Metrics of successful websites and companies

Danai Avratoglou

January 2017

Contents

1 Introduction

Over the past two decades, the uncanny growth of the World Wide Web (the Web or WWW), along with the expansion of the users that have access to this medium attracted the attention of enterprises and organisations. The popularity of the Web has created new opportunities for companies to attract new customers or even retain their existing ones through their websites.
Even though before the expansion of the Web the on-line presence of a company was not an important factor of the overall success that the enterprise would have in nowadays the vast spread of the impact that internet has on consumers, regarding their choices, render this hypothesis invalid.
Companies are obliged by the trends to be active on-line and to maintain a website that depicts the image they want their consumers to perceive. By creating a more consumer oriented website they lead the users to create a positive idea regarding the company and this could potentially lead to the increase of their revenues.
The purpose of this paper is to understand the relationship that exists between a company's website and the overall company's success. A measure that can be comparable between enterprises are their revenues. Trying to comprehend this relationship a comparison will take place between the enterprises that were deemed as the more successful ones from Fortune 500 (based on their revenues) in 2016 and a number of website metrics in order to understand through the performance of statistical analysis which of those metrics are correlated the most with the company's success.
In the chapter *Literature Review* a retrospect of previous studies will take place and the thesis hypothesis will be presented. Moreover in the chapter *Data Gathering* an analytical description of the way that the data were collected from the website pages will be take place. Furthermore in the next chapter called *Data Analysis* the steps that were followed for the performance of the analytical part of the paper will be explained as well as the finding of this analysis. Finally in the *Conclusion* chapter there will be a summarized version of the papers findings and its contribution to the already existing researches. The last chapter *Further Research* will give some ideas of what the next steps should be regarding the current state of the field's research.

2 Literature Review

The World Wide Web was created and developed in the United States of America and initially it was used from the government for educational and non commercial research institutions only. The first commercial use of the Web was not until 1993.[9] From then on there was an explosive growth that led to the existence of more than 200 million sites by 2005.[19] By now this number has been increased even more and it keeps growing daily. Nowadays it has become a part of the every day life of an average consumer. Taking into account this fact the research around the impact of the web pages into an organisation success is undoubtedly an interesting topic for examination.

2.1 Empirical Studies

For organizations the corporate website has emerged as one of the most important interfaces from where the consumers can infer a complete opinion of what the firm is standing for. Moreover there are many previous studies that have investigated the correlation of the website existence and the company's success. A content analysis of a website features was conducted in 1997[15] where the authors examined the companies of the Fortune 500 to see how many of those had web sites, in which industries did they belong, whether or not where there any differences in the revenues regarding the existence of a site and content of the pages. The findings of this research showed that even in 1997 the two thirds of the companies in the list had already set up their web pages regardless their industries. A research that was conducted two years later[25] again regarding the Fortune 500 sites showed that by then only 10 of the companies that where include in the 500 most successful ones in the U.S. continued to not have a web page. This findings shows a very high correlation of successful companies with the use of web sites.
Furthermore in 2006 there has been a study that compared the use of websites in the top 1000 successful companies between two different countries[13] U.S. and Taiwan in order to understand the diversities in the web practices. This time the firms that were examined derived from the Fortune 1000 since the specific source was established in both countries. The results showed that the countries are in different stages regarding the maturity of the internet usage for consumer attraction. This shows that the best way to examine the upcoming trends is by concentrating in the most mature countries and to study the correlations there.
In the attempt to provide information regarding a vast number of different aspects of a website a great number of studies have been taken place.
Researchers thought that the internet and more specifically the Web is the best platform to attract more visitors and to reach new clients.[10] Based on this idea there have been conducted many studies that examined the best way to create websites. They believed that a consumer friendly design could add value to the users. This assumption was confirmed from previous studies that showed that a user interface that is perceived as very good is a vital factor in the retention rate of the visitors.[8] Of course in order to examine had exactly constitutes an

excellent interface new studies had to take place.Wei-Shang Fan and Ming-Chun Tsai[23] in 2010 studied the relation between the internet customisation with the web design and the internet marketing strategy. This research showed that the web design quality can directly influence business performance.
The website design can be separated into six dimensions[14]:

1. **The information quality** : Researchers have suggested that the information that is include in a web site should be relevant to the purpose of the website[4, 6], to provide an adding value to the consumer by being useful and educational[12] and last but not least to be easy to read and to comprehend.[16]

2. **The navigational quality** : This dimension is referring to the way the website has organized the information and how it has arranged them in terms of design, layout and sequence. The aspects that are included in this dimension and effect the ease with which the website can be navigated are the number and the effectiveness of hyper links and of course the overall organization of the information.[1, 18, 21, 22, 24]

3. **The entertainment quality** : The extent to which a website is visually appealing[16, 24] seems to be an indicator of how easy it is to use it. Another factor that makes the website easy to use is the use of graphics, multimedia and other interactive elements.[2]

4. **The system quality** : By referring to the system quality the technical properties of the website comes to mind. For example the ease with which the user can access the website. The access speed is characterized by how fast can the website download its pages and how fast it can display them.[16, 19, 20]

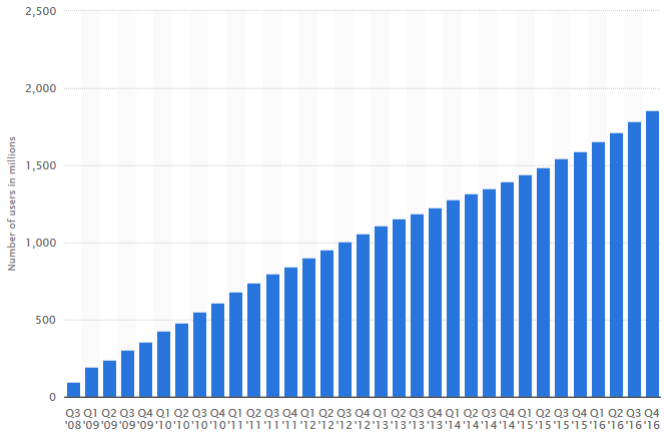
5. **The system use** : This dimension is also correlated with the technical properties of the website and is referring to the accessibility or the availability of the site.[22] The importance of accessibility can be observed from the fact that if the site is not available on a sustained basis, browsers would not be able to return to it.

6. **The service quality** : Finally the service quality dimension is referring to the capabilities and the option that the website is offering to their users.

Another important part of the Web usage from enterprises that should be taken into account is the use of the social media. One of the most representative definitions[11] of social media is the following: "*a group of internet based applications that builds on the ideological and technological foundations of Web 2.0 and it allows the creation and exchange of users generated content*". This particular definition implies that the content of the social media is not consumed passively by the users but instead they react and even create their own content with their own views. This means that firms should actually participate in the social media fever if they want to reach out to their consumers and even to

create new ones. Social media have become incredibly popular in a global level in the last few years. Facebook alone, a hallmark of social media has 1.86 billion active users per month as of the fourth quarter of 2016.[26] As active users are characterized those which have logged in to Facebook during the last 30 days. Also by the end of 2015 more than 50 million companies had already established their fan pages in Facebook.[33] In the following table the quarterly evolution from 2008 of the Facebook users is available:

Table 1: Facebook users by quarter



Furthermore, due to this rapid evolution of the use of social media many different platforms have been created in the last few years. Some of the most popular ones, besides from Facebook, are:

- **Twitter** : As of the fourth quarter of 2016, the microblogging service 319 million active users per month.[27]
- **Instagram** : As of December 2016, the mainly mobile photo sharing network had reached 600 million monthly active users.[28]
- **Pinterest** : As of October 2016, the content sharing service has reached 150 million monthly active users.[34]
- **Linkedin**: As of the third quarter of 2016, the business and employment-oriented social networking service had reached 467 million members.[30]
- **Youtube** : As of January 2017,the 3rd most visited website in the world has reached over 1,3 billion active users.[31]

Our goal is to show how all those aforementioned metrics can be used to show the relevance of a website with the success of an enterprise. We can now develop our hypothesis.

2.2 Thesis hypothesis

While there are quite a few previous researches that tried to correlated the success of a website with the overall success of a firm, most of the findings were derived from the perceptions of web users or web designers. Furthermore most of the variables at hand had to do with how those groups perceived the impact of them on their choices and not the actual prices of the variables that the company decided to use.
This paper extends beyond previous studies by devoting most effort on examining specific web site metrics that were deemed important from previous researches, not from the users perspective but from the actual metrics prices in relation to each enterprise's revenues.The questions that this study would try to answer are the following ones:

1. Do the revenues of a company relate to specific metrics of the company's website?

2. Which of the metrics under examination are correlated the most with the revenues of each company of the Fortune 500 ones?

3. Can there be a predictive model that, based on the prices of the metrics that will be characterized as the more important ones from the findings, could forecast the revenues of an enterprise?

3 Data gathering

3.1 Data Source

The first step in order to contact this research is to find which companies will be examined. Since the purpose of this paper is to see if the website metrics that will be chosen are influencing the success of a company it is a good idea to examine websites of some already successful firms and try to find out what they have in common. Moreover we want to make this examination in a country that has already reach to a certain maturity regarding the use of the internet for organizational purposes. Thus, we will examine the 500 companies that were ranked as the most successful ones from Fortune 500.
The Fortune 500 is an annual list compiled and published by Fortune magazine that ranks 500 of the largest United States corporations by total revenue for their respective fiscal years. The list includes public companies, along with privately held companies for which revenues are publicly available.[9, 13]
For the purposes of this paper we will use this list of companies and we will examine specific metrics of their websites in order to understand if there is a correlation between a firm's success and it's on-line presence.
In order to gather the need information from all the sites in the Fortune 500 we need to create an actual list of the names that are include in 2016 Fortune 500 ranking. In the variables collection section the way that the list will be obtain will be explained in detailed. In the following table the top 20 highest ranked enterprises of Fortune 500 during 2016 are available. The complete list by ranking is available in the Appendix AA.1.

Table 2: Fortune 500 - 20 first companies

1. Walmart	2. Exxon Mobil
3. Apple	4. Berkshire Hathaway
5. McKesson	6. UnitedHealth Group
7. CVS Health	8. General Motors
9. Ford Motor	10. AT&T
11. General Electric	12. AmerisourceBergen
13. Verizon	14. Chevron
15. Costco	16. Fannie Mae
17. Kroger	18. Amazon.com
19. Walgreens Boots Alliance	20. HP

It is obvious even from the first twenty companies that the firms that are included in the Fortune 500 list are from different industries. In previous studies the researchers tended to take into account the industry and divide the companies in groups. In this paper we will not create such a deviation since we want to see if there are some common variables across all the successful enterprises, regardless the industry they belong to, that can correlate with their success.

