Sprint 1

Straddle screening tool

Bassam Rizk - YU id 303525

Research Question

What is the highest reachable consistency in screening stocks that will exhibit abnormal price volatility at a foreseeable market or stock event (dividends announcement…)?

Candidate stocks should have their at-the-money options tree premiums exhibiting high correlation (positive for calls and/or negative for puts) with their underlying stocks' prices.

Data sources

- S&P 100 list of symbols - Wikipedia
- S&P 100 Stock historical (12 years) – API from Quantmod; QuantTools
- S&P 100 stock historical events – API from Quantmod
- Option Tree historical data (call on select stocks in sprint 2) – Quantmod; QuantTools
- Greeks historical data (call on select stocks in sprint 2) – API from fOptions

Update summary

To make sure I can meet all deadlines – I have taken the path of leveraging Rattle package in R Studio for clustering, modeling…

Rattle is a popup interface that calls upon 100s of other packages in the background. It offers all phases of modeling from exploration/transformation/clustering/modeling/testing/validating.

For my data to be ready for raddle and to avoid transformation within Rattle, I went back and added a key field – HLPPC – High Low Percentage Change - that measures the percentage price gap within each day's high and low price points.

Below is a bird eye view of what was done in Sprint 2:

- ✓ Pre-raddle transformation – adding HLPPC
- ✓ Principal component review of the aggregated data set S&P100Full
- ✓ Splitting the data into training, validation & testing.
- ✓ Clustering the data set S&P100Full (K means, EWKM…)
- ✓ Modeling the data for HLPPC (using decision tree, neural network & linear)
- ✓ Validating & testing all 3 models.

Contents

Research Question 1
Data sources 1
Update summary 1
R Studio Libraries 3
Pre-Rattle Transformation 3
Challenges 3
Solutions 3
Explore % change by stock vs stock volatility (outside Rattle) 4
Rattle 5
Re-Explore data within Rattle 5
Explore Correlation Between Key Variables 8
Principal component 9
Clustering 9
K means 9
EWKM 11
Modeling 13
Testing & Training 13
Decision trees 13
Neural Networks 16
Linear 17

R Studio Libraries

```
library(quandl)

library(QuantTools)

library(quantmod)

library(derivmkts)

library(RND)

setDefaults(getSymbols.av, api.key="V7YC53BOMBUB28FJ")

library(rattle)
```

Pre-Rattle Transformation

Comment: I noticed that I will need percent price change as a variable in any of my clustering & models - adding that to the merged full SnP100 file might be tricky

I added % closing price change column to the consolidated data frame

```
SnP100full$PerChange <- c(-diff(SnP100full$Close)/SnP100full$Close[-1]*100,0)
```

Challenges

Did a dry run on a test sample and have 2 challenges in running Clustering functions:

1. there are still NAs in relatively newer stocks (e.g. Netflix...) that were not existent throughout the 2007-2019 sample period.
2. Preset library functions for measuring percentage price change (Overnight & intra-day) do not seem to be compatible with various clustering functions.

Solutions

Go back to the individual stock data frames and do couple of clean-ups:

1. Transform all remaining NAs in individual data sets into nill value
2. add overnight and same day percentage price change
3. remerge the individual 100 stock data frames into a new large data frame

Transform all remaining NAs in individual data sets into nil value

```
AAPLfull[is.na(AAPLfull)] <-0

ABBVfull[is.na(ABBVfull)] <-0
```

… replicate for all stocks

Add overnight and same day percentage price change

 # start with overnight (difference between close of the day and the previous day)

```
AAPLfull$ON_PPC<- c(-diff(AAPLfull$Close)/AAPLfull$Close[-1]*100,0)
```

ABBVfull$ON_PPC<- c(-diff(ABBVfull$Close)/ABBVfull$Close[-1]*100,0)

… replicate for all 100 stocks

Measure percent change between a day's high & low

AAPLfull$HLppc<- c((AAPLfull$High - AAPLfull$Low)/AAPLfull$Low*100)

ABBVfull$HLppc<- c((ABBVfull$High - ABBVfull$Low)/ABBVfull$Low*100)

… replicate for all 100 stocks

Removing NAs

There seems to be few NAs in few stocks - this applies to period where a stock was not yet listed

Clean-up - another round of cleaning NAs in the percentage change fields

AAPLfull[is.na(AAPLfull)] <-0

ABBVfull[is.na(ABBVfull)] <-0

… replicate for all 100 stocks

Re-merging all 100 files after adding overnight and high/close percentage change

SnP100full<- rbind(AAPLfull, ABBVfull, ABTfull, ACNfull, ADBEfull, AGNfull, AIGfull, ALLfull, AMGNfull, AMZNfull, AXPfull, BAfull, BACfull, BIIBfull, BKfull, BKNGfull, BLKfull, BMYfull, Cfull, CATfull, CELGfull, CHTRfull, CLfull, CMCSAfull, COFfull, COPfull, COSTfull, CSCOfull, CVSfull, CVXfull, DDfull, DHRfull, DISfull, DOWfull, DUKfull, EMRfull, EXCfull, Ffull, FBfull, FDXfull, GDfull, GEfull, GILDfull, GMfull, GOOGfull, GOOGLfull, GSfull, HDfull, HONfull, IBMfull, INTCfull, JNJfull, JPMfull, KHCfull, KMIfull, KOfull, LLYfull, LMTfull, LOWfull, MAfull, MCDfull, MDLZfull, MDTfull, METfull, MMMfull, MOfull, MRKfull, MSfull, MSFTfull, NEEfull, NFLXfull, NKEfull, NVDAfull, ORCLfull, OXYfull, PEPfull, PFEfull, PGfull, PMfull, PYPLfull, QCOMfull, RTNfull, SBUXfull, SLBfull, SOfull, SPGfull, Tfull, TGTfull, TXNfull, UNHfull, UNPfull, UPSfull, USBfull, UTXfull, Vfull, VZfull, WBAfull, WFCfull, WMTfull, XOMfull)

Explore % change by stock vs stock volatility (outside Rattle)

<u>Comment</u>: Obviously new stocks like Netflix and stocks that witnessed turmoil during the 2008 global economic crisis (e.g. AIG) top the list when sorted by average daily percentage price change.

