Predicting SaaS Revenue Multiples

O. Oladunjoy, M. Thomas, M. Dagnok, M.S.M. Htet, R. Alam, and J. Jackson

29 April 2022

Introduction

How to value a company is one of the greatest debates in all of finance. One of the most common ways that investors value companies is using revenue multiples. The appropriate revenue multiple to apply to a subject company is obtained from comparable public companies or precedent transaction multiples. Some industries have higher multiples than others. One industry that enjoys some of the highest revenue multiples is software and in particular SaaS which aslo known as Software as a Service. Some notable SaaS companies include Zoom, Slack, and Dropbox, to name a few. We set out to predict what the revenue multiple of a SaaS company is given several financial metrics. Additionally, we want to find out what gives some SaaS companies higher revenue multiples than others. We are particularity interested in these data questions because all of our group members plan to work as software engineers after graduating. We will use linear regression, a decision tree, and a random forest regression to explore these data questions.

About the data

We define a revenue multiple as the enterprise value of a company divided by the trailing 12 months' revenue of the company. If a company has a 4 billion dollar enterprise value and did 200 million in revenue over the last 12 months, then its multiple would be 20x. All of the companies that we analyzed are publicly traded and we were able to obtain their financial data through lawfully required SEC filings via Ycharts. We put together a dat a set of 90 companies on our own but is based on the SEG SaaS Index 2022 Annual Report. SEG / Software Equity Group is a financial anlayst firm that focuses on analyzing SaaS companies. The index includes about 100 companies but we removed a handful that had incomplete data. Our response variable is ev_ttm_multiple which stands for enterprise value / trailing twelve months multiple. Our first predictor is revenue_growth which is revenue growth % year over year. Our second predictor is ebitda_margin % which is a way to measure the profitability of a company. Our third predictor is sales_efficiency which measures how effective a company is at turning sales and marketing spend into revenue. Our last and fourth predictor is gross_margin which measures the amount of profit made for every dollar in revenue.

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.1.2
```

```
StockData <- read_excel("stockdatasetfinal.xlsx")
#View(StockData)
```

Linear Regression Model

First we wanted to see the correlation of all the predictors just so we could get a good grasp and what variables would and wouldn't work. Next, we wanted to see if all or only some variables correlated in predicting our response variable. So we used the regsubsets() function that will essentially perform the best subset selection by identifying the best model with a given number of predictors. So in this case we have 4 predictors in total so the function tests for the best 1 variable model all the one to the best 4 variable model. We saw that our best model was the one with all 4 predictors. To confirm this we then created a table to show the rsq, adjusted r squared, Cp, and BIC for all of the best models from 1 predictor to 4. And we then decided that we would use the model with all four predictors for linear regression.

Here's the equation for the our linear regression model:

Y = $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$, where Y represents the evm_ multiple, X_1, X_2, X_3, X_4 represent, revenue_growth, sales_efficiency, ebitda_margin and gross_margin respectively.

```r
stock_lm4 = lm(ev_ttm_multiple~revenue_growth+sales_efficiency
               +ebitda_margin+gross_margin,data=StockData)
summary(stock_lm4)
```