3.2 Metrics

The next step is to identify the website metrics that will be examined. When a firm is referring to website metrics it usually means elements such as the number of page views, the number of visits, the number of unique visitors and also the geographical distributional of the users. These types of measurements are not available outside of the company, which is usually using a tool such as Google analytics[37] in order to track them. Since we cannot gather this kind of metrics we will have to examine metrics that are more related to how the site is structure and what exactly does the home page of each site includes. Those information can be retrieved from the html code of a company's site which is available for each internet user to see.
The two main categories in which we can divide the type of elements that we will retrieve from the website's html code are the following ones:

- **What we see:**
  In the first category we are referring to metrics that can easily be conceived by the naked eye as well. For example the number of images that a website is using in its landing page and their dimensions. From the empirical studies that have taken place we saw that these variables have been examined as part of the entertainment quality measure. This means that previous studies examined the impact that the existence of images had on the users and not the relation that the actual number of images that are being used have with a company's revenue. In other words researchers have already examined the users perspective of what constitutes a visually pleasant website but not what do the firms decide to actually implement on their websites.

- **What lays behind of what we see:**
  The second category is not so obvious and it includes informations that usually is visible only to the web developer or the creator of the page. The information can only be taken by the html code and not with the naked eye. For instance we can see the type an image, an information not visible with the naked eye.

Now that we have a first understanding of the two main categories that the metrics can be divided in, we can see in detail the metrics that will be examined in this paper:

3.2.1 Loading time

One aspect of a website that is crucial is the time it takes for it to load. Nowadays that the internet speed is going higher and higher most people do not have the patient to wait for a page to load. Several user experience studies evaluate how page load times impact user satisfaction. The ill effects of slow websites are well documented. Recent surveys suggest that 49% of users will abandon a site or even switch to a competitor after experiencing performance issues.

This highlights the importance of including this measurement to the study and determine it's relation with the company's status in terms of revenues.

3.2.2 Number of links

The overwhelming information that is available on line can render a user incapable of finding the piece of information he is looking for. Many internet users tend to browse across different sites searching for a particular information that they cannot remember where they have found it before. Considering this tendency firms should create hyper links that will make the navigation into the site easier for the user but will also give him the opportunity to grasp the highest level of information that he can. This translates to the existences of internal but also external hyper links. An internal link can direct the user in another page of the same site while an external link can lead the user to another site.
The internal links are responsible mainly for the ease of navigation in the website. The better their structure the easier it gets for the user to find what he is looking for. The external links are mainly responsible for the quality of available information. Based on the industry that a firm belongs to, it has to provide the users with information that can be relevant not only for the specific company but also for the whole industry. Sometimes those informations are not available or cannot be available in the enterprise site so the firm should be proactive and create hyper links that will lead the user to the information that he is looking for regardless if this can be found in their site or not. With this approach the user gets a feeling of satisfaction that is connected with the firm's website that helped him find the information he needed.
For the purpose of this paper since it is not so clear which type of links are more important to a user we will examine both the internal and the external links and moreover the total links of a website to see which of those three indicators is correlated the most with a firm's success.

3.2.3 Social media

Our era is marked by the social media wave that has changed our lifestyle and our daily habits. So it would be considered an oversight if we didn't take under consideration the number of social media that the company chooses to participate in. Even though they can also be considered as external links of the website we will examine them separately in order to see if any particular social medium effects the company's revenues. Since there is an ongoing debate on whether social media should be used from firms to attract customers or if their use should be only for connecting people with each other and not brands the results of this paper would give a perspective of what is really happening in terms of numbers and not from the users perspective.

3.2.4 Number, type and sizes of images

Since the site is the first thing that a user will see on line regarding a company and there is a famous quote that says that *"First impressions counts"* we should

also examine how the companies decide to visualize their landing page. In other words to see how many images they include in it and more on that what are the dimensions of those images.
It is completely different to see only one huge image in the landing page of a site with not many words or descriptions than to see many small images with different information. The purpose is to examine if these type of diversities between the examined websites are actually related to how they are doing success wise.
Moreover an information a little more complicated for a simple user to understand, but which plays a very important role in many cases is the type of the image. For example some websites are using specific type of images or banners that are not compatible with all the browsers, leading the user to see some break points in the website and even stop visiting it. Based on those facts we will examine the most commonly known type of images and see in which degree they are being used from the companies under examination and whether or not there is a correlation with the revenues.

3.2.5 Content

They say a picture worth a thousand words but that is not enough in our case. After exploring the number, sizes and type of pictures that a website is using we should also explore the number of words it is using to accompany the images and complete the outcome that a user will come across. The metrics we will use will be two. The first one will be the total words that are being used in the landing page and the other one will be the total unique words that are being used. When we are referring to unique words we mean words that are not so commonly use such as "a" or "and" and they give an air of individuality to the text. By using this metric we would try to see if the words that are being used are just as important as the actual content and if the words can make a difference.
Furthermore we should also take under consideration how comprehending is the text used in the websites for the users. This information can be obtained by calculating the readability index of the website and also the number of sentences that exists in a page (a metric that comes to complete the previous ones).How the readability index is being calculating will be furthered explained in the variables collection section.

3.2.6 HTML Validation

Moreover we will have to check the quality of the html code behind the website we are seeing. Are there any mistakes in the code for example any brackets that opened and never closed or any links that do not work. We will examine again two different metrics here the number of errors and the number of warnings. The warning are parts of the code that even though they work at the time there is a good chance to malfunction if any changes or addition are to be made to the html code.

3.3 Python Language

After explaining the reasons that we decide to explore the variables/ metrics that were mentioned in the previous section we should now see how we are going to obtain all these information.
Since the needed information can be subtract from the html code of a company's website we should use a programming language in order to download the html pages and then to extract the specific metrics we want to examine.
For the purposes of this paper the programming language that will be used for downloading the website's html code and then extract from them the metrics is Python. More specifically the version of Python that will be used is the 2.7 one.[41]
Python is a widely used high-level programming language used for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using white space indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.
The language provides constructs intended to enable writing clear programs on both a small and large scale. Furthermore the way that Python allows a user to programming is common to all users which gives this language a leverage as a program build in Python can be easily understood from another user without any difficulty.
The environment that is going to be used is from the Anaconda package which is a free open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. To be more precise from this package we are going to use the Jupyter Notebook. All the scripts that were created in the Jupyter Notebook are available in the Appendix.A.2

3.4 Variables collection

In order to gather all the needed metrics we had to create a variety of small scripts so as to collect them. In this section we will present in detail the procedures that were used in order to create the scripts and extract the information that later will help us contact the analysis of the relationship between those metrics and the company's status.

3.4.1 Companies ranking, names and url

For starters we need to create a list with the names, the ranking and the URL of the sites that we will later download and extract the necessary info from them. So as it was mentioned in the previous section3.1 the first step that needs to be done is to download and gather those informations into a data frame[1] so as to be able to use them later on.
The easiest way to obtain this list is by retrieving it from an already existing list that has been created in an article on line.[32]. The way to keep only those three informations as different variables is by separating from the html code of this page the needed elements.

Step 1 : The first step is to create three empty lists where we will include the informations we are going to extract. The first list will contain the rank of each site as a number, the second one will contain the name of the company as a text and the 3rd one will contain the actual link of the company's site again as a text and without the http:// in the beginning.A.2

Step 2 : The second step is to upload some libraries that will help us create this function but also the rest ones that are going to follow.A.2

Step 3 : Finally the third step is to create the function that will firstly download the html code of the url at hand, secondly keep only the part of the code that we need to examine and thirdly save this part into the empty lists we created above. This function is called websites and takes as variable to work only the url of the site we need to examine.A.2. More specifically the methodology we followed to create the following function are the following:

1. We create a fake browser that we are going to use in order to open the page and downloaded. The reason we do that is that many sites do not allow us to download their page because they are afraid of stealing important information. Since we are not using any private information we use this method to avoid issues while trying to open the html page at hand.

2. We open the url and we read it while saving it in the variable "myHTML".

3. With the help of the Beautiful Soup library[2] we read the page as a lxml

[1] a table in Python environment with rows and columns
[2] Beautiful Soup is a Python library for pulling data out of HTML and XML files

file and then for each row of this file we are looking for the "td" parts of the code where the informations we want are included.

4. Since we need the names and the urls of all the 500 sites we created a loop from 0 to 500 where for each i we try to isolate the part of the code that contains the information that we want. Moreover even thought it seems that with this loop we calculate 501 numbers since in Python the second bracket is always open we actually count from zero to 499.

5. We use reg expressions[3] in order to state precisely what part of the already selected code we want to keep.

6. We insert with a specific order the names, the ranking and the url to the corresponding lists and finally we create a text that will appear when the function is completed. Here we have also calculated the time that this function took to be completed and we will appear it as well along with the text.

3.4.2 URL validation

Now that we have saved in the three lists the names, the ranking and the URL of the companies that we are going to examine we first have to check whether or not those URL are valid in order to proceed with the download of the html code behind the initial web page of each one of those companies.

Step 1 : We first have to install the validators package in python so as to proceed with the validation of the url. In the command prompt window that opens when you open the Jupyter Notebook you have to write: "pip install validators" and then press enter in order for the package to be installed. This specific package currently supports python versions 2.7, 3.3, 3.4, 3.5 and PyPy.[4] The function that we are going to use from this package is called validators.url. This functions returns True if the url at hand is a valid url or False if it is not.

Step 2 : Next we created a loop that will run as many time as the length of the list that we created in the previous section.

Step 3 : During this loop we create a string variable where we use the URL that we have saved in the list, one at each time, and we add the "http://" prefix.

Step 4 : Now that we have create the correct way that a url is supposed to be written we will use the function of the validators package and we will create an if function that will check if the answer to a site is different from True and if it is it would add a unit in the variable nv so as to know at the end how many sites did not had valid URL.

[3] A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

[4] https://validators.readthedocs.io/en/latest

Step 5 : Finally we ask from the programme to print the final result so as to know how many URL are not valid. The code that was created for this procedure is available in the Appendix.A.2
In our case all the URL where valid so we are eligible to proceed at the next part of the code.

3.4.3 Download sites html code

After checking the validity of the URL that we have saved in the list the next step is to download the actual html code of the initial pages of each one of the 500 websites we want to examine.

Step 1 : As we did in the section 2.4.1 we have to create a fake browser so as the to be able to download the html code without contacting any problems. In our case we created a Mozilla browser.

Step 2 : In order for the url to be opened we must first bring it to the correct form. Initially we create again a loop and for each loop we will examine a specific url. We first replace some symbols on the string that will not be recognized if we try to open the url in this form and then we add the "http://" in the begging of each url.