I wanted to explore a summarized table by stock of key metrics

```
SnPSummary <- group_by(SnP100full, SnP100full$Symbol)

SnPSummary = summarise(SnPSummary,

        avg_vlty = mean(Volatility),

        min_vlty = min(Volatility),

        max_vlty = max(Volatility),

        avg_ON_ppc = mean(ON_PPC),

        min_On_ppc = min(ON_PPC),
```

max_on_ppc = max(ON_PPC),

avg_HL_ppc = mean(HLppc),

min_HL_ppc = min(HLppc),

max_HL_ppc = max(HLppc))

SnPSummary

Filter

	SnP100full$Symbol	avg_vlty	min_vlty	max_vlty	avg_ON_ppc	min_On_ppc	max_on_ppc	avg_HL_ppc	min_HL_ppc	max_HL_ppc
100	NFLX	0.4582624	0	2.5101368	-0.08265853809	-29.688136	53.599581	3.808675	0.61086947	29.595746
99	AIG	0.4156295	0	7.0795822	0.21284567693	-39.759036	155.042028	3.639627	0.31728341	309.600008
98	NVDA	0.4155334	0	1.9600368	-0.01218033521	-22.962378	44.355492	3.550914	0.66979236	33.146067
97	MS	0.3868942	0	4.7694282	0.07635247392	-46.519337	34.939751	3.405129	0.48515410	111.282043
96	C	0.3832878	0	4.2090489	0.13016364372	-36.638654	64.000000	3.248808	0.42153862	81.311472
95	F	0.3324752	0	2.5995613	0.02972956565	-22.790698	33.333333	3.133792	0.50000000	160.476190
94	BAC	0.3793326	0	3.1143036	0.07719676457	-26.073298	40.784314	3.111502	0.39318480	61.660079
93	COF	0.3545884	0	2.4113365	0.04054039161	-20.904921	33.408072	3.105245	0.32330516	33.410138
92	MET	0.3391642	0	2.7983662	0.04699267873	-21.874999	36.555554	2.852607	0.43948903	31.444098

Rattle

INSTALL.PACKAGES("RATTLE")

INSTALL.PACKAGES("RATTLE", DEPENDENCIES=C("DEPENDS", "SUGGESTS"))

LIBRARY(RATTLE)

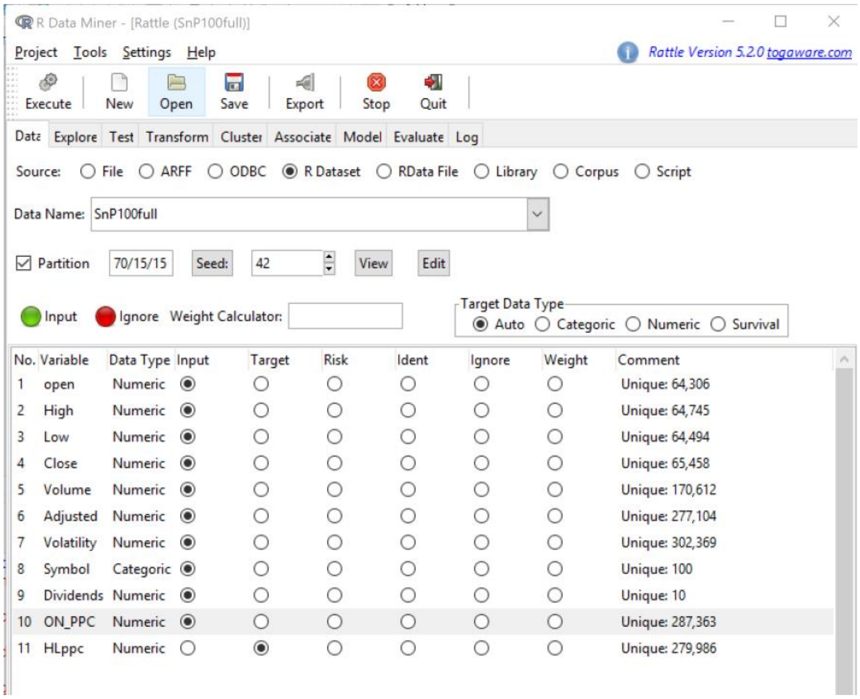
RATTLE()

https://cran.r-project.org/src/contrib/Archive/RGtk2/RGtk2_2.20.35.tar.gz

Select in case of a difficulty to open .rattle file in RGtk2 – please use the above link to download the earlier 2.20.35 version of RGtk2 library – that should solve the issue

Re-Explore data within Rattle

Select file SnP100full with High Low percentage change as a target variable.



==

Below is a description of the dataset.

The data is limited to the training dataset.

crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)]

11 Variables 212352 Observations

The data was few pages and couldn't clearly move it from text to clean tables – I summarized in the table below

Rattle timestamp: 2019-08-09 09:33:06 bassa

==

Basic statistics for key numeric variable of the dataset.

metric	$Volatility	$ON_PPC	$HLppc	$Dividends
nobs	212352	212352	212352	212352
NAs	0	0	0	0

Minimum	0	-100	0	0
Maximum	7.079582	155.04203	309.6	0.77
1 Quartile	0.140928	-0.845982	1.203861	0
3 Quartile	0.29334	0.761575	2.55102	0
Mean	0.250731	**-0.012775**	**2.207995**	0.005104
Median	0.200953	-0.046004	1.712853	0
Sum	53243.1645	-2712.801	468872.2	1083.78137
SE Mean	0.00044	0.004421	0.004533	0.000116
LCL Mean	0.249868	-0.02144	2.199111	0.004876
UCL Mean	0.251593	-0.00411	2.216879	0.005332
Variance	0.041118	4.150226	4.362836	0.002871
Stdev	0.202775	2.03721	2.08874	0.053586
Skewness	5.600613	1.686744	23.2184	10.91106
Kurtosis	79.753947	287.57577	2484.471	121.978523

==

Kurtosis for each numeric variable of the dataset.