```
##
## Call:
## lm(formula = ev_ttm_multiple ~ revenue_growth + sales_efficiency +
##     ebitda_margin + gross_margin, data = StockData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.928  -5.814  -1.195   4.924  55.042
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -23.672      8.771  -2.699  0.00839 **
## revenue_growth     40.315      8.926   4.516 2.01e-05 ***
## sales_efficiency   11.523      4.950   2.328  0.02228 *
## ebitda_margin     -16.314      5.518  -2.957  0.00402 **
## gross_margin       32.569     12.233   2.662  0.00928 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.35 on 85 degrees of freedom
## Multiple R-squared:  0.5442, Adjusted R-squared:  0.5227
## F-statistic: 25.37 on 4 and 85 DF,  p-value: 7.596e-14
```

Because our p-value is less than $2.2X10^{-36}$. We can the reject the null hypothesis which states that $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$. This further tells us that at least one of the predictors can be used to predict the model as shown above. An average R^2 value of 52.28% was gotten which tells us that about 52% of the data can be explained bu our model. Also, we it is evident from the model that each predictor has some sort of significance. However, because we know that increasing the number of predictors increases the R^2 value, we tried to get the best subset of predictors and all signs point to using the 4 variables as our predictors

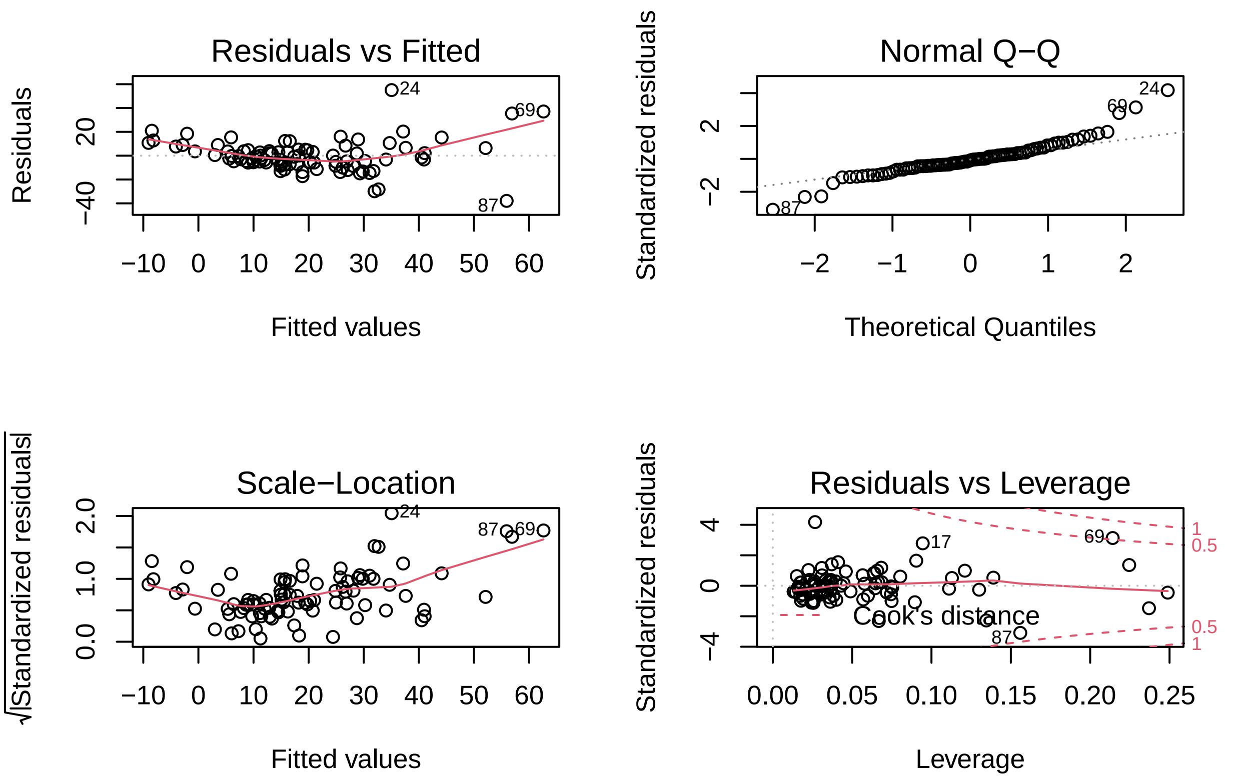
```r
library(leaps)
stock_fit = regsubsets(ev_ttm_multiple~revenue_growth+sales_efficiency
                       +ebitda_margin+gross_margin,data=StockData)

stock_res = summary(stock_fit)
stock_stat = cbind(stock_res$rsq,
                   stock_res$adjr2,
                   stock_res$cp,
                   stock_res$bic)
colnames(stock_stat) = c("rsq","Adjr2","Cp","BIC")
stock_stat
```