Step 3 : Before starting downloading we create some rules for some exceptions. For example the sites 71 (Best Bay),119 (Arrow Electronics) and 465 (St. Jude Medical) in ranking create problem when we tried to download them and the code is stop working. So in order to avoid such incidents we created an exception and the python code will not even try to download these sites and in their position in the new list with the html pages that we will create a zero will be saved.The same thing would happen if an exception is thrown in the function if that we are creating. In any other case we will open the site and save this action in the variable response2.

Step 4 : After saving the action we will use the command read and we will save the html code which is essentially a long piece of text and then we will save this "text" in the list.

Step 5 : Finally we will wait for 2 seconds in order for the browser not to be suspicious from the extreme speed that we are going to open the next page. In that way we avoid having any crushing incidents on the code.

Step 6 : After completing this procedure for all the site we will have a list that in each position the html code will be held as a very large text. Except from the sites that were not able to be opened.
The code that was created for this procedure is available in the Appendix.A.2

3.4.4 Not downloadable pages

As we explained in the previous section there are some pages that were not able to be downloaded. In order to know which are those pages and more specifically which companies sites will not be available for further exploration we created this part of the code that creates a list with the names of those companies.
In the previous code we used a zero in each position that we weren't able to download the site. Here we will use this information in order to gather in a new list only the parts of the list that did get a zero.
Here we created a small function that is called "not downloadables". In the first part of the code we create the function and in the second part we run the function for our lists. Finally we create a data frame with the results so as to be more easy on the eyes.
The code that was created is available in the Appendix.A.2 In the following table we can see the sites that were not downloaded:

Table 3: Not downloadable sites

16. Fannie Mae	63. HCA Holdings
71. Best Buy	91. Nike
98. Tesoro	119. Arrow Electronics
136. AutoNation	142. Southwest Airlines
162. Southern	165. American Electric Power
196. Office Depot	217. PBF Energy
229. Consolidated Edison	240. Toys R Us
243. Dominion Resources	276. Global Partners
307. PayPal Holdings	327. News Corp.
364. Williams	415. Tractor Supply
442. Old Republic International	465. St. Jude Medical

3.4.5 Content

Now that we have downloaded the html code we should start extracting some of the variables we are going to use for the analysis in the next chapter.

Step 1 : The first thing we are going to check is how easy it is for a user to read the content of each site. In order to check this performance indicator we will need to gather 4 variables:

1. Number of Words
2. Number of Unique Words
3. Number of Sentences
4. Flesch score

The three first are quite clear. The number of words are the total words that appear in the texts that the user is seeing in a website. The number of unique words are the words that are not very common and can attract the readers attention or even make it harder for him to comprehend the text. The number of sentences is related to those variables as we can understand from this is comparison to the number of total words how big or short is the sentence and as a conclusion how easy or not is the complete text.
Finally regarding the forth variable the flesh score is referring to the Flesh reading ease test score.In the Flesch reading-ease test, higher scores indicate material that is easier to read; lower numbers mark passages that are more difficult to read. The formula for the Flesch reading-ease score (FRES) test is:

$$206.835 - 1.015(totalwords/totalsentences) - 84.6(totalsyllables/totalwords)$$

We will calculate the flesh reading test and the other three aforementioned variables with the help of an on-line readability test tool.[36] This site calculates and return all these results that we need. The way to implement them in a new data frame to our python code is by dividing the html code, behind the page with the results for each site, and keeping only the actual numbers and sizes that we seek.

Step 2 : We create four lists where we will save the variables that we are looking for. Then we start a loop for the 500 sites where in each loop we change the value of the 3 main variables the sites that contain the html code the url that contain the url of the company at hand each time and the url check where we save the part of the web address that remains the same while doing the on line check and we add at the end the url of the site we want to check. Also we open the browser as we have done in previous scripts as well.

Step 3 : Then we create an if part where we check that the variable site is not zero and that we will not examine the site 108 (Tech Data) as the code seems to have a problem in that case. In this check we put on the respective sites n/a values so as to be clear that we did not retrieve these info for them.

Step 4 : We open the browser link and we check whether or not we drop to any exception in which case we also put n/a values in the respective companies.

Step 5 : We read the link and after checking that it is not empty (this we can check by comparing it with an empty list that we have created for this use) we use the Beautiful soup library to extract the part of the code that we need.

Step 6 : After taking a look at the html code of the pages we find out that the parts that we need are between the following brackets ¡tr¿...¡/tr¿ so we will extract each such bracket from the code and save them in a list. By checking the list we found which numbers of the list items we want and we saved them in the variables we have created. Always of course checking first that the specific

prices exist or in any other way we again put n/a.

Step 7 : Now that we have the for key indicators that we will use later on the analysis we should also create a more easily comprehend variable that is related to the flesh measure variable we just created. The numbers that each site gathered mean different things. So we will try to create a correlation and build a variable called readability that will say in text how easily read or not a site is. The score can be interpreted by the following logic:

- Flesch measure < 30 : Very Confusing
- Flesch measure > 30 : Difficult
- Flesch measure > 50 : Fairly Difficult
- Flesch measure > 60 : Standard
- Flesch measure > 70 : Fairly Easy
- Flesch measure > 80 : Easy
- Flesch measure > 90 : Very Easy

Step 8 : We created a function that creates a new list where the flesh measure variable is being saved as the above descriptions based on the scores each of the site achieved.

Step 9 : Now that we have all the needed informations in lists we should combine them by creating one data frame with the use of the variable company name again in order to have a common key so as to merge all the data frames that we will create in the end.
The code that was created is available in the Appendix.A.2A.2

3.4.6 HTML Validation

After downloading and saving in data frames the first batch of metrics that we will need for the analysis we should also check the quality of the html code.
Most pages on the World Wide Web are written in computer languages (such as HTML). One of the advantages of writing in a computer language is that it allows the developer to structure text, add multimedia content, and specify what appearance, or style, the result should have based on his needs.
As every speaking language,so the computer languages do have their own grammar, vocabulary and syntax too. Each document that is written in these computer languages is supposed to follow these rules in order to consider it well structure and written.
However, just as texts in a natural language can include spelling or grammar errors, documents using computer languages may (for various reasons) not be following these rules as well.
The process of verifying whether or not a document actually follows the rules for

the language it uses is called validation, and the tool used for that is a validator. A document that passes this process with success is called valid.
With these concepts in mind, we can define "validation" as the process of checking a web document against the grammar (generally a DTD[5]) it claims to be using.
For the purposes of this paper we will use an on-line validator site called W3C[6] which will help as see how many errors and warnings does each site have.

Step 1 : Initially we create the empty lists in which we will save the variables we will extract that will give as a clear glance on how many errors does each page has, how many warnings, how many pages weren't recognized as documents and how many pages did open during the process.

Step 2 : The next step is to create a function that will do a similar job as the script we used to extract the previous variables. Initially we locate the part of the url that remains the same when the check is completed and we see the part that does change where the name of the url we want to examine should go. In order for the code to recognize that we save the result in a variable that in the end is a new url that when we open it, it will show as the page of the results for each site in each loop that we are examining.

Step 3 : One difference in this script is that now the part of the code that we need is not between $< tr > .. < /tr >$ but between $< div > ... < /div >$ parts of the code. So we follow the same procedure as before and we locate in the list of all the divs that we create with the help of the Beatuful soup package the parts that gives us the informations we want to keep.

Step 4 : One other difference is that here after the procedure ends we have counted the actual time we needed to complete this procedure. Then next step is to run the function we created.

Step 5 : Finally we save the lists that we created in a new data frame where the common column remains the name of the company as it already is in the previous dataframes we have created.
The code that was created is available in the Appendix.A.2

3.4.7 Social media

One other group of variables that we should include in our analysis is the social media that each company choose to use. Since this era is being characterized from the massive use of social media we can't help but wonder if some of them could actually play a crucial role in a business success.
The way we are going to see which social media does a company use is quite simple. We will create a list with the name of each of the six most well known

[5]DTT: Document Type definition
[6]https://validator.w3.org/

social media with the suffix ".com" and then we will check if the expression appears in the html code of each company. If an expression does exists that means that the specific company does indeed uses this specific social media and as so it appears a respective link in it's home page so as the user to have the opportunity to subscribe in it.
The procedure we will use is similar to the previous ones.

Step 1 : Firstly we create the lists for the social media that we want to examine. More specifically the social media we will search for are the following:

- **Facebook**: a social media that lets you upload images, thoughts, songs and lets you interact with your friends
- **Twitter**: an on-line news and social networking service where users post and interact with messages, "tweets," restricted to 140 characters
- **Pinterest**: an internet photo sharing and publishing service that allows users to "Pin" pictures they like and upload their own recommendations to their "pinboards".
- **YouTube**: a free video sharing website that lets people upload, view, and share videos.
- **Instagram**: an online photo and video sharing social networking service. It allows users to take pictures and videos, apply digital filters to them and share them to their followers.
- **Linkedin**: a social networking website for people in professional jobs. Users can make connections with other people they have worked with, post their work experience and skills, look for jobs, and look for workers.

Step 2 : The next step is to create a function that first creates the lists with the 6 elements we would search on each html code.

Step 3 : Then after checking that the html code for each specific site has been downloaded properly (in other words is not equal to zero) we search in a loop for each of the social media at hand if it exists in the html code. If it exists we put True in the respective list.After completing this procedure we export also the time we did to run the function.

Step 4 : Now that we have the function ready we run it and afterwards we again save the results in a new data frame.
The code that was created is available in the Appendix.A.2

3.4.8 Links - Internal and External

The navigation inside a website is extremely important as we mentioned in the literature review and highly correlated to the number of hyper links that are included in it. Thus, the next step is to see how many internal, external and

finally total links each html code has. The links show in how many other pages does this home page leads to. If the links are external that means as we said in the Metrics section that they lead in a page that is not of this website. While the internal links lead to other pages in the same website.
By extracting this information we want to see if it is important for the user to have the ability to browse in various pages inside the site from the home page and also if it plays any role the use of external links that give the user the opportunity to be transferred in another site that is relative of course with the one he is looking in.

Step 1 : A great way to spot the links in an html code is the prefix that they all use which is "href". So the first step (after creating the initial lists of course and the loop we do in each script) is to locate how many times does the expression "href" appears in the code. The result of this search will give us the total links of the site.

Step 2 : Next step is to separate somehow the internal and the external ones. An easy way to do that is to find all the "href" references that are followed by "="https: ". By locating specifically those expressions we have located all the external links because unlike the internal links that do not need to be written in this form the external links since they lead to other sites they should always begin with this expression.

Step 3 : Now that we have the total links and the external links as well we can easily find the internal links with a simple subtraction:

$$\textit{total links - external links} = \textit{internal links}$$

Finally we put the results in the respective lists for each site and we export the time we did to run the script.