Larger values mean sharper peaks and flatter tails.

Positive values indicate an acute peak around the mean.

Negative values indicate a smaller peak around the mean.

Open	High	Low	Close	Volume	Adjusted	Volatility	ON_PPC	HLppc	Dividends
55.91057	55.84031	55.97522	55.86483	140.44138	57.52392	79.75395	287.57577	2484.47069	55.91057

==

Skewness for each numeric variable of the dataset.

Positive means the right tail is longer.

Open	High	Low	Close	Volume	Adjusted	Volatility	ON_PPC	HLppc	Dividends
6.709407	6.706561	6.712658	6.707722	8.901054	6.784499	5.600613	1.686744	23.218403	10.911060

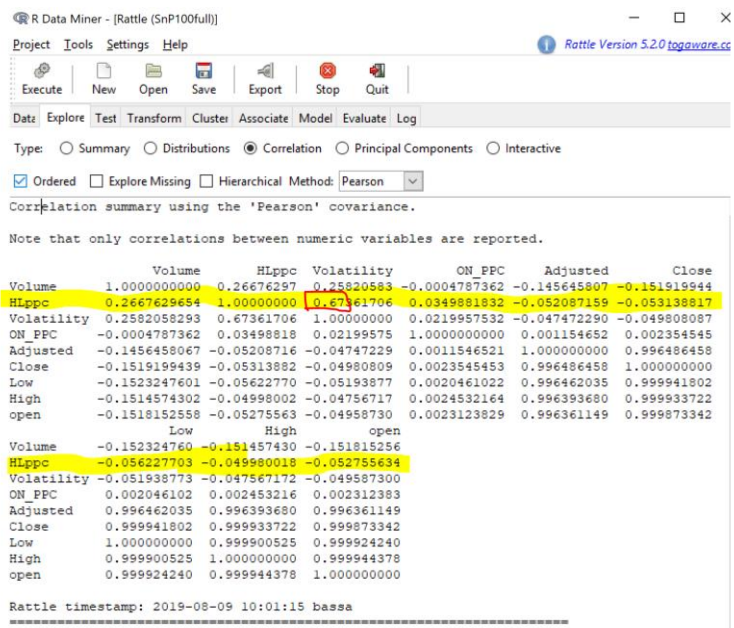
Rattle timestamp: 2019-08-09 09:33:09 bassa

==

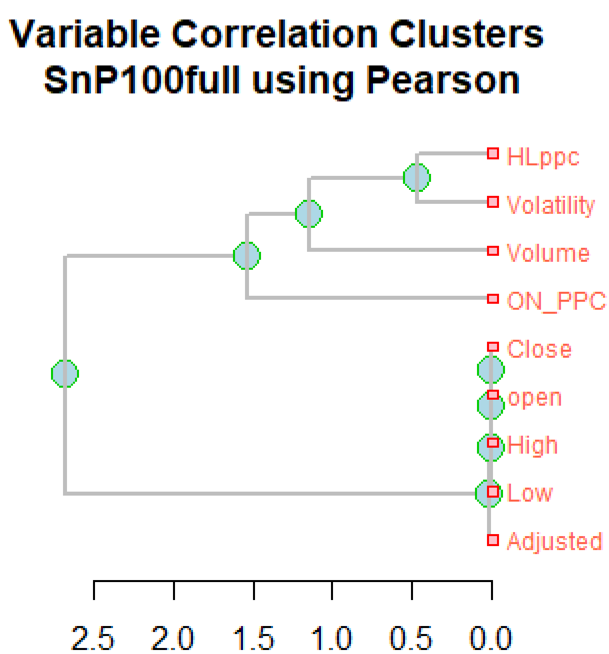
Explore Correlation Between Key Variables

<u>Take away</u>: Its very obvious that HLPPC High low percentage price change per day has a high correlation with stock volatility at 67%

Also a significant variable with high correlation to High/Low ppc is the traded volume at 26%



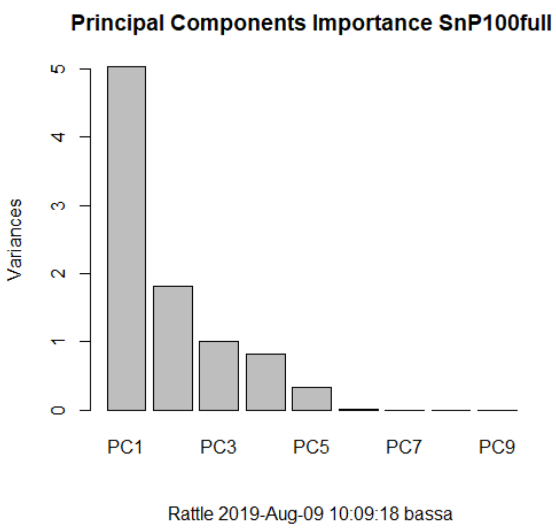

```
Correlation summary using the 'Pearson' covariance.

Note that only correlations between numeric variables are reported.

                 Volume       HLppc  Volatility        ON_PPC     Adjusted        Close
Volume      1.0000000000  0.26676297  0.25820583 -0.0004787362 -0.145645807 -0.151919944
HLppc       0.2667629654  1.00000000  0.67361706  0.0349881832 -0.052087159 -0.053138817
Volatility  0.2582058293  0.67361706  1.00000000  0.0219957532 -0.047472290 -0.049808087
ON_PPC     -0.0004787362  0.03498818  0.02199575  1.0000000000  0.001154652  0.002354545
Adjusted   -0.1456458067 -0.05208716 -0.04747229  0.0011546521  1.000000000  0.996486458
Close      -0.1519199439 -0.05313882 -0.04980809  0.0023545453  0.996486458  1.000000000
Low        -0.1523247601 -0.05622770 -0.05193877  0.0020461022  0.996462035  0.999941802
High       -0.1514574302 -0.04998002 -0.04756717  0.0024532164  0.996393680  0.999933722
open       -0.1518152558 -0.05275563 -0.04958730  0.0023123829  0.996361149  0.999873342
                   Low        High        open
Volume     -0.152324760 -0.151457430 -0.151815256
HLppc      -0.056227703 -0.049980018 -0.052755634
Volatility -0.051938773 -0.047567172 -0.049587300
ON_PPC      0.002046102  0.002453216  0.002312383
Adjusted    0.996462035  0.996393680  0.996361149
Close       0.999941802  0.999933722  0.999873342
Low         1.000000000  0.999900525  0.999924240
High        0.999900525  1.000000000  0.999944378
open        0.999924240  0.999944378  1.000000000

Rattle timestamp: 2019-08-09 10:01:15 bassa
======================================================================
```

