepal.Length Petal.Length
## 27 4 Sepal.Length Petal.Length
## 27 5 Sepal.Length Petal.Length
## 28 1 Sepal.Length Petal.Length
## 28 2 Sepal.Length Petal.Length
## 28 3 Sepal.Length Petal.Length
## 28 4 Sepal.Length Petal.Length
## 28 5 Sepal.Length Petal.Length
## 29 1 Sepal.Length Petal.Length
## 29 2 Sepal.Length Petal.Length
## 29 3 Sepal.Length Petal.Length
## 29 4 Sepal.Length Petal.Length
## 29 5 Sepal.Length Petal.Length
```

```
## 30 1 Sepal.Length Petal.Length
## 30 2 Sepal.Length Petal.Length
## 30 3 Sepal.Length Petal.Length
## 30 4 Sepal.Length Petal.Length
## 30 5 Sepal.Length Petal.Length
## 31 1 Sepal.Length Petal.Length
## 31 2 Sepal.Length Petal.Length
## 31 3 Sepal.Length Petal.Length
## 31 4 Sepal.Length Petal.Length
## 31 5 Sepal.Length Petal.Length
## 32 1 Sepal.Length Petal.Length
## 32 2 Sepal.Length Petal.Length
## 32 3 Sepal.Length Petal.Length
## 32 4 Sepal.Length Petal.Length
## 32 5 Sepal.Length Petal.Length
## 33 1 Sepal.Length Petal.Length
## 33 2 Sepal.Length Petal.Length
## 33 3 Sepal.Length Petal.Length
## 33 4 Sepal.Length Petal.Length
## 33 5 Sepal.Length Petal.Length
## 34 1 Sepal.Length Petal.Length
## 34 2 Sepal.Length Petal.Length
## 34 3 Sepal.Length Petal.Length
## 34 4 Sepal.Length Petal.Length
## 34 5 Sepal.Length Petal.Length
## 35 1 Sepal.Length Petal.Length
## 35 2 Sepal.Length Petal.Length
## 35 3 Sepal.Length Petal.Length
## 35 4 Sepal.Length Petal.Length
## 35 5 Sepal.Length Petal.Length
## 36 1 Sepal.Length Petal.Length
## 36 2 Sepal.Length Petal.Length
## 36 3 Sepal.Length Petal.Length
## 36 4 Sepal.Length Petal.Length
## 36 5 Sepal.Length Petal.Length
## 37 1 Sepal.Length Petal.Length
## 37 2 Sepal.Length Petal.Length
## 37 3 Sepal.Length Petal.Length
## 37 4 Sepal.Length Petal.Length
## 37 5 Sepal.Length Petal.Length
## 38 1 Sepal.Length Petal.Length
## 38 2 Sepal.Length Petal.Length
## 38 3 Sepal.Length Petal.Length
## 38 4 Sepal.Length Petal.Length
## 38 5 Sepal.Length Petal.Length
## 39 1 Sepal.Length Petal.Length
## 39 2 Sepal.Length Petal.Length
## 39 3 Sepal.Length Petal.Length
## 39 4 Sepal.Length Petal.Length
## 39 5 Sepal.Length Petal.Length
## 40 1 Sepal.Length Petal.Length
## 40 2 Sepal.Length Petal.Length
## 40 3 Sepal.Length Petal.Length
## 40 4 Sepal.Length Petal.Length
```