Step 4 : Now that the function is completed we run it and then we create a data frame as we have done in the previous scripts as well.
The code that was created is available in the Appendix.A.2

3.4.9 Loading time

One very important factor as we mentioned before3.2.1 is the time that the site does to be loaded and with this part of the code we will count exactly that.
We have already counted in previous scripts the time they did to be completed. In a similar logic we will count the time it does for the browser create after opening the url to read it.

Step 1 : The procedure is the same, firstly we create the empty list where we will save the time it does to open for each site.

Step 2 : Then we create the function and we create the loop for the 500

sites. In case of an exception we put "n/a" in the respective list position or if the site is the number 119(Arrow Electronics) or 465(St. Jude Medical)[7] which create a problem in the code and we continue. In the end of the function we appear a text that lets us know that the process has been completed.

Step 3 : Next step is to run the function and finally create a data frame with the results.
The code that was created is available in the Appendix.A.2

3.4.10 Number, type and sizes of images

We reach to the final group of variables that we are going to examine regarding the home pages of the websites of the 500 companies that ranked first in Fortune 500. This group has to do with the images of each site. We can divide the informations that we want to extract into 3 major categories:

1. Type of images
2. Number of images
3. Different image sizes

3.4.10.1 Type of images

There are many different types of images that can be used in a site. Here we will try to see between the most common types which ones are the most preferable from the sites and later on in the analysis to see if this has any correlation to their success. The type of images that we will examine are the following ones:

- **PNG**: Portable Network Graphics (PNG) is a raster graphics file format that supports lossless data compression[8].
- **BMP**: Bitmap (BMP) is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter), especially on Microsoft Windows and OS operating systems.
- **DIB**: Device-Independent Bitmap (DIB) is a graphics file format used by Windows. DIB files are bitmapped graphics that represent color formats. Similar to BMP format, except they have a different header. DIB files can be opened and edited in most image editing programs.
- **JPEG/JPG/JPE**: The term (JPEG) is an acronym for the Joint Photographic Experts Group, which created the standard method of lossy compression for digital images, particularly for those images produced by

[7] or 118/464 in the loop's numbering since it start from zero
[8] Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data

digital photography. The degree of compression can be adjusted, allowing a selectable trade-off between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.

- **GIF**: The Graphics Interchange Format (GIF) is a bitmap image format that supports up to 8 bits per pixel for each image, allowing a single image to reference its own palette of up to 256 different colors chosen from the 24-bit RGB color space. It also supports animations and allows a separate palette of up to 256 colors for each frame. These palette limitations make the GIF format less suitable for reproducing color photographs and other images with continuous color, but it is well-suited for simpler images such as graphics or logos with solid areas of color.

- **TIFF/TIF**:Tagged Image File Format (TIFF or TIF) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry and photographers. The TIFF format is widely supported by image-manipulation applications, by publishing and page layout applications, and by scanning, faxing, word processing, optical character recognition and other applications.

As we can see some of the type of images have more than one ways that can appear so in order to be precise we will examine all nine different endings. The procedure we will follow is the same one.

Step 1 : Initially we create the lists we will use then we create the function and the loop for the 500 sites.

Step 2 : For each site we check if the html code is not empty and then we search for all the endings that exists in each site and we save in the lists the number of times the ending appeared so as to know how many such images the site has.

3.4.10.2 Number of images

Step 3 : The total number of images is the sum of all the different types of images and the number of each of those in a site. So in order to catch to birds with one stone we create a variable in the same function where in each loop for each different type of image it adds them in this variable so as to have the total images in the end.

Step 4 : In the end of the function we export the time the function did to run again. The next step is to run the function and finally create the data frame with the variables we created.
The code that was created is available in the Appendix.A.2

3.4.10.3 Different image sizes

Aside from the number of images and the types that are being used another important factor is the size of each image. There is a vast majority of different sizes and it won't be wise to restrict the research to specific sizes so in the following code will find the different sizes that are being used in the companies.

Step 1: The first step is to find the different dimensions that each site uses. We initially create the lists that we will use for this first function. In the script there are also comments of what each list represents. Then we create a function that will gather all the different dimensions from all the sites and in another variable all the times each of this dimension appears.

Step 2: Next with the help of the library Beautiful soup we read the html code as lxml and we save the result into a new variable. Then we search in this new variable with the help of the same library all the elements that are between the $< img > ... < /img >$ parts of the code.

Step 3: Then we try to separate the height and the weight since in some cases there aren't available both the dimensions and in order to be more precise we will keep only the sizes that have both the dimensions available. We do that by counting in the next loop - that we will combine the two dimensions to a form like (300x300)- the length of the smallest list.

Step 4: Now that we have the combinations first we check if the list is empty, so as to put zeros in the lists of dimentions and times that they occured for the specific site and then if it is not zero we create a counter to find how many different combinations we have for this site.

Step 5: Then we the help of the split and replace packages we keep only the dimensions names and the times that they occured for each site in a list.

Step 6: Then this list should go to the list with all the sites in the position of the site at hand. In other words we create another list for which each of the list's elements are other lists. Finally we extract the time the function did to run.

Step 7: We run the function

Step 8: Next we create a new function which will help us locate only the unique sizes from all the companies. We create the initial lists that we will need and then the loop for the 500 sites. For each site we put the different dimensions in a new variable. Then we check if the elements of this new list have already been added in the list with the unique sizes and if they haven't we add them. In the end we print the results.

Step 9: We run this function as well

Step 10: We create a new function where we want to make to variables that will give True or False depending on whether or not a specific company's website has a specific dimension of the ones we gathered in the previous script. Firstly we create the initial lists and then the function and the initial loop for the 500 sites.

Step 11: Again in this function we create lists inside of lists since we have lists of different dimensions and lists of different sizes and we must combine them in one list with the results of True or False first for each site and further more for each dimension. In the end we print the duration of the function.

Step 12: We run the function

Step 13: We create an initial data frame where we will add the sizes later on.

Step 14: As we said before we have created a final list that its results is other lists so what we need to do now is to create a function in order to break this variable that we created so as to add it in a proper manner in the data frame we created. Again we calculate the time it does for the function to be completed.

Step 15: Run the last function and create the final data frame.
The codes that was created is available in the Appendix.A.2A.2A.2A.2

3.4.11 Fortune 500 - Metrics download

Just in order to have some metrics that will depict the status of the companies, we will also download some metrics from the Fortune 500 sites for each company.[35] In order to achieve that we should open the pages for each one of the sites separately. Since there is a pattern in the way the pages are named it shouldn't be difficult.

Step 1: Firstly we should create the pattern with which we will download the pages. By running the code we can see that the names of each company are not written exactly as we have saved them.

Step 2: So the second step we need to do is to alter the names that are not similar in order for the next functions to run.

Step 3: Now that we have the correct names we can download the html code of each one of the fortune 500 companies pages from the fortune 500 site so as to extract the informations we need. Again here we follow the same steps. Firstly we create an initial list and then a function and a loop for the 500 sites. And for each loop turn we download and save the html code behind the fortune 500 page for the site at hand. Finally we run the function.

Step 4: Now that we have saved the html codes we are going to extract some informations that we need from them. In order to do that initially we have to create the variables we will need.

Step 5: Now that we have the initial variable we are going to create a function in order to extract the specific parts we need. More precisely we are going to download the following informations:

- Revenues in dollars
- Revenues in percentage
- Assets in dollars
- Total Stockholder Equity in dollars
- Market Value in dollars

For each loop we use the library Beautiful soup to find all the parts between $< tbody > ... < /tbody >$ and we locate in which of the parts do the elements we want to extract lies on.

Step 6: After locating them we find in this specific part all the parts that are between $< td > ... < /td >$. Then we locate in those new parts where each of the information we need is and with the help of the regular expressions[9] we keep only the numbers of each part.

Step 7: We do that separately for each one of the five variables we want to create and finally we print that the function is completed.

Step 8: Next we run the function

Step 9: Finally we create the data frame with the elements we extracted.
The codes that was created is available in the Appendix.A.2A.2A.2A.2A.2

3.4.12 Merge data-frames and extract final csv file

The final step in order to conclude the section with the downloading and gathering of the data is to create a final data frame which will include all the information gathered and then create a csv file that will be used for the further analysis that will be take place with the use of the program language R.
In order to do that we merge the data frames we created by two using the company name as key and then we repeat the merging procedure until we end up with only one data frame that will include all the others.
Finally we create the csv file that we will use in the analysis in R in the next chapter. The code that was created is available in the Appendix.**??**

[9] A regular expression, regex or regexp is a sequence of characters that define a search pattern. Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings

4 Data Analysis

By concluding the gathering of all the information needed the next step is to proceed to their analysis. For the purposes of this assignment we will use the programming language R to perform this analysis.[9]

4.1 R Language

R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.
Furthermore R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends available. For the purpose of this paper we will use the R-Studio which is a free and open-source integrated development environment (IDE) for R.

4.2 Analysis

In this section we will present the steps that we followed to contact this analysis and also we will explain elaborately the findings. The analysis that we are going to perform will be explained by the following order as it was performed:

1. Data cleansing
2. Variable analysis and correlation
   (a) Fortune variables correlation
   (b) Social media analysis and correlation
   (c) Links analysis and correlations
   (d) Words analysis and correlation
   (e) HTML validation variables analysis and correlation
   (f) Image types analysis and correlation
3. Data manipulation
4. Logistic Regression
   (a) Training and test set creation
   (b) Null and Full model creation
   (c) Lasso Method
   (d) Both Method
   (e) Predictions and comparison of models
5. Comparisons and other methods

(a) Correlation testing

(b) Clustering testing

(c) Final testing

(d) Final model

4.2.1 Data cleansing

Before we will be able to begin the analysis we should first check the data that we have gathered.
The first step is to upload in the R environment the csv file we have created and see how many variables and observations we have.

Table 4: Data

Observations	Variables
500	730

The data have 500 observations as it was expected since we will examine 500 companies and 730 variables. The variables are far too many so in the next sections we should find a way to remove some of them that will not be useful in our research.
Another change that we have to make is to change the decimal point in the variables we gathered from the Fortune 500 site in order to be read correctly from R.
The next thing we should do is to make all the variables numeric in order to be more easy to compare them. The only variable that we keep as factor is the Readability one since we made it on purpose like that so as to have a clear view of what the flesch test results really mean.
Moreover we should omit all the variables and observations that contain nas from this analysis so as for the results to have a point.
The next step is to rename some variables in order to have a better and clearer understanding of what they represent. For example a variable named X that was added when we extracted the csv from Python represents the actual ranking of each company, an information that will prove to be very useful later on. So it will be a good idea to rename this variable to Ranking.
Some of the variables while they were important during the python scripts they do not actually play any important role in this analysis and we should remove them from the data that we are going to analyse. For example the company's url was an important variable while we ran the python scripts but since there are unique for each site they cannot give us any useful information for the analysis we want to perform.
Another important step before going on with the analysis is to upload the libraries that we are going to use further along. More specifically we will upload the following libraries for starters:

- **ggplot2**: The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots.
- **reshape2**: The reshape2 is an R package written by Hadley Wickham that makes it easy to transform data between wide and long formats.
- **DAAG**:This package is called Data Analysis and Graphics Data and Functions and it consists of data sets and functions.