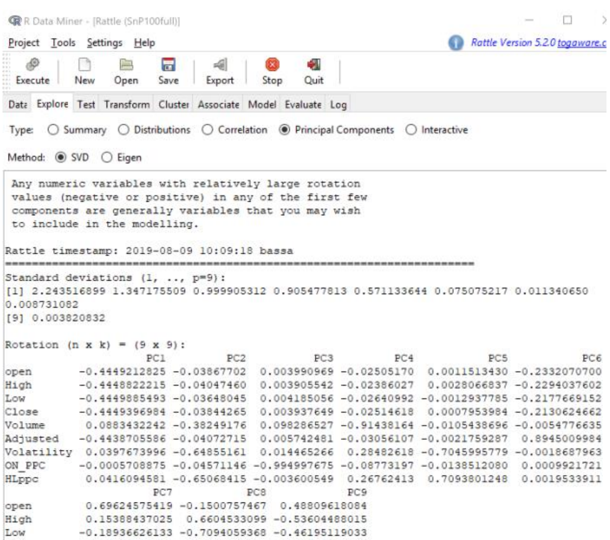


Principal component

***Take-away*: PC 1 and 2 provide the most significant results – interestingly:**

- ***PC1 is driven relatively more by other variables than volatility and HLPPC.***
- ***PC2 is mostly driven by volatility and HLPPC***






```
Any numeric variables with relatively large rotation
values (negative or positive) in any of the first few
components are generally variables that you may wish
to include in the modelling.

Rattle timestamp: 2019-08-09 10:09:18 bassa
======================================================================
Standard deviations (1, .., p=9):
[1] 2.243516899 1.347175509 0.999905312 0.905477813 0.571133644 0.075075217 0.011340650
0.008731082
[9] 0.003820832

Rotation (n x k) = (9 x 9):
                     PC1         PC2          PC3         PC4           PC5           PC6
open       -0.4449212825 -0.03867702  0.003990969 -0.02505170  0.0011513430 -0.2332070700
High       -0.4448822215 -0.04047460  0.003905542 -0.02386027  0.0028066837 -0.2294037602
Low        -0.4449885493 -0.03648045  0.004185056 -0.02640992 -0.0012937785 -0.2177669152
Close      -0.4449396984 -0.03844265  0.003937649 -0.02514618  0.0007953984 -0.2130624662
Volume      0.0883432242 -0.38249176  0.098286527 -0.91438164 -0.0105438696 -0.0054776635
Adjusted   -0.4438705586 -0.04072715  0.005742481 -0.03056107 -0.0021759287  0.8945009984
Volatility  0.0397673996 -0.64855161  0.014465266  0.28482618 -0.7045995779 -0.0018687963
ON_PPC     -0.0005708875 -0.04571146 -0.994997675 -0.08773197 -0.0138512080  0.0009921721
HLppc       0.0416094581 -0.65068415 -0.003600549  0.26762413  0.7093801248  0.0019533911
                     PC7           PC8           PC9
open        0.69624575419 -0.1500757467  0.48809618084
High        0.15388437025  0.6604533099 -0.53604488015
Low        -0.18936626133 -0.7094059368 -0.46195119033
```

Clustering

In clustering we used 2 methods Kmeans and EWKM

K means

***Take-away*: Very obvious from the data means that volatility & ON_PPC (overnight volatility) have the highest "Data means" vs our target variable HLPPC.**

