Regression Trees: 5/10/2018

Eric Rupert

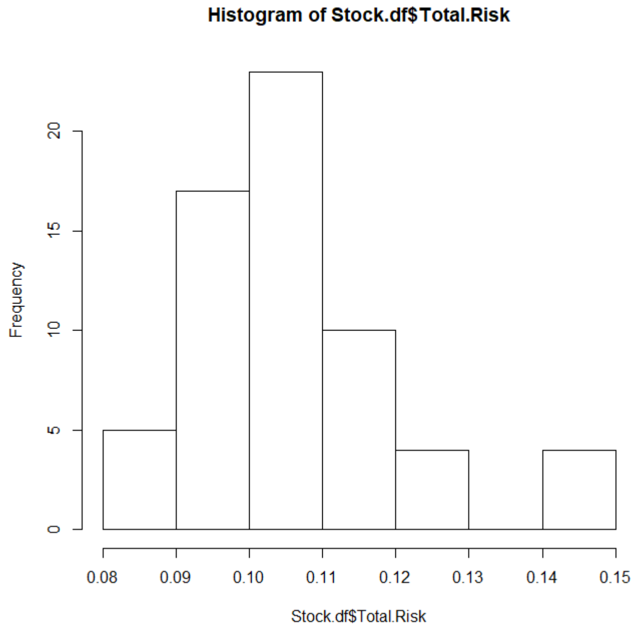
May 10, 2018

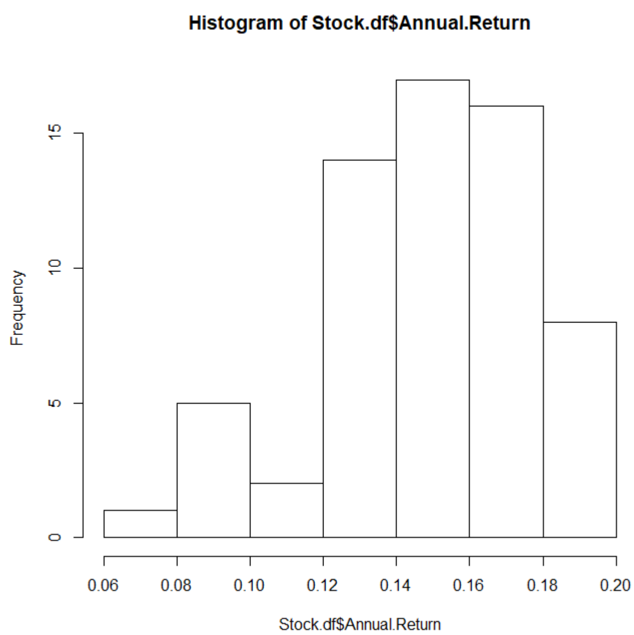
Exploratory data analysis:
Using R's summary function on the stocks data frame from the UCI Repository, there are 6 independent variables and 2 dependent variables (of interest). Each of the independent variables is an investment strategy:

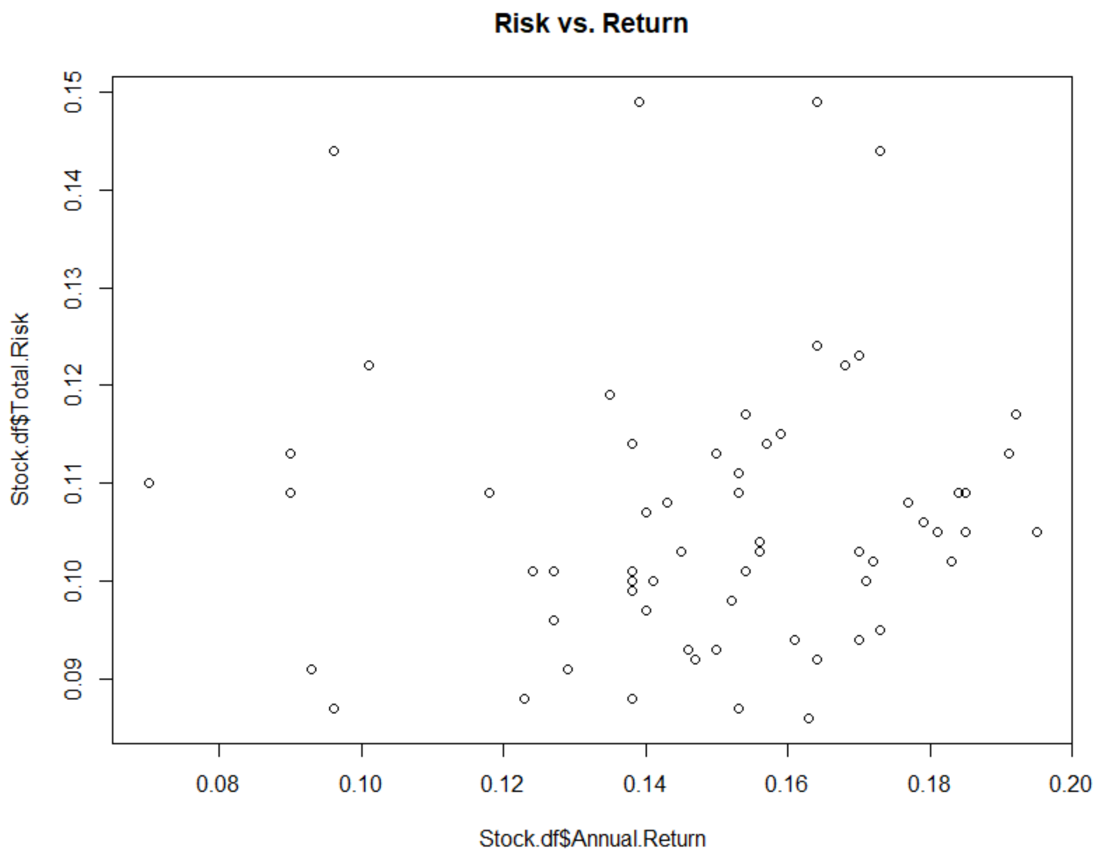
```
> summary(Stock.df)
   Large.B.P        Large.ROE        Large.S.P      Large.Return.Rate
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.1670   Median :0.1670   Median :0.1670   Median :0.1670
 Mean   :0.1666   Mean   :0.1666   Mean   :0.1666   Mean   :0.1666
 3rd Qu.:0.2915   3rd Qu.:0.2915   3rd Qu.:0.2915   3rd Qu.:0.2915
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
 Large.Market.Value Small.sys.Risk   Annual.Return      Total.Risk
 Min.   :0.0000     Min.   :0.0000   Min.   :0.0700   Min.   :0.0860
 1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.1380   1st Qu.:0.0965
 Median :0.1670     Median :0.1670   Median :0.1530   Median :0.1040
 Mean   :0.1666     Mean   :0.1666   Mean   :0.1492   Mean   :0.1063
 3rd Qu.:0.2915     3rd Qu.:0.2915   3rd Qu.:0.1700   3rd Qu.:0.1130
 Max.   :1.0000     Max.   :1.0000   Max.   :0.1950   Max.   :0.1490
```