Finally we see how many observations and variables we have now.

Table 5: Data after cleansing

Observations	Variables
408	708

The script we used to perform all the above steps could be found in the Appendix.A.3.1

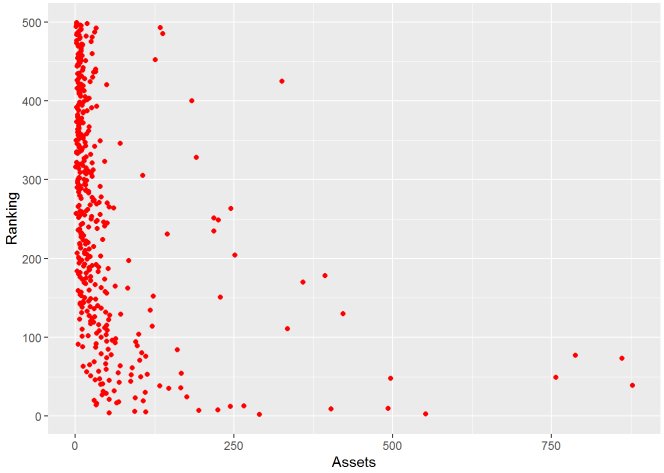
4.2.2 Variable analysis and correlation

Now that the variables are cleansed we can perform the first step of the analysis which is the analysis of the variables so as to have a first look of how they are distributed and how they are correlated to each other. In order to proceed with this analysis we have to separate the variables into same groups as we did when we downloaded them and examine them together to see if some of those prices are correlated.

4.2.2.1 Fortune variables correlation

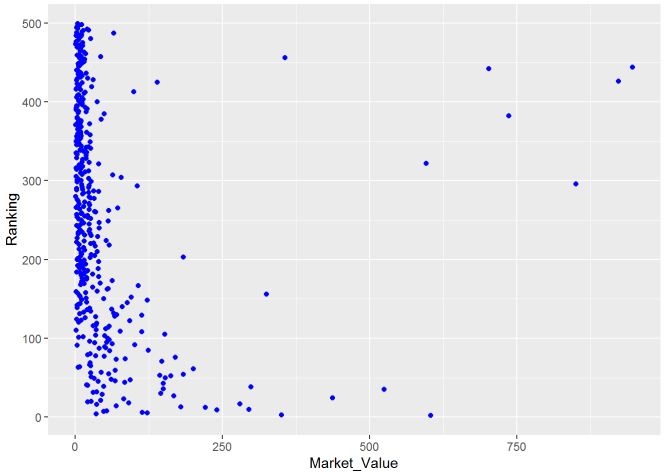
The first step is to determine the variable with which we will compare all the other ones. Obviously this variable will be one of the metrics we downloaded from the fortune 500 site so as to have an actual variable that shows the status of the company and compare the rest of the variables with it.
In order to see which variable we will use as the comparison metric we should check which of them is better correlated with the Ranking of the companies. Since the Fortune 500 magazine is ranking the sites based on the revenues we expect to see that this variable will be the most correlated with the Ranking. In the following diagrams we can see the correlation of the Ranking with each one of the Fortune 500 variables and finally with all of them:

Table 6: Assets vs Ranking



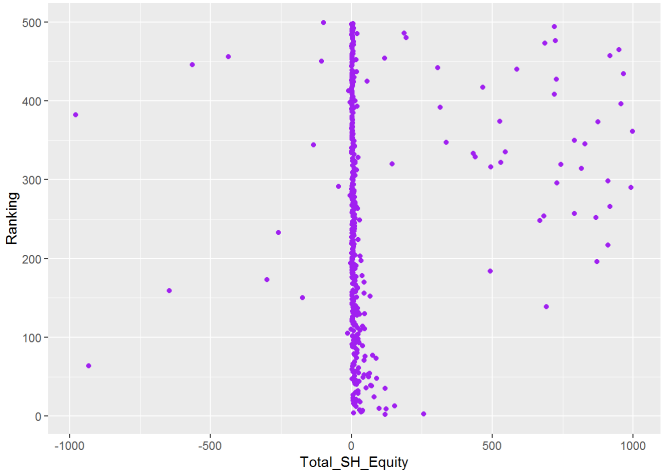
We can see that there is no specific pattern that goes along with the Ranking here. Most companies have Assets until 100 million dollars.

Table 7: Market Value vs Ranking



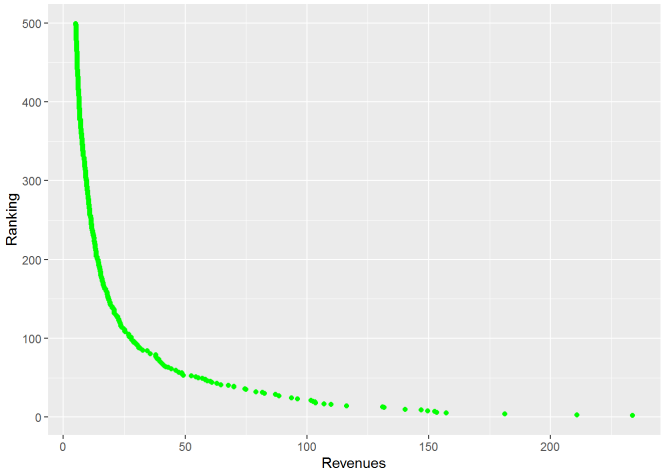
Market value even though in the most high ranking (the lowest prices of ranking) shows a small differentiation and is bigger that in the lowest ranking still there is not a clear pattern.

Table 8: Total Stockholder Equity vs Ranking



Most company total stockholder equity is until 50 million dollars but it does not change based on the Ranking.

Table 9: Revenues vs Ranking

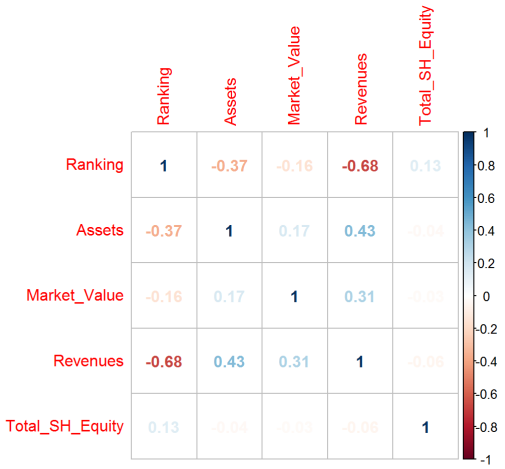


As it was expected the smaller the Ranking (closer to top1 the bigger the Revenues. We can see a linear correlation between the two variables

Based on the above diagrams we can conclude that the variable that we will use in order to see the influence of the websites variables in the company is

the Revenues. As it is clear from the last diagram the Revenues have a linear correlation with the Ranking. The better the ranking the highest the Revenues so it is the ideal variable to see the actual success of each company in relation to the variables we have created. We can double check those finding with the help of the libraries corrplot and caret. We run a correlation plot with all the variables we have seen above:

Table 10: Fortune variables correlation plot



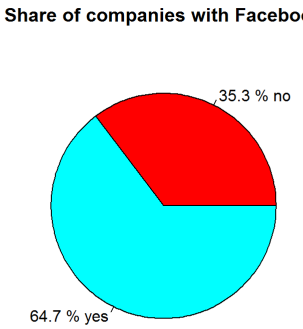
From the chart we see that the strongest negative correlation is between Ranking and Revenues as we already have seen.

Moreover the code that was used to create these diagrams is available in the Appendix.A.3.2.1

4.2.2.2 Social media analysis and correlation

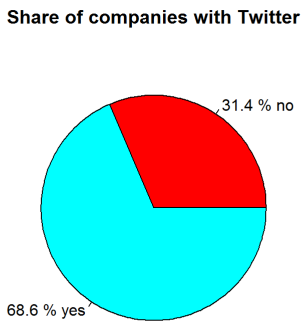
Now that we have reach to the conclusion that the metric we will use from the Fortune 500 is the Revenues we will proceed with analysing the social media variables to see how they are distributed and then to see how they are correlated with the chosen variable.

Table 11: Facebook



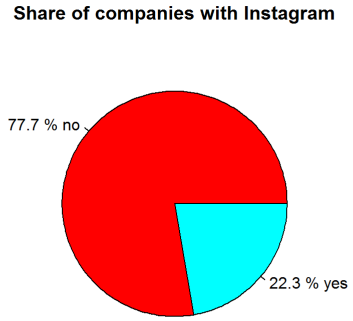
Almost the 65 percent of the companies that we are examining have Facebook links on their home page.

Table 12: Twitter



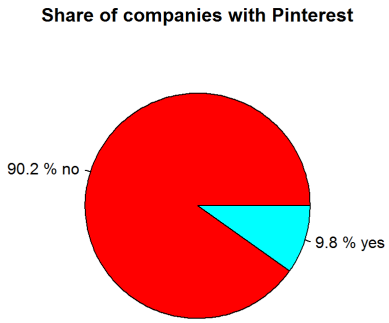
Almost the 67 percent of the companies that we are examining have Twitter links on their home page.

Table 13: Instagram



Only the 22 percent of the companies that we are examining have instagram links on their home page.

Table 14: Pinterest

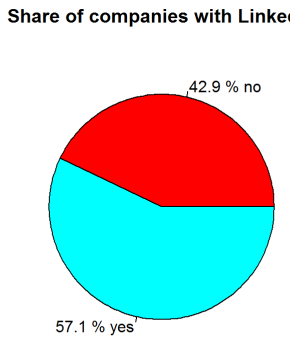


Only a small 10 percent of the companies that we are examining have pinterest links on their home page.

Table 15: YouTube

The 58 percent of the companies that we are examining have YouTube links on their home page.

Table 16: LinkedIn

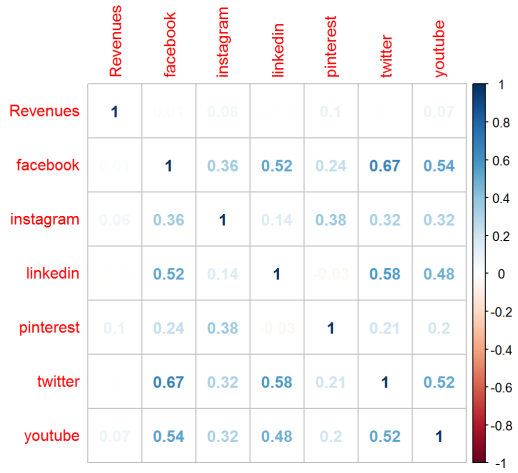


The 57 percent of the companies that we are examining have YouTube links on their home page.