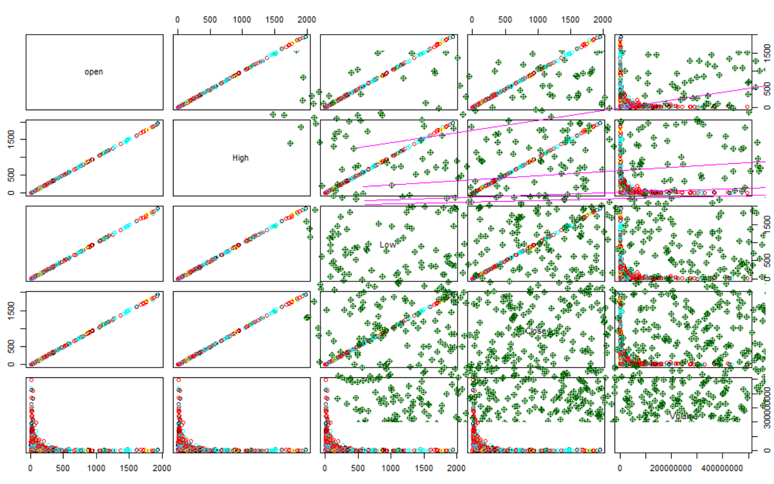
<u>Limitation</u>: It was challenging to map more than 5 variables in a clustering chart – the key variables (volatility & ON_PPC) could not be captured in the chart below – which made the chart plot less useful.


```
Cluster sizes:

[1] "8098 102386 2331 1927 65683 737 6802 2168 1754 20466"

Data means:

       open        High         Low       Close      Volume    Adjusted
0.043698124 0.043764123 0.043987077 0.043792787 0.011390931 0.040135184
 Volatility   Dividends      ON_PPC
0.035416030 0.006628185 0.392042150

Cluster centers:

         open       High        Low      Close      Volume    Adjusted
1  0.01609991 0.01648120 0.01581672 0.01611960 0.023074059 0.013373868
2  0.01710462 0.01713751 0.01721555 0.01714485 0.013391147 0.014307049
3  0.01050156 0.01059775 0.01045047 0.01050034 0.164794708 0.009164876
4  0.04395180 0.04395646 0.04424581 0.04403101 0.008290425 0.041285596
5  0.03860041 0.03861447 0.03891687 0.03869412 0.005413192 0.034162415
6  0.82010688 0.82098539 0.82587030 0.82155784 0.001577265 0.821557842
7  0.14571484 0.14600334 0.14651397 0.14597216 0.002997334 0.143084923
8  0.29287297 0.29315104 0.29483071 0.29342391 0.001742851 0.289259846
9  0.51976416 0.52043452 0.52314435 0.52090077 0.001217580 0.501276113
10 0.07871503 0.07881348 0.07926253 0.07888660 0.003803185 0.074256202
   Volatility    Dividends    ON_PPC
1  0.13244210 0.00000000000 0.3928479
2  0.03305692 0.00005254951 0.3919905
3  0.06702872 0.00000000000 0.3924286
4  0.02973600 0.70943998814 0.3916255
5  0.02753662 0.00009038747 0.3920493
6  0.03317629 0.01348041375 0.3919954
7  0.03594380 0.00005455124 0.3920990
8  0.03398906 0.00053912706 0.3919984
9  0.03373692 0.00962548471 0.3921376
10 0.03124923 0.00003626087 0.3919332

Within cluster sum of squares:
```

Within cluster sum of squares:

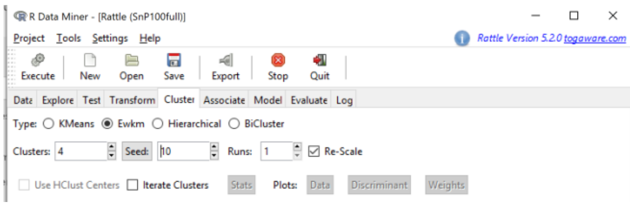
[1] 46.34335 70.02040 23.55418 62.58690 40.09805 29.81370 25.71628

[8] 30.84591 50.33915 29.06806

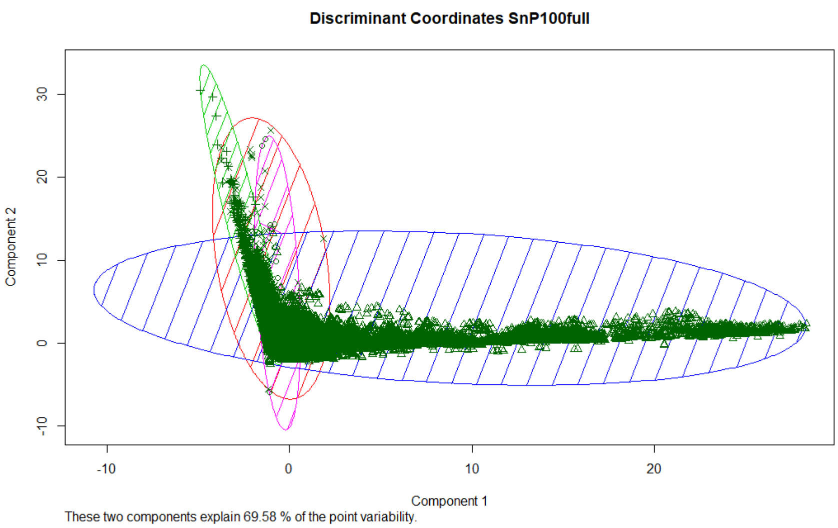


EWKM

<u>Take-away</u>: Because of the limitations in charting Kmeans cluster – I tried using the EWKM - Entropy Weighted K-Means which is more useful for high dimensional data.



Below is summarized version of the findings.



4 clusters, 1 iterations, 0 restarts, 2 total iterations.

Cluster sizes:

[1] "27464 122157 34997 27734"

Data means:

open	High	Low
0.043698124	0.043764123	0.043987077
Close	Volume	Adjusted
0.043792787	0.011390931	0.040135184
Volatility	Dividends	ON_PPC
0.035416030	0.006628185	0.392042150

Cluster centers:

	open	High	Low
1	0.022540517	0.022620328	0.022644125
2	0.064403462	0.064470034	0.064862211
3	0.009875018	0.009922726	0.009907357
4	0.016131883	0.016204758	0.016180303

	Close	Volume	Adjusted
1	0.022589408	0.009935015	0.019211047
2	0.064544478	0.004567639	0.060118188
3	0.009897513	0.031075741	0.008027256
4	0.016158773	0.018046617	0.013354978

	Volatility	Dividends	ON_PPC
1	0.03710539	0.00798676223	0.3924519
2	0.03002261	0.00844149761	0.3919591
3	0.04609568	0.00444670015	0.3917770
4	0.04402251	0.00004870013	0.3923368

Cluster weights:

	open	High	Low	Close	Volume	Adjusted
1	0.18	0.18	0.19	0.19	0.05	0.18
2	0.00	0.00	0.00	0.00	0.77	0.00
3	0.19	0.19	0.19	0.19	0.00	0.21
4	0.18	0.18	0.18	0.18	0.00	0.18