```
##            rsq     Adjr2        Cp       BIC
## [1,] 0.4392949 0.4329232 18.562532 -43.07079
## [2,] 0.4891809 0.4774379 11.259586 -46.95715
## [3,] 0.5151324 0.4982184  8.420053 -47.14990
## [4,] 0.5441968 0.5227472  5.000000 -48.21342
```

To select the best model, we need to look at these 3 statistics. The C_p value must be value close to the number of predictors, The BIC, should be a low value and the Adjusted R^2 should be as high as possible. Based on the following statistcs, it is easy to conclude that the model woth 4 predictors is the best model to fit our data.

```r
par(mfrow = c(2,2))
plot(stock_lm4)
```



Hence, this is the equation for the model
ev_ttm_multiple = -23.888 + 40.315Xrevenue_growth + 11.526Xsales_efficiency - 16.316xebitda_margin + 32.586Xgross_margin.

From this equation, ebitda_margin will negatively affect the ev_ttm_multiple while all others wil affect it ositively. To be more precise, for every %increase in revenue_growth, sales_efficiency and gross_margin , the ev_ttm_multiple increases by 40.315, 11.526 and 32.586 respectively while the ev_multiple decreases by 16.316 with a percent increase in the ebitda_margin.

We trained our model based on an 80%-20% division in order to calculate the MSE in order to get its prediction accuracy and perform cross validation. Here are our findings

```r
set.seed(10)
errors = rep(0,10)

#TO GET MSE'S
for (i in 1:10){
  sample = sample.int(n = nrow(StockData), size=round(.80*nrow(StockData),0),replace=F)
  training_data = StockData[sample,]
  testing_data = StockData[-sample,]
  my_fit = lm(ev_ttm_multiple~revenue_growth+sales_efficiency
              +ebitda_margin+gross_margin,data=training_data)
  errors[i] = mean((StockData$ev_ttm_multiple - predict(my_fit,StockData))[-sample]^2)
}
errors
```

```
##  [1] 144.94456 321.94547  75.82288 139.96263 269.92279 221.88979 207.28650
##  [8] 242.51262 205.00650 286.05096
```

```
mean(errors)
```

```
## [1] 211.5345
```

Upon doing 10 iterations, we arrived at a wide vaiety of MSE values. MSE are important because it is a means of showing how far off in prediction are we. Statistically, it tells us the square of the difference between the predicted and the actual observation. Our MSE values range from a low of 75.8 to a high of 285.90 which shows how fairly accurate the model is in predicting the ev_ttm_multiple.

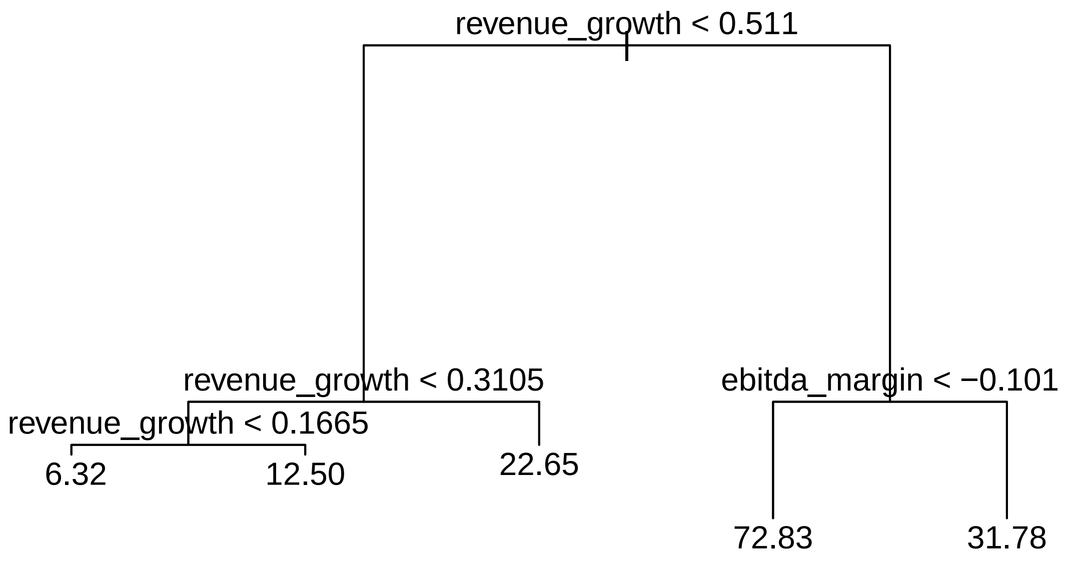
Decision Tree

Decision trees are easier to interpret than regression models. That is the mean reason why we are using a decision tree. However we have to remember that decision trees perform worst in prediction that most regression model. This code makes a tree