From the above pie charts we can conclude that the most frequently used social media is Facebook and Twitter and the least used ones is Instagram and

Pinterest. The first conclusion can be easily confirmed since those two first social media are the most widely known. Nevertheless instagram is being used from a great majority of people and many companies use it for campaigns.
The next step is to create a correlation chart with all the social media and the Ranking variable so as to see their relation:

Table 17: Social media correlation



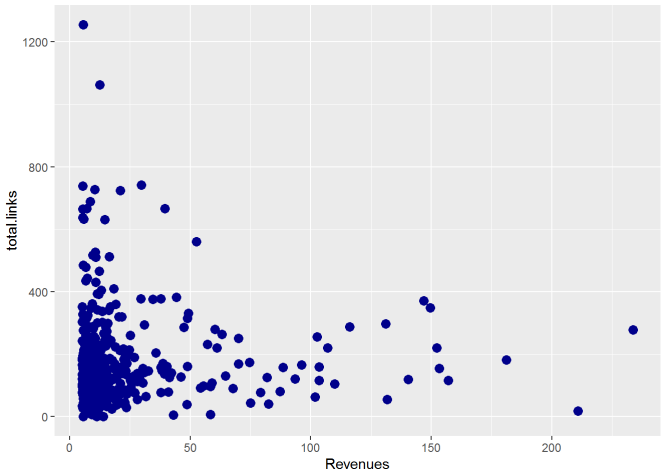
Facebook has correlation more than 50 percent with twitter, youtube and linkedin and that the smallest correlations are those of pinterest and instagram.

The same conclusions can derive from the correlation plot as well. The code we used to create those charts is available in the Appendix.A.3.2.2

4.2.2.3 Links analysis and correlations

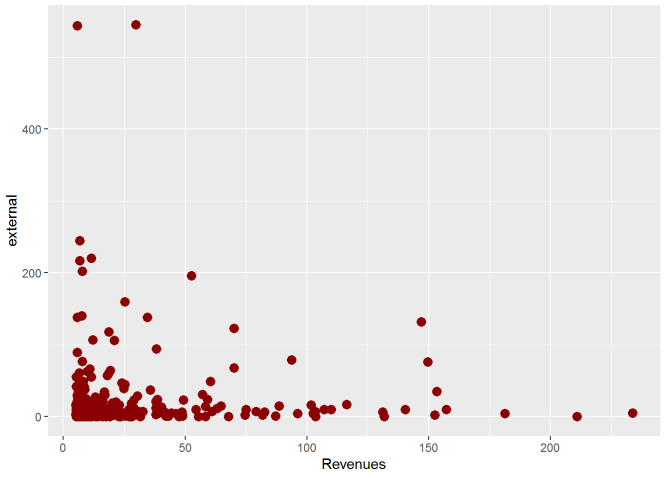
Next step is to analyse the distribution of the internal, external and total links and of course their correlation with each other and with the Revenues. We begin with the distribution of the variables in correlation with the Revenue variable:

Table 18: Total Links



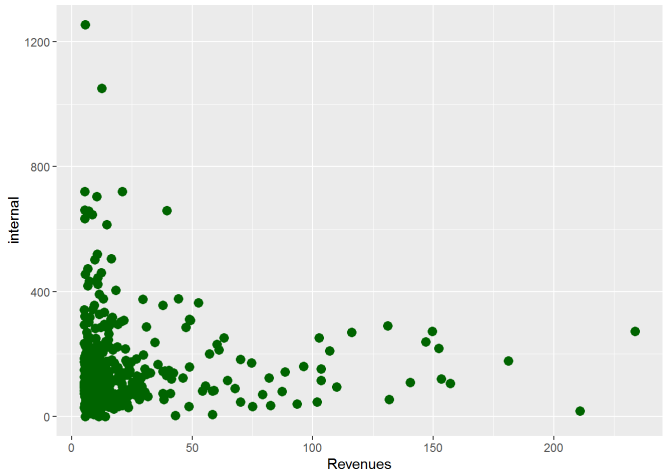
We can see that as the Revenues increase the total links are steadily going under 400.

Table 19: External links



Again as in the previous table as the Revenues increase the external links tend to go very low.

Table 20: Internal links



The internal links have a very similar distribution to the total links which is logical since the one is a subset of the other.

Now that we have the first glimpse of the variables and we understand that when the Revenues are increasing the number of the link is under some specific margins we can create also the correlation table to see if we can extract any other useful information:

Table 21: Links correlation table

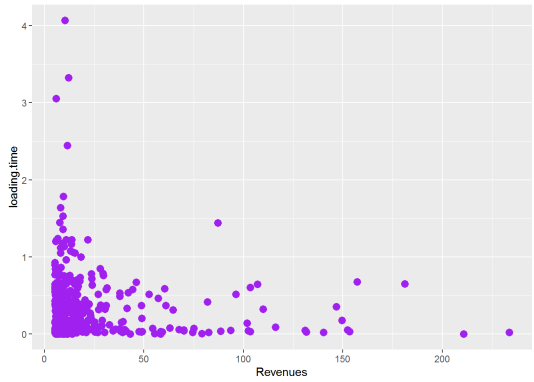
The internal links are extremely correlated with the total links.

From the correlation table we can see that the observation we made based on the previous charts, that the internal links distribution in relation to the Revenues is very similar to the one of the total links, can also be supported from this table were the correlation between the 2 variables is 95%. This information might come handy later on when we are going to create regression models. The code for this section is available in the Appendix.A.3.2.3

4.2.2.4 Loading time analysis and correlation

The loading time of the sites is a very important key performance indicator usually in website performance so it is very interesting to see if it has any strong correlation with the Revenues of these very successful companies that we are examining.:

Table 22: Loading time



Low loading time in the sites with the most Revenues.

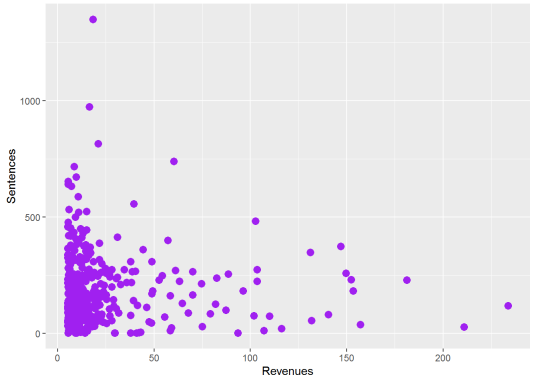
From the table above we can see that the loading time in relation to the Revenues has a small correlation. The companies that have the highest Revenues have also a very small loading time. Moreover we can see that the companies that have more than 100 million Revenues have loading times under 1 second. This observation shows that maybe the loading time is in fact a very important factor regarding the success of a website and moreover of a company overall success. The code for this table is available in the Appendix.A.3.2.4

4.2.2.5 Content analysis and correlation

The next step is to analyse the distribution of the variables that are related to the content and the text of the websites in relation to the Revenues of the companies at hand. As we explained in the Python chapter3.2.5 we gathered 3 variables that are related to the grammatical overview of the text that is being used in the sites - which are the numbers of words, the number of unique

words and the number of sentences - and 2 variables that are trying to explain if the content of this text is comprehensible from the majority of the users or not. Following below we have the charts with the distribution of those variables prices across the sites under examination and their relation to the Revenues.

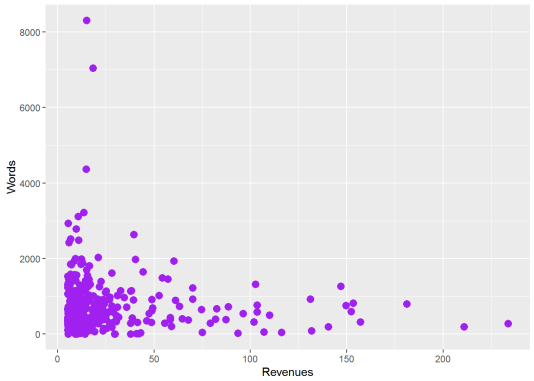
Table 23: Number of Sentences vs Revenues



In high Revenues the number of sentences is smaller that 500.

From the table we can see that most sites have less than 500 sentences in their home page. Moreover while the Revenues price is going up the number of sentences seems to getting lower. For the sites that have more than 100 million dollars in Revenues the number of sentences drops under 250. Half the number of sentences that we can see in sites with lower Revenues.

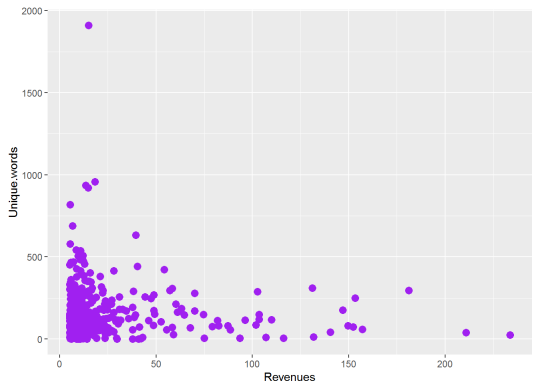
Table 24: Number of Words vs Revenues



In high Revenues the number of words is smaller that 1000.

By the same logic as the sentences in relation to the revenues so does the number of words seems to drop when the revenues go up. Furthermore here we can see that the decrease starts in lower revenues prices (around the 50 million dollars). This is expected if we take under consideration that the number of words is always correlated to the number of sentences. Most sites do not want to have very long sentences in order not to bore the reader. So since the number of sentences incline when the revenues are increasing we anticipate that the number of words will have the same reaction to the revenues.

Table 25: Number of Unique Words vs Revenues



In high Revenues the number of unique words is close to zero.

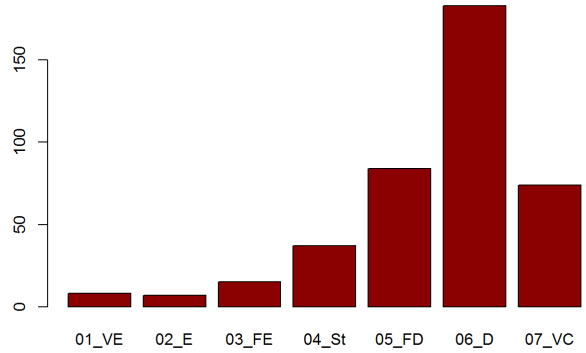
The number of unique words differs from the actual number of all the words. Here we are trying to see how many not commonly used words does each site uses. Here the distribution is not so clear as it was for the two previous variables but still we can see that in the most sites that have high Revenues the number of unique words are very small.