```
##   40   5  Sepal.Length  Petal.Length
##   41   1  Sepal.Length  Petal.Length
##   41   2  Sepal.Length  Petal.Length
##   41   3  Sepal.Length  Petal.Length
##   41   4  Sepal.Length  Petal.Length
##   41   5  Sepal.Length  Petal.Length
##   42   1  Sepal.Length  Petal.Length
##   42   2  Sepal.Length  Petal.Length
##   42   3  Sepal.Length  Petal.Length
##   42   4  Sepal.Length  Petal.Length
##   42   5  Sepal.Length  Petal.Length
##   43   1  Sepal.Length  Petal.Length
##   43   2  Sepal.Length  Petal.Length
##   43   3  Sepal.Length  Petal.Length
##   43   4  Sepal.Length  Petal.Length
##   43   5  Sepal.Length  Petal.Length
##   44   1  Sepal.Length  Petal.Length
##   44   2  Sepal.Length  Petal.Length
##   44   3  Sepal.Length  Petal.Length
##   44   4  Sepal.Length  Petal.Length
##   44   5  Sepal.Length  Petal.Length
##   45   1  Sepal.Length  Petal.Length
##   45   2  Sepal.Length  Petal.Length
##   45   3  Sepal.Length  Petal.Length
##   45   4  Sepal.Length  Petal.Length
##   45   5  Sepal.Length  Petal.Length
##   46   1  Sepal.Length  Petal.Length
##   46   2  Sepal.Length  Petal.Length
##   46   3  Sepal.Length  Petal.Length
##   46   4  Sepal.Length  Petal.Length
##   46   5  Sepal.Length  Petal.Length
##   47   1  Sepal.Length  Petal.Length
##   47   2  Sepal.Length  Petal.Length
##   47   3  Sepal.Length  Petal.Length
##   47   4  Sepal.Length  Petal.Length
##   47   5  Sepal.Length  Petal.Length
##   48   1  Sepal.Length  Petal.Length
##   48   2  Sepal.Length  Petal.Length
##   48   3  Sepal.Length  Petal.Length
##   48   4  Sepal.Length  Petal.Length
##   48   5  Sepal.Length  Petal.Length
##   49   1  Sepal.Length  Petal.Length
##   49   2  Sepal.Length  Petal.Length
##   49   3  Sepal.Length  Petal.Length
##   49   4  Sepal.Length  Petal.Length
##   49   5  Sepal.Length  Petal.Length
##   50   1  Sepal.Length  Petal.Length
##   50   2  Sepal.Length  Petal.Length
##   50   3  Sepal.Length  Petal.Length
##   50   4  Sepal.Length  Petal.Length
##   50   5  Sepal.Length  Petal.Length
```

```r
fulldata1<-complete(imputed,1)
head(fulldata1)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.6         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.5         0.4  setosa
```

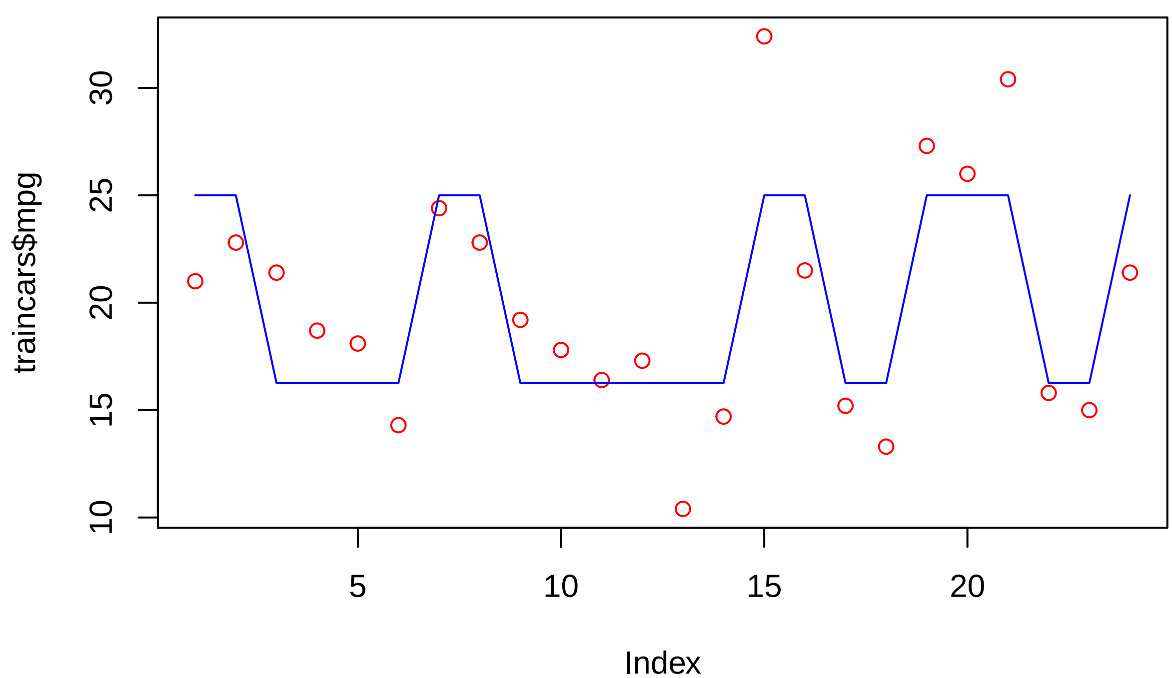
```r
sum(is.na(fulldata1))
```