```
set.seed(20)
library(tree)
tree.model <- tree(ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_margin, data=trai
summary(tree.model)
```

```
##
## Regression tree:
## tree(formula = ev_ttm_multiple ~ revenue_growth + sales_efficiency +
##     gross_margin + ebitda_margin, data = training_data)
## Variables actually used in tree construction:
## [1] "revenue_growth" "ebitda_margin"
## Number of terminal nodes:  5
## Residual mean deviance:  127.2 = 8525 / 67
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -33.830  -4.715  -1.210   0.000   4.050  27.620
```

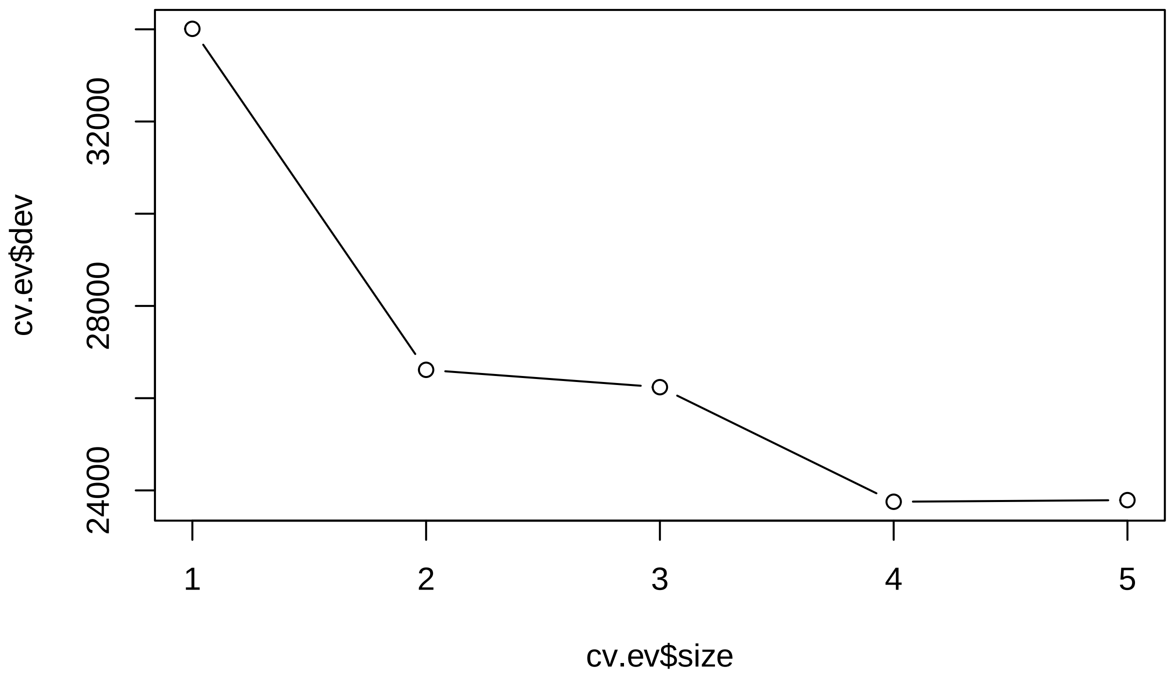
```
plot(tree.model)
text(tree.model)
```



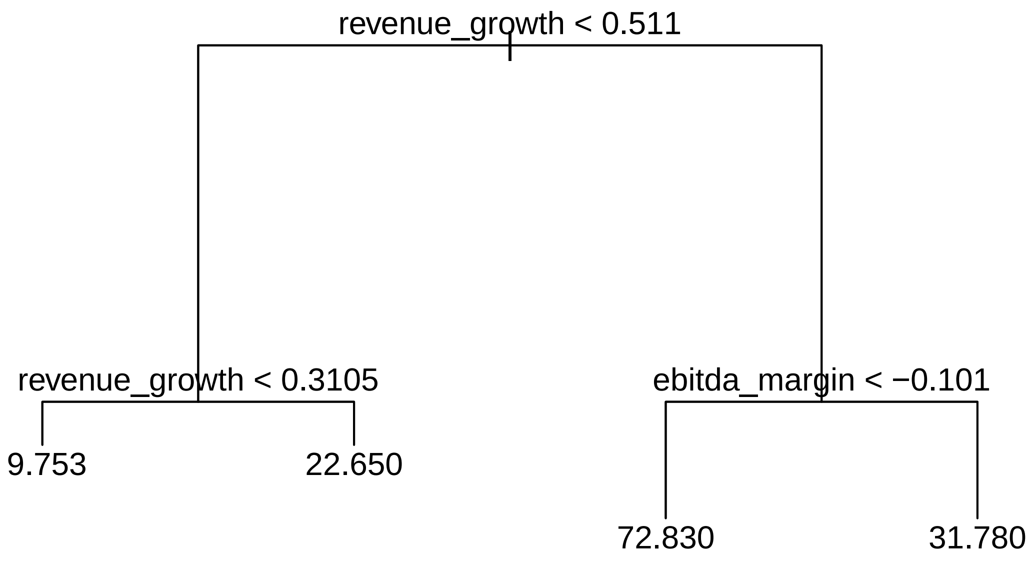
Prune Tree

Then we proceeded to prune the tree at the best node using this code.