Table 26: Correlation table

Sentences, words and unique words are highly correlated.

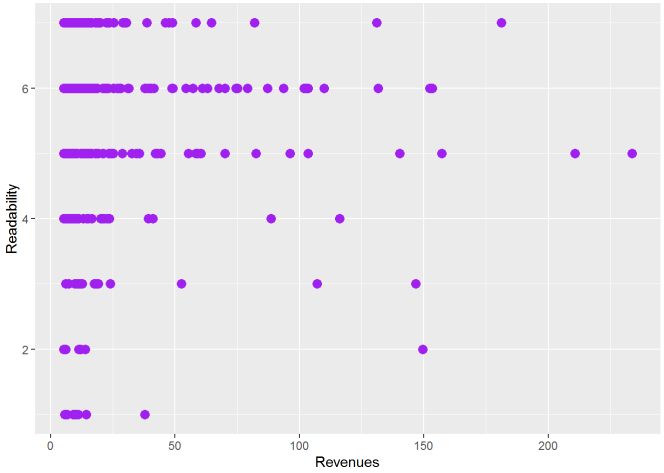
From the correlation table we can also observe that the three variables we examined regarding the text of the websites are highly correlated with each other which as we already explained is an expected outcome.Moreover we can see that the Revenues with the loading time has a small negative correlation. This means that when the one variable price is going up the others' variable price should go down. This is also an expected outcome since as we said before the time a site does to open is very important to the users as they can easily grow impatient and look somewhere else. This is due to the grand variety of choices available on-line that makes the user feeling powerful.
Now that we analysed the 3 variables regarding the text attributes we should also analyse the 2 variables regarding the readability index of the websites' texts. Since of course the flesch measure variable was the source for calculating the readability variable we will analyse here only the readability variable thoroughly while the diagrammatically analysis of the flesh measure variable will be available in the Appendix.A.3.2.5

Table 27: Readability distribution table



Most sites are difficult to read.

Table 28: Readability vs Revenues table



Most sites are difficult to read.

From the above two charts we can see that firstly the most sites are difficult to be read and secondly that the readability index in relation to the Revenues does not show a clear pattern. The two sites with the most Revenues seems to be fairly difficult to read. What we can definitely see is that the sites that

are very easy to be read all have revenues under 50 million dollars. This could mean that even though the users do not want to read very difficult texts they also do not want extremely easy ones. Maybe the correlation plot could help us have a clearer view of the relationship between the 2 variables.

Table 29: Correlation table

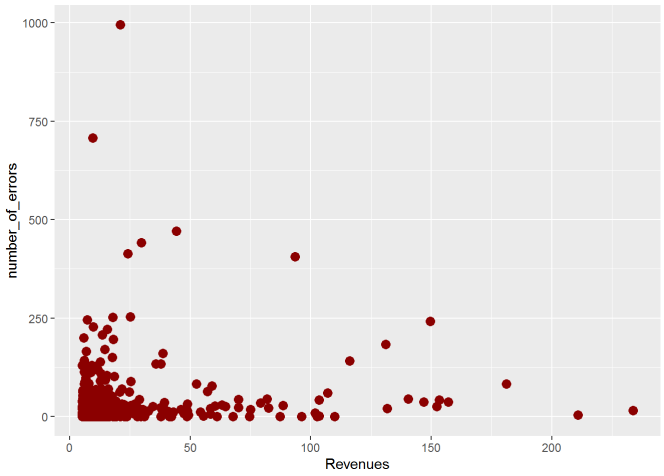
Flesh measure and readability variables are negative correlated.

From the correlation plot we can see that the Revenues have a very small negative correlation with the readability variable. While the readability variable and the flesh measure variable as it was expected they have a negative correlation. This means that the bigger the price in the flesch measure the lower the price in the readability variable. If we consider that the best prices of the flesh measure are the highest while respectively for the readability variable are the lowest then it is a normal correlation.

4.2.2.6 HTML validation variables analysis and correlation

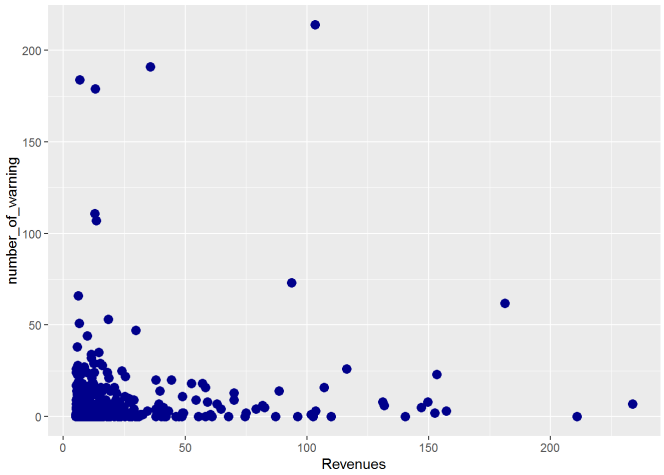
During the extraction of the html validation variables we created 4 different metrics: the number of errors, the number of warnings the non document error and the page not opened one. In this section we are going to analyse the distributions of those variables and also their correlation with the Revenues.

Table 30: Number of errors vs Revenues table



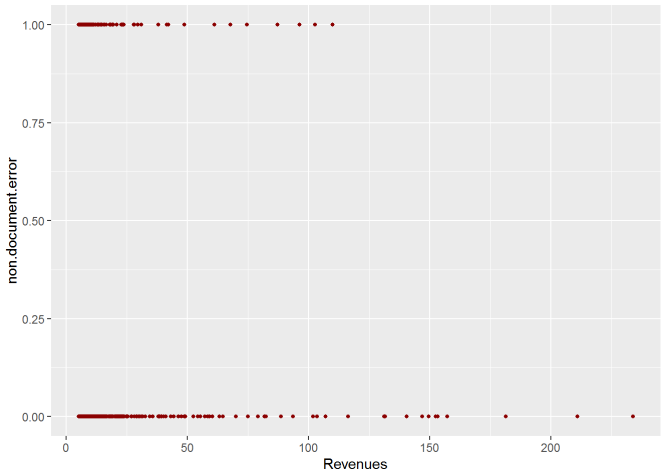
Number of errors in highest revenues are small.

Table 31: Number of warnings vs Revenues table



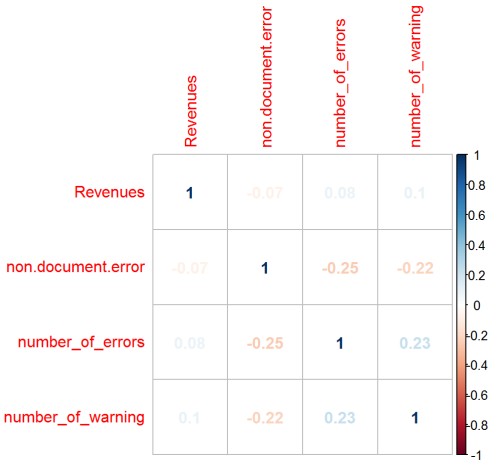
Number of warnings in highest revenues are relatively small.

Table 32: Non document error vs Revenues table



In highest revenues there is no document error.

Table 33: Correlation table



Number of errors positive correlated with number of warnings.

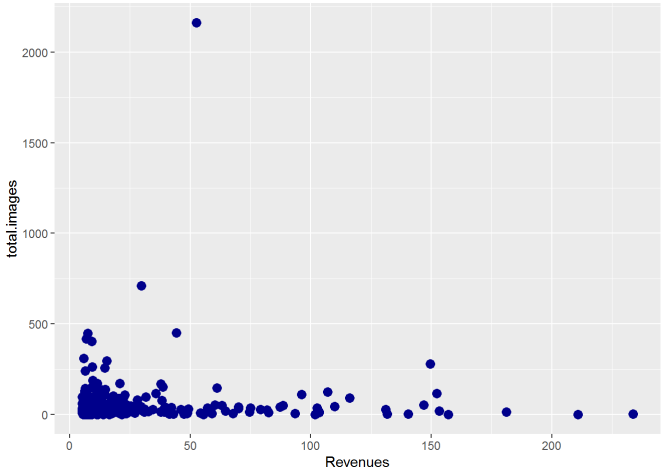
The table of the document opened variable which is available in the AppendixA.3.2.6 is not included since it has the same price for all the sites thus it has no meaning examine its impact to the revenues. Regarding the four tables above we can see that the number of errors and the number of warning in the

majority of the websites are getting smaller while the revenues are getting bigger - although a small number of sites doesn't seem to follow that logic and this can be seen from the correlation table as well that shows a very small (next to zero) positive relation between the pair of variables. Moreover the variable non document error has a negative correlation to the revenues since as the revenues increase the variable tends to become zero. This means that in sites with high revenues the documents of their sites had no errors.

4.2.2.7 Image types analysis and correlation

Last but not least we should analyse the distribution of the different types of images that are being used on the websites we are examining and also compare the total images with the Revenues to see their correlation.

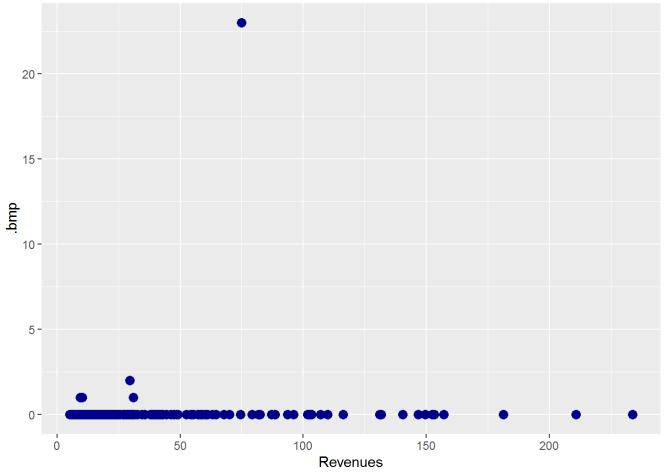
Table 34: Total images vs Revenue table



Most sites under 250 images.

From the above chart we can see that the most sites have less than 250 images. Although there is one site that has more than 2000 images. Moreover we can see that the sites with more than 100 million dollars revenues have even less than 175 images one their home page. Now that we have a first glance of the total images in the sites we should examine the nine different types of images that we have gathered.