	Volatility	Dividends	ON_PPC
1	0	0.0	0.02
2	0	0.0	0.23
3	0	0.0	0.03
4	0	0.1	0.00

Within cluster sum of squares:

[1] 0 0 0 0

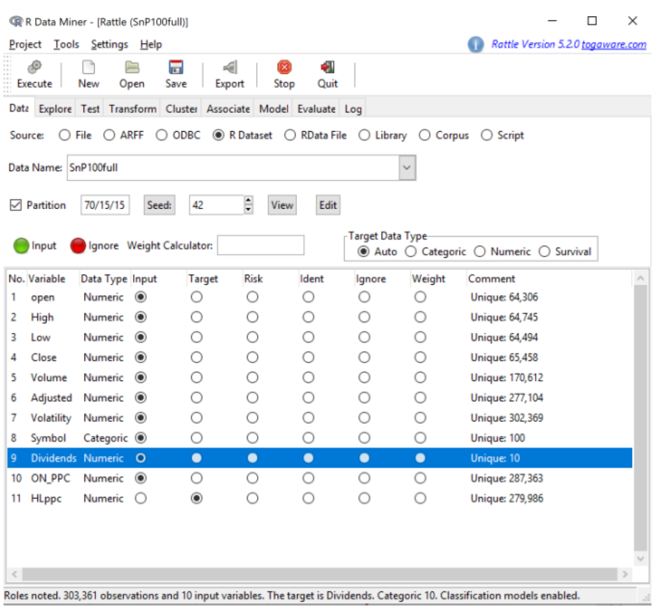
Comments: ONPPC – Overnight percentage price change seem to be the most determinant in the EWKM clustering – on the other hand Volatility seems to be at par with other variables.

Modeling

Testing & Training

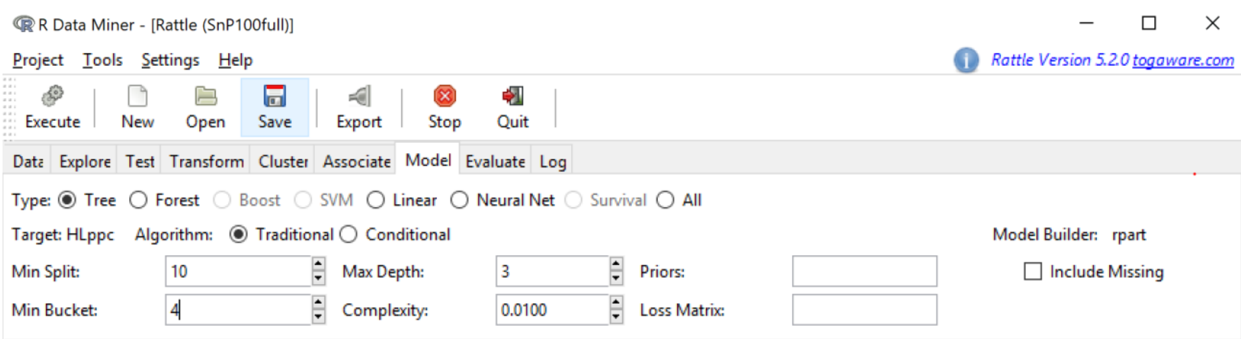
Take-aways:

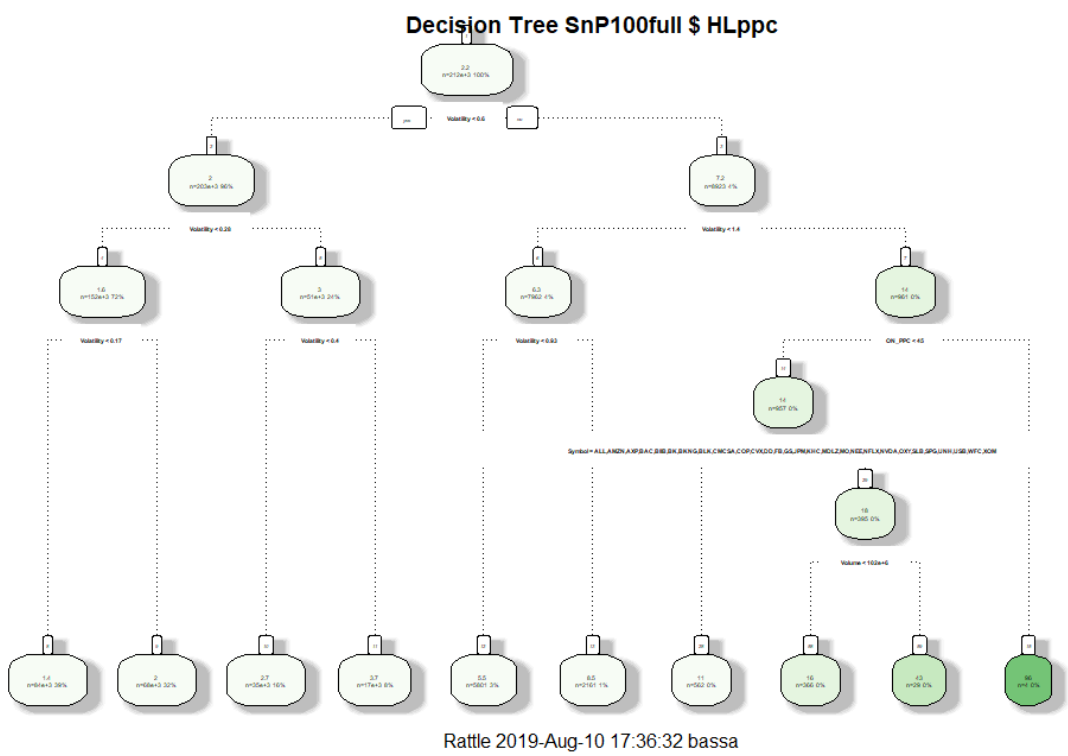
- ***Data was split in a 70% training, 15% validation & 15% testing format.***
- ***We are trying to predict a stock's percentage change to screen the highest for a straddle strategy.***
- ***3 models were compatible with the nature our data frame.***
    1. ***Decision trees***
    2. ***Neural Networks***
    3. ***Linear regression***
- ***Testing and validation came back with very similar results – so only one was reported to save on real estate.***
- ***Also to save on real estate – the bulk of coefficient tables, nodes tables were removed.***
- ***Predict & Observe method was used to compare prediction capability of all 3 models with Pseudo R-Square score used as a benchmark.***



Decision trees

Take-away: Driven by volatility & volume variables – decision tree were able to predict at 0.476 R-square.