```
cv.ev=cv.tree(tree.model)
plot(cv.ev$size, cv.ev$dev, type="b")
```



```
pruned_tree = prune.tree(tree.model, best = 4)
plot(pruned_tree)
text(pruned_tree)
```



We can see that revenue growth, sales_efficiency and ebitda_margin are the predictors with the most siginifance.

Next we are interested in checking how well this model predicts our data in comparison to our linear regression model.

The following code does a cross validation on the tree model

```
library(tree)
mse.tree = rep(0, 10)
for (i in 1:10){
set.seed(i)
sample<- sample.int(n=nrow(StockData), size=round(.80*nrow(StockData), 0))
train<-StockData[sample, ]
test<-StockData[-sample, ]
tree.model <- tree(ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_margin, data=trai
yhat.tree=predict(tree.model, newdata=test)
mse.tree[i]=mean((test$ev_ttm_multiple - yhat.tree)^2)

}
mean(mse.tree)
```

```
## [1] 252.2586
```

The obtained MSE is 252.2586 which is much higher than the one we have for the linear regression model.

Random Forest Model

Although three-based methods are simple and easier to interpret, they lack the accuracy provided by regression models. For that reason, we decided to use a random forest model to add more accuracy to our prediction. A random forest is a good choice because it reduces the variance introduced in decision trees and tree bagging. However, using random forest comes at a cost of longer training periods thus there is a higher computation time and we cannot see the trees so we loose the simplicity of single decision trees.

The following code does a random forest using ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_m