```
## [1] 0
```

All missing values impuced using mice.

Regression Tree:

```r
library(rpart)
rt<-rpart(mpg~.,data=traincars,method='anova')
predtrain<-predict(rt,newdata=traincars)
plot(traincars$mpg,col='red')
lines(predtrain,col='blue')
```



```r
RMSE(predtrain,traincars$mpg)
```

```
## [1] 3.173026
```

KNN:

```r
knnmedv<-train(medv~.,data=trainbost[,-15],
               method="knn",
               trControl=trcon,
               metric='RMSE',
               preProc=c('center','scale'),
               tuneGrid=expand.grid(k=c(1:20)))
knnmedv
```

```
## k-Nearest Neighbors
##
## 356 samples
##  13 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 320, 322, 320, 320, 320, 320, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    1  4.940883  0.7210763  3.170421
##    2  4.570209  0.7472235  2.987262
##    3  4.771874  0.7327316  3.084740
##    4  4.965988  0.7218284  3.184009
##    5  5.042121  0.7172340  3.283352
##    6  5.107019  0.7124899  3.384703
##    7  5.087360  0.7168344  3.377463
##    8  5.085766  0.7170333  3.368396
##    9  5.070250  0.7191337  3.362426
##   10  5.124408  0.7138685  3.436922
##   11  5.144185  0.7133415  3.463343
##   12  5.171378  0.7132273  3.493157
##   13  5.175010  0.7151697  3.491038
##   14  5.221505  0.7131643  3.507437
##   15  5.262921  0.7112154  3.539155
##   16  5.310254  0.7084434  3.578041
##   17  5.356081  0.7042580  3.605718
##   18  5.389684  0.7017908  3.621221
##   19  5.411407  0.7018390  3.629325
##   20  5.433535  0.7008726  3.625294
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 2.
```

```
knnpred<-predict(knnmedv,newdata =trainbost)
RMSE(knnpred,trainbost$medv)
```

```
## [1] 2.589915
```

GLM:

Generalised Linear models are used for classification.

```
admit<-read.csv("/home/himank/Documents/workimages/admit.csv")
part<-createDataPartition(y=admit$Admit,p=0.7,list=F)
trainadmit<-admit[part,]
testadmit<-admit[-part,]
logmodel<-glm(Admit~.,data=trainadmit,family=binomial(link='logit'))
nullModel<-glm(Admit~1,data=trainadmit,family=binomial(link='logit'))
#overdispersion
#if overdispersion >1 its overfitting
# deviance is same as variance for classification models
# var(observed) > var(predicted)
logmodel$deviance/logmodel$df.residual
```

```
## [1] 0.6229526
```

```
logmodel$deviance
```

```
## [1] 169.4431
```

```
aov(logmodel)
```

```
## Call:
##    aov(formula = logmodel)
##
## Terms:
##                       GRE     TOEFL Univ_Rating       SOP       LOR      CGPA
## Sum of Squares  29.245150  1.257283    1.807322  0.735092  1.377080  1.012596
## Deg. of Freedom         1         1           1         1         1         1
##                  Research Residuals
## Sum of Squares   1.624385 31.212520
## Deg. of Freedom         1       272
##
## Residual standard error: 0.3387505
## Estimated effects may be unbalanced
```