Summary of the Decision Tree model for Classification (built using 'rpart'):

n= 212352 node), split, n, deviance, yval

* denotes terminal node

1) root 212352 926452.60 2.207995

2) Volatility< 0.6044209 203429 322549.00 1.990545

4) Volatility< 0.2761405 152209 103667.00 1.637718

8) Volatility< 0.1741081 83749 31857.02 1.366199 *

9) Volatility>=0.1741081 68460 58082.70 1.969875 *

5) Volatility>=0.2761405 51220 143626.70 3.039031

10) Volatility< 0.4033684 34670 66522.50 2.708230 *

11) Volatility>=0.4033684 16550 65362.53 3.732015 *

3) Volatility>=0.6044209 8923 374988.00 7.165469

6) Volatility< 1.378729 7962 120587.00 6.334969

12) Volatility< 0.9283672 5801 54583.42 5.541613 *

13) Volatility>=0.9283672 2161 52551.00 8.464659 *

7) Volatility>=1.378729 961 203410.40 14.046260

14) ON_PPC< 44.62153 957 115484.80 13.704970

28)
Symbol=ALL,AMZN,AXP,BAC,BIIB,BK,BKNG,BLK,CMCSA,COP,CVX,DD,FB,GS,JPM,KHC,MDLZ,MO,NEE,NFLX,NVDA,OXY,SLB,SPG,UNH,USB,WFC,XOM 562 22652.93 10.935450 *

29) Symbol=AIG,C,COF,F,GE,MET,MS 395 82388.08 17.645390

58) Volume< 1.019247e+08 366 32205.36 15.597690 *

59) Volume>=1.019247e+08 29 29279.57 43.488750 *

15) ON_PPC>=44.62153 4 61144.00 95.701280 *

Regression tree:

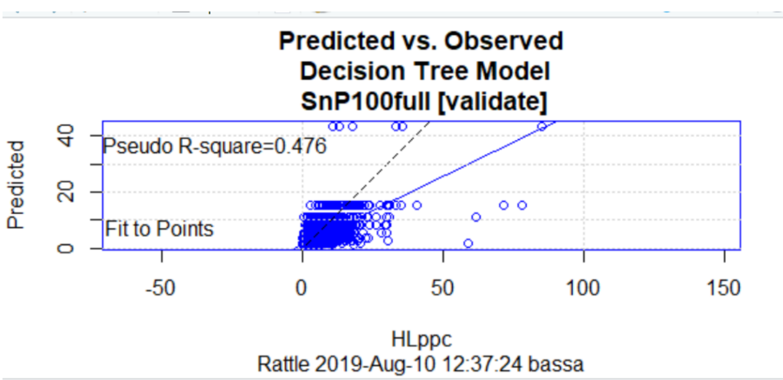
rpart(formula = HLppc ~ ., data = crs$dataset[crs$train, c(crs$input,

crs$target)], method = "anova", model = TRUE, parms = list(split = "information"),

control = rpart.control(minsplit = 10, minbucket = 4, usesurrogate = 0,

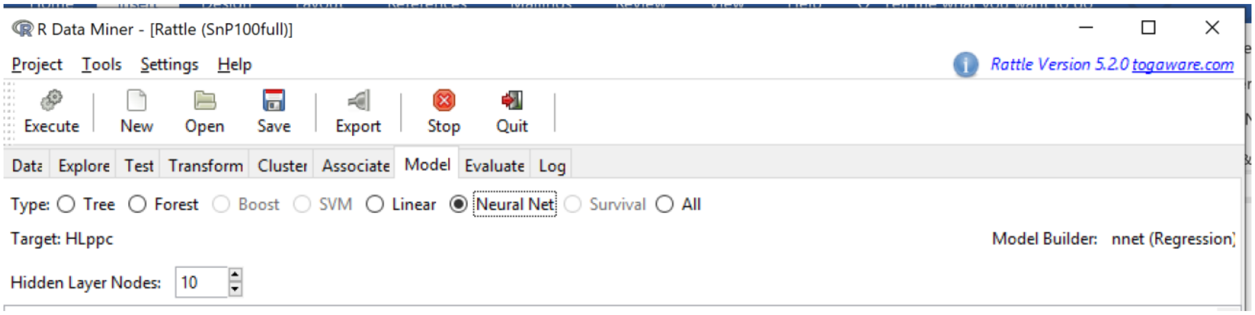
maxsurrogate = 0))

Test – validation test came back with very similar results



Neural Networks

Take-away: Neural Network was able to predict at 0.36 R-square (vs. 0.47 for decision trees)



Summary of the Neural Net model (built using nnet):