By looking at mixes of strategies, we will seek to minimize risk and maximize annual return. These are being used as proxies for variance and (time variant) mean respectively. Looking at the risk as variance does remove the inheriently negative connotation of risk (high variance can be good: sell after a high variance upswing, buy after a high variance downswing).

The total risk and annual return are distributed as:







Assuming high return and low risk are desirable (the data repository made no mention of anonymising or otherwise altering data), a first look indicates that most strategy mixes are in the preferred quadrant.

The following code was used in a manual grid search for total risk:

```
>TRmodel<-svm(Total.Risk ~.-Annual.Return-Total.Risk,data=Stock.df,
type="eps-regression",kernel="linear",epsilon=0.01,cost=1,cross=10)
>summary(TRmodel)
```

Where cost, ϵ, γ (where used) and kernel (including order) were varied. The results were judged based on total mean square error The following are examples of the results (total risk):

Kernel	ϵ	Cost	MSE
Linear	0.1	1.0	1.18e-4
	1	1	1.34e-4
	0.01	0.1	1.16e-4

	0.001	0.1	1.10e-4
Poly-2	0.1	1	1.13e-4
	1	1	1.33e-4
	0.01	1	1.04e-4
	0.001	1	1.25e-4
	0.01	10	3.54e-4
	0.01	0.1	1.67e-4
Poly-3	0.1	1	4.04e-4
radial	0.1	1	7.5e-5
$\gamma = 1/6$	1	1	1.58e-4
	0.01	1	7.45e-5
	0.001	1	7.48e-5
	0.01	10	6.89e-5
	0.01	100	7.92e-5
$\gamma = 10/6$	0.01	10	1.83e-4
$\gamma = 1/60$	0.01	10	3.03e-5
$\gamma = 1/600$	0.01	10	1.13e-5
$\gamma = 1/100$	0.01	10	5.68e-5
$\gamma = 1/20$	0.01	10	2.50e-5
$\gamma = 1/20$	0.001	10	2.02e-5
$\gamma = 1/20$	0.0001	10	2.13e-5

The best results are with a radial distribution:
$\gamma = 1/20$, ϵ=0.001 and 1e-4, cost=10.
A similar grid search was performed for annual return, and is not included to save space. The best results for annual return are also radial fits:
$\gamma = 1/20$ and 1/60, ϵ=0.01, cost=10.

Mean Square Error:
Total mean square error from the svm was used to determine the best model. For bootstrapping, a mean square error function was defined as:

```
rmse<- function(error)
{ sqrt(mean(error^2)) }
```

This data has not been declared to be anonymised, and so, presumably, high annual return and low total risk are best. Unfortunately, with a regression, we don't have a convenient analogue to false positive/false negative, as we do with classification.

Bootstrapping for Confidence Intervals:
The following code was used for total risk, with each of the two best models represented once:

```
>err<-integer(200)
>for (i in 1:200) {
>Btrain.df<-Stock.df[sample.int(63, size=50, replace=TRUE),]
>Btest.df<-Stock.df[sample.int(63, size=13, replace=TRUE),]
```

```
>svm.Bmodel<-svm(Total.Risk ~.-Annual.Return-Total.Risk, data=Btrain.df,
type="eps-regression", kernel="radial", epsilon=0.0001, cost=10, gamma=1/20,
cross=10)
>pred<-predict(svm.Bmodel, Btest.df)
>cm<-table(Btest.df$Total.Risk.,pred)
>err[i]=rmse(Btest.df$Total.Risk-pred)}
>serr<-sort(err)
>ub<-serr[195]
>lb<-serr[5]
```

With the bounds found to be (total risk):

Kernel	ub	lb
Radial $\epsilon = 0.001$	0.0102	0.0012
Radial $\epsilon = 0.0001$	0.0089	0.0012

Bounds (annual return):

Kernel	ub	lb
Radial $\gamma = 1/20$	0.0145	0.0022
Radial $\gamma = 1/60$	0.0145	0.0035

Here it can be seen that the Confidence Intervals almost entirely overlap in both cases. Because the Kernels are the same, no computational advantage can be gained from choosing one over the other. In the risk case, I would use $\epsilon = 0.0001$ because the upper bound is lower. For return, I would use $\gamma = 1/20$ because the lower bound is lower. In either case the choice is trivial, and may change if the bootstrapping were run with more than 200 iterations.