```
predtest<-predict(logmodel,newdata=testadmit,type='response')
pred<-ifelse(predtest>0.7,1,0)
confusionMatrix(factor(unname(pred)),factor(testadmit$Admit))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 44 22
##          1  3 51
##
##                Accuracy : 0.7917
##                  95% CI : (0.708, 0.8604)
##     No Information Rate : 0.6083
##     P-Value [Acc > NIR] : 1.423e-05
##
##                   Kappa : 0.5922
##
##  Mcnemar's Test P-Value : 0.0003182
##
##             Sensitivity : 0.9362
##             Specificity : 0.6986
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.9444
##              Prevalence : 0.3917
##          Detection Rate : 0.3667
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.8174
##
##        'Positive' Class : 0
##
```

```r
logLik(logmodel)
```

```
## 'log Lik.' -84.72155 (df=8)
```

```r
logLik(nullModel)
```

```
## 'log Lik.' -190.6097 (df=1)
```

Various tests:

- comparitng two group var: f.test()
- comparing more than two group var: bartlett.test()
- comparing means of more than two groups: global anova(Anova for regression) where alternate hypothesis is that atleast one of the coef is not zero.
- Partial T-test To get which coef is significant in anova f ratio we use partialt test $\frac{(\beta-0)}{SE(\beta)}$ $SE(\beta) = \sqrt{\frac{mse^2}{ss(x)}}$

Multinomial Logistic Regression:

```r
library(nnet)
part<-createDataPartition(iris$Species,p=0.7,list=F)
trainiris<-iris[part,]
testiris<-iris[-part,]


glmodel<-multinom(Species~.,data=iris,Hess=T,)
```

```
## # weights:  18 (10 variable)
## initial  value 164.791843
## iter  10 value 16.177348
## iter  20 value 7.111438
## iter  30 value 6.182999
## iter  40 value 5.984028
## iter  50 value 5.961278
## iter  60 value 5.954900
## iter  70 value 5.951851
## iter  80 value 5.950343
## iter  90 value 5.949904
## iter 100 value 5.949867
## final  value 5.949867
## stopped after 100 iterations
```

```r
summary(glmodel)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris, Hess = T)
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    18.69037    -5.458424   -8.707401     14.24477   -3.097684
## virginica    -23.83628    -7.923634  -15.370769     23.65978   15.135301
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    34.97116     89.89215    157.0415     60.19170    45.48852
## virginica     35.76649     89.91153    157.1196     60.46753    45.93406
##
## Residual Deviance: 11.89973
## AIC: 31.89973
```

```r
pred<-predict(glmodel,testiris,type='prob')

pred<-factor(max.col(pred),labels=c('setosa','versicolor','virginica'))
confusionMatrix(pred,testiris$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
```

```
##   versicolor      0         14         1
##   virginica       0          1        14
##
## Overall Statistics
##
##                Accuracy : 0.9556
##                  95% CI : (0.8485, 0.9946)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9333
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9333           0.9333
## Specificity                 1.0000            0.9667           0.9667
## Pos Pred Value              1.0000            0.9333           0.9333
## Neg Pred Value              1.0000            0.9667           0.9667
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3111           0.3111
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9500           0.9500
```

```r
# Coefficients are log of the odds. probability is:
exp(-5.4584)/(1+exp(-5.4584))
```

```
## [1] 0.004242293
```

```r
xtabs(~Univ_Rating+Admit,data=admit)
```

```
##            Admit
## Univ_Rating  0  1
##           1 24  2
##           2 80 27
##           3 50 83
##           4  9 65
##           5  2 58
```

```r
xtabs(~Admit,data=admit)
```

```
## Admit
##   0   1
## 165 235
```

Therefore imbalanced dataset. We need to oversample or undersample.