```
p=4
B=100
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.2

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.
```

```
rf.model=randomForest(ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_margin, data=S
rf.model
```

```
##
## Call:
##  randomForest(formula = ev_ttm_multiple ~ revenue_growth + sales_efficiency +      gross_margin + ebi
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 248.1106
##                     % Var explained: 32.85
```

From this preliminary model, we get an MSE of 228.22 and and explain about 38.23% of the variance in the data set. Seeing that this value is lower than the MSE for the decision tree, we go on making a training and a testing set.

```
library(randomForest)
mse.randforest = rep(0, 10)
p=4
B=100
for (i in 1:10) {
set.seed(i)
sample<- sample.int(n=nrow(StockData), size=round(.80*nrow(StockData)))
train<-StockData[sample, ]
test<-StockData[-sample, ]
bag.model=randomForest(ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_margin, data
yhat.randforest=predict(bag.model, newdata=test)
mse.randforest[i] = mean((test$ev_ttm_multiple - yhat.randforest)^2)
}

mean(mse.randforest)
```

```
## [1] 217.6153
```

The MSE obtained is 217.2678 which is still lower than decision tree MSE, proving that random forest is better in prediction than decision trees.

Boosting

Note: This is not part of our project but we were curious to see how a boosting model would perform compared to our other models. The following does boosting and shows the MSE.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
mse.boost = rep(0, 10)
for (i in 1:10) {
set.seed(i)
sample<- sample.int(n=nrow(StockData), size=round(.80*nrow(StockData), 0))
train<-StockData[sample, ]
test<-StockData[-sample, ]
boost.model = gbm(ev_ttm_multiple~revenue_growth+sales_efficiency+gross_margin+ebitda_margin, data=trai
yhat.boost=predict(boost.model, newdata=test)
mse.boost[i] = mean((test$ev_ttm_multiple - yhat.boost)^2)
}
```

```
## Using 100 trees...
```

```
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
##
## Using 100 trees...
```

```
mean(mse.boost)
```

```
## [1] 182.79
```

Conclusion

The multiple linear model was our best model as it had the lowest MSE. The linear regression was able to give us more insight into which predictors are more associated with a higher revenue multiple. Revenue_Growth was our best predictor and sales efficiency was the second best. Decision Tree was the best model in terms of inference. We were surprised that the Random Forest has performed worse than the Linear Regression. It is unreasonable to expect any model to predict the value of a company with high precision, however that are clear patterns that our models bring to light. It is logical that revenue growth plays a large role in the value of a SaaS company, considering that revenue is a company's oxygen. One of our most surprising finding was to see that ebitda_margin % was somewhat negatively correlated with a higher multiple. A lower ebitda_margin indicates that a company is "losing money," but it may be doing so to prioritize revenue growth which we found to be the most predictive. There is an expression in SaaS - grow at all costs and that may be true. Though it was not part of our plan, we performed Boosting and we found out that it was the best model for the data.

To tie everything back to why we chose to explore this data, every one of our group members plans to work professionally in software after graduating. Stock is often big part of compensation packages. If any of us are considering a job offer from a SaaS company we will definitely be looking through their financials to see what kind of revenue multiple they should have.

Endnotes

Analysis for the linear regression model was performed by Oluwatobiloba Oladunjoye and Michael Thomas. Analysis for the random forest and boosting regression model was performed by Mahamadou Dagnoko and Min Shwe Maung Htet. Decision Tree analysis was performed by Rafay Alam and John Jackson. This report was written by Mahamadou Dagnoko, Oluwatobiloba Oladunjoye, Michael Thomas, and John Jackson. ´diting done by Rafay Alam and Min Shwe Maung Htet. To view our data set and source code, visit our GitHub repository https://github.com/johnjackson59/saas_multiples.