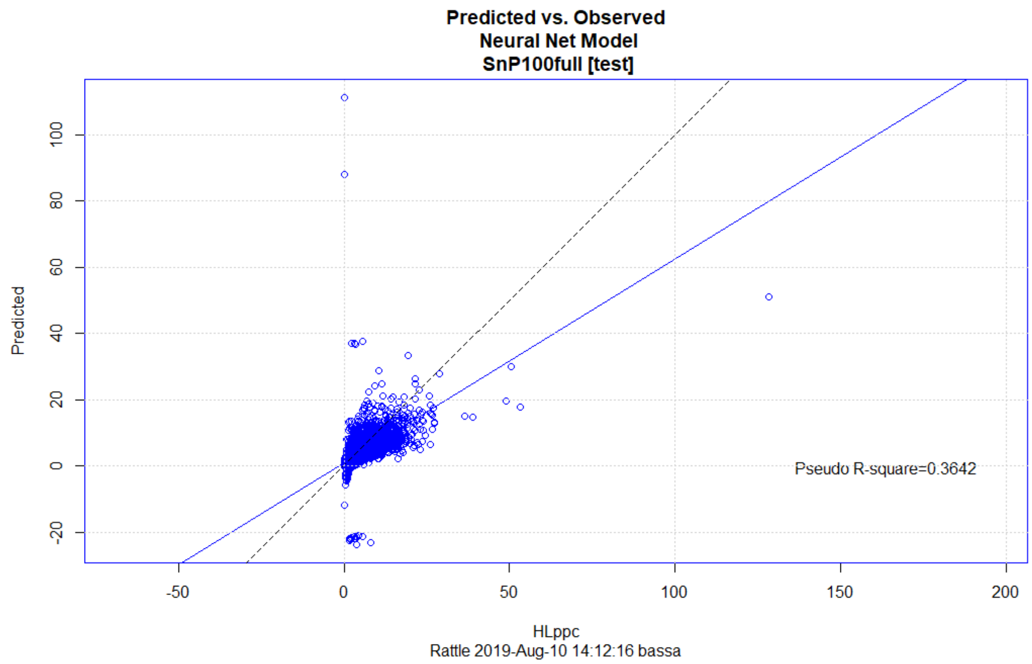
A 108-10-1 network with 1209 weights.

Inputs: open, High, Low, Close, Volume, Adjusted, Volatility, SymbolABBV, SymbolABT, SymbolACN, SymbolADBE, SymbolAGN, SymbolAIG, SymbolALL, SymbolAMGN, SymbolAMZN, SymbolAXP, SymbolBA, SymbolBAC, SymbolBIIB, SymbolBK, SymbolBKNG, SymbolBLK, SymbolBMY, SymbolC, SymbolCAT, SymbolCELG, SymbolCHTR, SymbolCL, SymbolCMCSA, SymbolCOF, SymbolCOP, SymbolCOST, SymbolCSCO, SymbolCVS, SymbolCVX, SymbolDD, SymbolDHR, SymbolDIS, SymbolDOW, SymbolDUK, SymbolEMR, SymbolEXC, SymbolF, SymbolFB, SymbolFDX, SymbolGD, SymbolGE, SymbolGILD, SymbolGM, SymbolGOOG, SymbolGOOGL, SymbolGS, SymbolHD, SymbolHON, SymbolIBM, SymbolINTC, SymbolJNJ, SymbolJPM, SymbolKHC, SymbolKMI, SymbolKO, SymbolLLY, SymbolLMT, SymbolLOW, SymbolMA, SymbolMCD, SymbolMDLZ, SymbolMDT, SymbolMET, SymbolMMM, SymbolMO, SymbolMRK, SymbolMS, SymbolMSFT, SymbolNEE, SymbolNFLX, SymbolNKE, SymbolNVDA, SymbolORCL, SymbolOXY, SymbolPEP, SymbolPFE, SymbolPG, SymbolPM, SymbolPYPL, SymbolQCOM, SymbolRTN, SymbolSBUX, SymbolSLB, SymbolSO, SymbolSPG, SymbolT, SymbolTGT, SymbolTXN, SymbolUNH, SymbolUNP, SymbolUPS, SymbolUSB, SymbolUTX, SymbolV, SymbolVZ, SymbolWBA, SymbolWFC, SymbolWMT, SymbolXOM, Dividends, ON_PPC.

Output: HLppc.

Sum of Squares Residuals: 22887008171185948.0000.

Neural Network build options: skip-layer connections; linear output units.

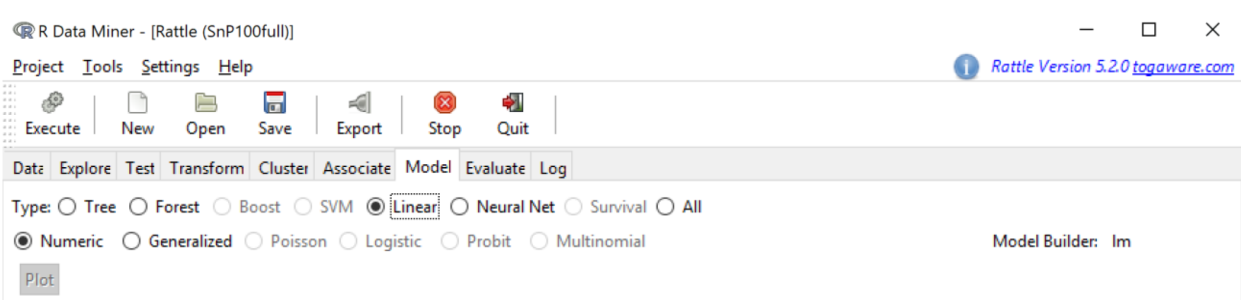


Linear

Take-away: Compared with decision trees and neural network, Linear regression was able to produce the highest R-square of 0.58.

Linear model also produced a clear list of predicted percentage change:

	Estimate	*Std. Error*	*t value*
SymbolABBV	*1.734e+00*	*5.441e-02*	*31.869*
SymbolABT	*1.573e+00*	*4.605e-02*	*34.162*



Summary of the Linear Regression model (built using lm):

Call:

lm(formula = HLppc ~ ., data = crs$dataset[crs$train, c(crs$input,

crs$target)])

Residuals:

Min 1Q Median 3Q Max

-35.298 -0.478 -0.089   0.336 260.503

Test