```r
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```r
dim(admit)
```

```
## [1] 400   8
```

```r
oversample=ovun.sample(formula =Admit~.,data=admit, method='over',N=235*2)$data
dim(oversample)
```

```
## [1] 470   8
```

```r
sum(oversample[oversample$Admit==1,8])
```

```
## [1] 235
```

```r
undersample=ovun.sample(Admit~.,data=admit,method='under',N=165*2)$data
dim(undersample)
```

```
## [1] 330   8
```

```r
sum(undersample[undersample$Admit==1,8])
```

```
## [1] 165
```

Feature Selection:

Backword and Forward Feature selection is computationally expensive as all combinations of features are need to be modelled. A better approach is using boruta which intrisically uses random forests, bootstrap aggregating and feature subsampling for feature selection.

```r
library(Boruta)
```

```
## Loading required package: ranger
```

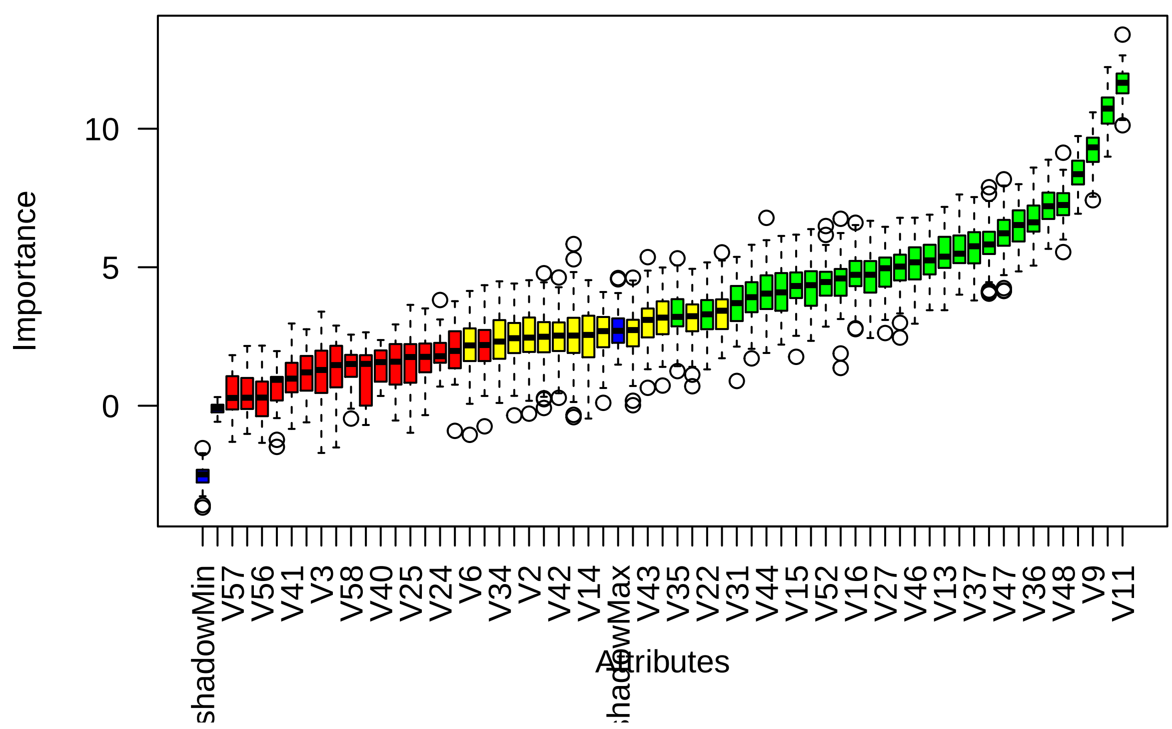
```r
library(mlbench)
data(Sonar)
boruta=Boruta(Class~.,data=Sonar,doTrace=T,maxRuns=100)
```

```
## After 13 iterations, +3.5 secs:

##  confirmed 18 attributes: V10, V11, V12, V13, V20 and 13 more;

##  rejected 3 attributes: V40, V57, V60;

##  still have 39 attributes left.

## After 17 iterations, +4.6 secs:
```

```
##  confirmed 4 attributes: V17, V23, V27, V52;
##  rejected 2 attributes: V53, V58;
##  still have 33 attributes left.
## After 21 iterations, +5.6 secs:
##  rejected 4 attributes: V24, V41, V50, V56;
##  still have 29 attributes left.
## After 24 iterations, +6.3 secs:
##  confirmed 2 attributes: V15, V16;
##  rejected 3 attributes: V3, V33, V55;
##  still have 24 attributes left.
## After 27 iterations, +7 secs:
##  confirmed 2 attributes: V18, V44;
##  still have 22 attributes left.
## After 33 iterations, +8.4 secs:
##  rejected 1 attribute: V25;
##  still have 21 attributes left.
## After 36 iterations, +9 secs:
##  rejected 2 attributes: V38, V7;
##  still have 19 attributes left.
## After 42 iterations, +10 secs:
##  rejected 1 attribute: V59;
##  still have 18 attributes left.
## After 58 iterations, +14 secs:
##  confirmed 2 attributes: V31, V35;
```

```
##  still have 16 attributes left.

## After 64 iterations, +15 secs:

##  confirmed 1 attribute: V22;

##  still have 15 attributes left.

## After 71 iterations, +17 secs:

##  rejected 1 attribute: V29;

##  still have 14 attributes left.
```

```r
plot(boruta,las=2)
```

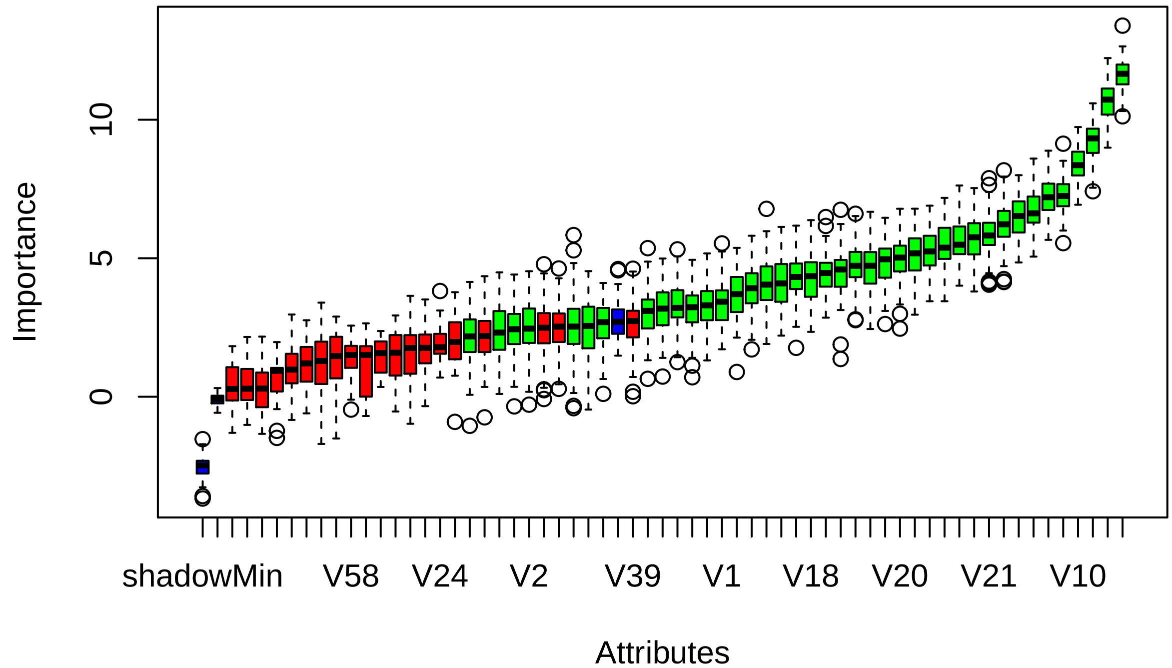



```r
boruta
```

```
## Boruta performed 99 iterations in 23.00021 secs.
##  29 attributes confirmed important: V10, V11, V12, V13, V15 and 24
## more;
##  17 attributes confirmed unimportant: V24, V25, V29, V3, V33 and 12
## more;
##  14 tentative attributes left: V1, V14, V19, V2, V26 and 9 more;
```

Assigning all all tentative variables:

```r
final=TentativeRoughFix(boruta)
plot(final)
```




```r
attStats(final)
```

```
##        meanImp  medianImp      minImp    maxImp   normHits  decision
## V1   3.3897474  3.4300192  1.71491829  5.536224 0.66666667 Confirmed
## V2   2.4681457  2.4598911 -0.28545732  4.531760 0.35353535 Confirmed
## V3   1.1295226  1.2907939 -1.70328692  3.397501 0.03030303  Rejected
## V4   5.2274677  5.2477024  3.44790697  6.897777 1.00000000 Confirmed
## V5   3.8786025  3.9183417  1.70898592  5.811680 0.86868687 Confirmed
## V6   2.1502001  2.1789384 -1.04874241  4.144328 0.39393939 Confirmed
## V7   1.4668597  1.5896007 -0.53284597  2.935932 0.07070707  Rejected
## V8   2.4097357  2.4377063 -0.34683381  4.413219 0.45454545 Confirmed
## V9   9.2157970  9.3270586  7.41771186 10.592535 1.00000000 Confirmed
## V10  8.3761990  8.3598197  6.93113424  9.735245 1.00000000 Confirmed
## V11 11.6129936 11.6579686 10.12419579 13.396534 1.00000000 Confirmed
## V12 10.6560475 10.7261452  8.99137950 12.223990 1.00000000 Confirmed
## V13  5.5037685  5.3817539  3.44941894  7.178768 0.98989899 Confirmed
## V14  2.4369697  2.5565540 -0.46283124  4.533811 0.41414141 Confirmed
## V15  4.2981925  4.3222163  1.76608159  6.176748 0.90909091 Confirmed
## V16  4.7871100  4.7268761  2.76627300  6.605588 0.94949495 Confirmed
## V17  4.6436898  4.7287844  2.44422450  6.676017 0.94949495 Confirmed
```

```
## V18  4.2778793  4.3528421  2.34185777  6.375380 0.87878788 Confirmed
## V19  3.1416611  3.2315937  0.70402635  4.942513 0.67676768 Confirmed
## V20  4.9889396  5.0241912  2.45930817  6.785926 0.94949495 Confirmed
## V21  5.8560718  5.8233506  4.04592867  7.888842 1.00000000 Confirmed
## V22  3.2887706  3.3004008  1.31330782  5.175281 0.70707071 Confirmed
## V23  4.0976701  4.0927132  2.20657461  6.130360 0.92929293 Confirmed
## V24  1.9240965  1.7881693  0.69214310  3.816895 0.02020202  Rejected
## V25  1.5895295  1.7587452 -0.97575729  3.642236 0.06060606  Rejected
## V26  3.1853370  3.1817797  0.72895359  4.991803 0.66666667 Confirmed
## V27  4.8826581  4.9632488  2.62399195  6.459697 0.94949495 Confirmed
## V28  5.6163329  5.4879368  4.00993708  7.627380 0.98989899 Confirmed
## V29  2.2120453  2.1925620 -0.74314071  4.352602 0.20202020  Rejected
## V30  2.5054971  2.5337696 -0.41248635  5.836346 0.45454545 Confirmed
## V31  3.6832341  3.7041991  0.89431142  5.373201 0.76767677 Confirmed
## V32  2.6064866  2.6936219  0.10536987  4.105683 0.49494949 Confirmed
## V33  1.3575505  1.4680790 -1.50616830  2.893588 0.03030303  Rejected
## V34  2.3987231  2.3169130  0.10132102  4.492243 0.40404040 Confirmed
## V35  3.2895248  3.2077540  1.25077766  5.321273 0.70707071 Confirmed
## V36  6.7496730  6.6209850  5.05861047  8.598274 1.00000000 Confirmed
## V37  5.7222483  5.7565327  3.80085418  7.532191 1.00000000 Confirmed
## V38  1.7063254  1.7645310 -0.33806952  3.514219 0.07070707  Rejected
## V39  2.5861114  2.7320489  0.01823627  4.622500 0.45454545  Rejected
## V40  1.4630609  1.5747657  0.35352097  2.371830 0.00000000  Rejected
## V41  0.9855693  0.9796971 -0.83606097  2.969479 0.02020202  Rejected
## V42  2.4964465  2.5323260  0.28485777  4.630261 0.47474747  Rejected
## V43  3.0057124  3.0989714  0.64724289  5.364536 0.60606061 Confirmed
## V44  4.0888445  4.0483770  1.90502266  6.782713 0.86868687 Confirmed
## V45  6.5030728  6.5237187  4.84917567  7.998226 1.00000000 Confirmed
## V46  5.1335751  5.1782750  2.96209536  6.787338 1.00000000 Confirmed
## V47  6.2181637  6.2254896  4.14112702  8.174238 1.00000000 Confirmed
## V48  7.2823656  7.2468306  5.54538698  9.134013 1.00000000 Confirmed
## V49  7.2307864  7.2027670  5.66373538  8.881891 1.00000000 Confirmed
## V50  1.1092690  1.5110791 -0.69454545  2.651475 0.02020202  Rejected
## V51  4.4889408  4.5960510  1.36358180  6.749526 0.97979798 Confirmed
## V52  4.4089702  4.4656287  2.85492033  6.485866 0.95959596 Confirmed
## V53  0.4235676  0.2903019 -1.01649525  2.158747 0.01010101  Rejected
## V54  2.4579419  2.4903433 -0.08184592  4.783397 0.39393939  Rejected
## V55  1.1737714  1.2049106 -0.59862670  2.761869 0.03030303  Rejected
## V56  0.3391607  0.2949186 -1.33924488  2.170904 0.02020202  Rejected
## V57  0.3521015  0.2819449 -1.30331617  1.826433 0.00000000  Rejected
## V58  1.3425942  1.5069706 -0.46380110  2.567998 0.01010101  Rejected
## V59  2.0375340  1.9788384 -0.90634637  3.774814 0.09090909  Rejected
## V60  0.5331242  0.9362159 -1.48510214  1.972512 0.00000000  Rejected
```

```r
getSelectedAttributes(final)
```

```
##  [1] "V1"  "V2"  "V4"  "V5"  "V6"  "V8"  "V9"  "V10" "V11" "V12" "V13" "V14"
## [13] "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V26" "V27" "V28"
## [25] "V30" "V31" "V32" "V34" "V35" "V36" "V37" "V43" "V44" "V45" "V46" "V47"
## [37] "V48" "V49" "V51" "V52"
```

```r
getConfirmedFormula(final)
```

```
## Class ~ V1 + V2 + V4 + V5 + V6 + V8 + V9 + V10 + V11 + V12 +
##     V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 +
##     V23 + V26 + V27 + V28 + V30 + V31 + V32 + V34 + V35 + V36 +
##     V37 + V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52
## <environment: 0x55e01ad56f68>
```

```r
getConfirmedFormula(boruta)
```

```
## Class ~ V4 + V5 + V9 + V10 + V11 + V12 + V13 + V15 + V16 + V17 +
##     V18 + V20 + V21 + V22 + V23 + V27 + V28 + V31 + V35 + V36 +
##     V37 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52
## <environment: 0x55e01cc70ec8>
```

```r
#getting even tentative variables in the forula
getNonRejectedFormula(boruta)
```

```
## Class ~ V1 + V2 + V4 + V5 + V6 + V8 + V9 + V10 + V11 + V12 +
##     V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 +
##     V23 + V26 + V27 + V28 + V30 + V31 + V32 + V34 + V35 + V36 +
##     V37 + V39 + V42 + V43 + V44 + V45 + V46 + V47 + V48 + V49 +
##     V51 + V52 + V54
## <environment: 0x55e01f5cb570>
```