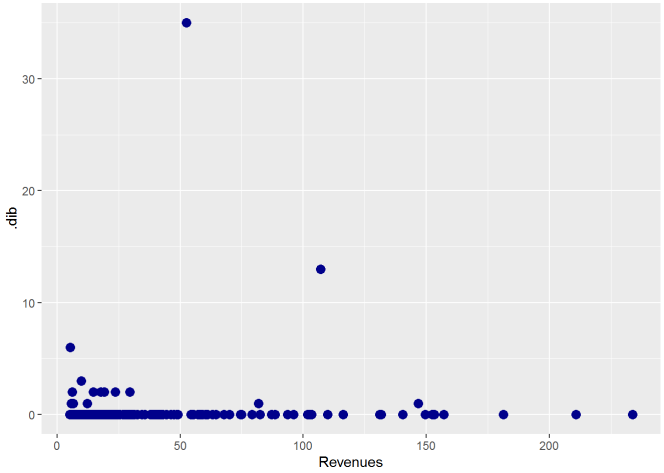
Table 35: BMP vs Revenue table



Most sites do not have bmp images.

From this chart we can see that only 4 sites actually uses bmp images.

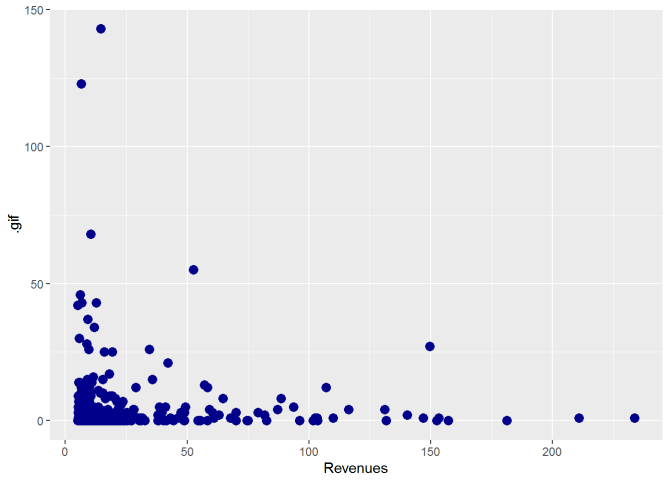
Table 36: DIB vs Revenue table



Only used by sites with small revenues.

The .dib type is being used from more sites but all of them are making less than 100 million dollars.

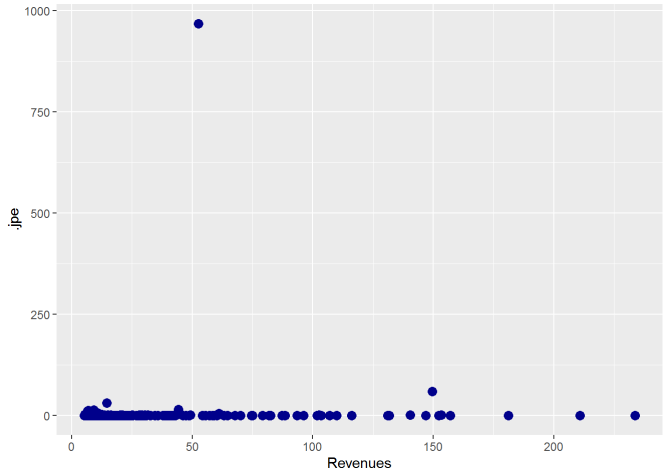
Table 37: GIF vs Revenue table



Used a lot in companies with the smallest revenues.

The .gif type is being used from many sites but in the highest revenues the number of gif images are being reduced to very small numbers.

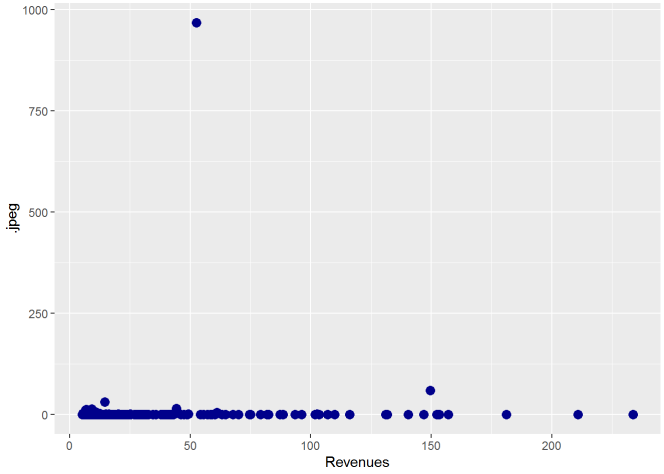
Table 38: JPE vs Revenue table



Very small number of companies use this type of images.

The jpe images is not being used from many sites and the ones that are using them are under 100 million dollars of revenues

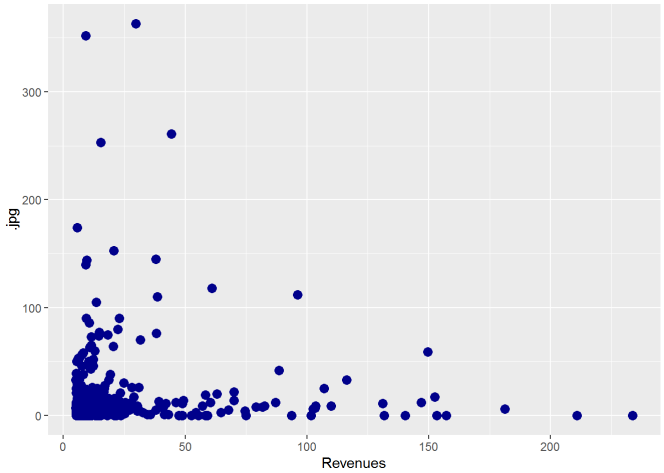
Table 39: JPEG vs Revenue table



Very small number of companies use this type of images.

Again this type is being used by very few companies sites and all of them have less than 100 million dollars revenues.

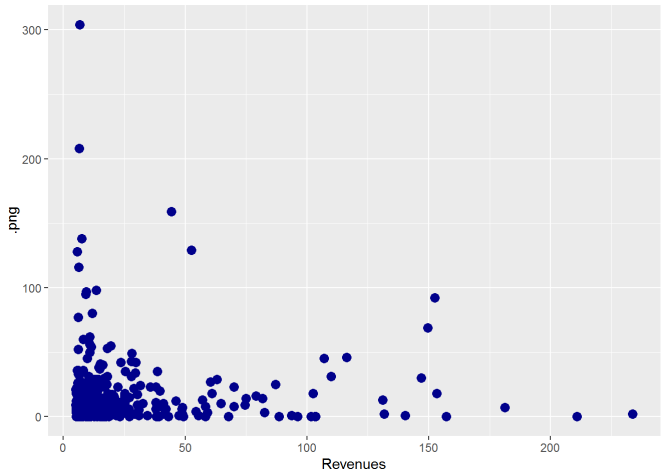
Table 40: JPG vs Revenue table



Frequent use in websites.

The jpg type of image is widely used in almost every website. We can see that when the revenues decrease the number of jpg images is increasing.

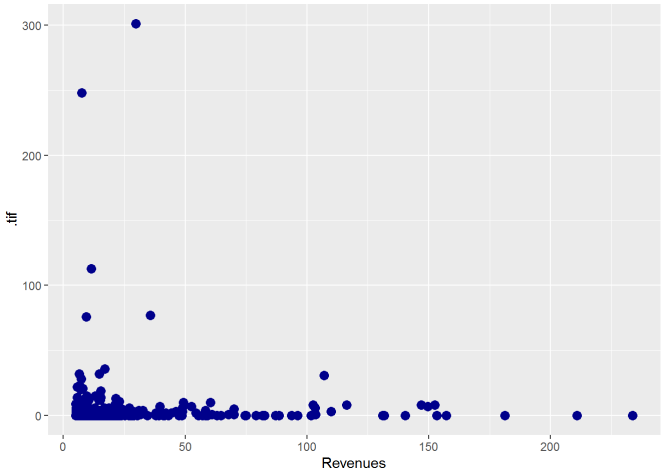
Table 41: PNG vs Revenue table



Frequent use in websites.

Also the .png type is being used from most companies. Here we can see that companies with more than 100 million dollars use a large amount of png photos in their home page.

Table 42: TIF vs Revenue table

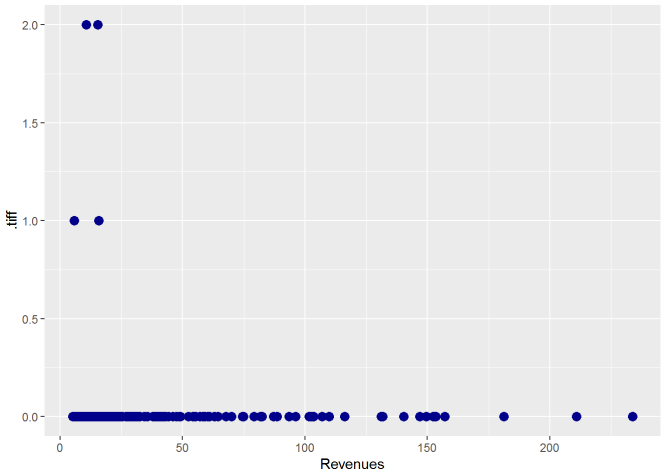


Used by low revenue sites.

The tif type even though it doesn't seem to be as widely used as the png one

it appears to be used from many sites but most of them belong to companies with less than 150 million dollars.

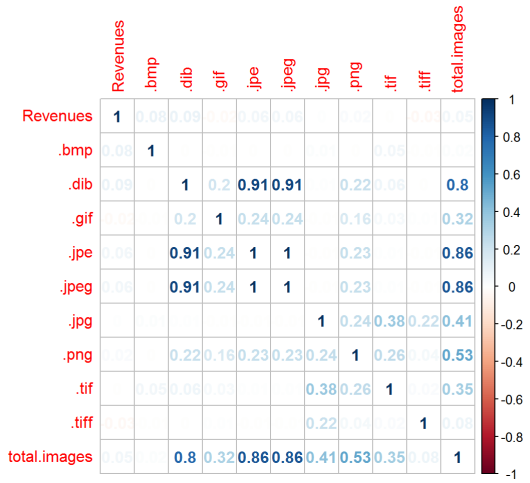
Table 43: TIFF vs Revenue table



Not used often in websites.

The .tiff type is used only in four sites and in very low quantity of images. while the companies that are using this type have less than 25($M)revenues.

Table 44: Correlation table



Number of errors positive correlated with number of warnings.

From the correlation table we can see that the revenues are positively related to every type of images except from the .tif and the .gif. A hypothesis that if we take into account that the largest number of those type of photos occurred in companies with small revenues can be easily supported.
Furthermore it seems to be a very high correlation between three types of images the .dib the .jpe and the .jpeg. Also the total images is highly correlated with all the types which is logical since the total images is an outcome of the sum of all the occurrences of all the types of images at hand. In addition based on all the previous tables of this section we can conclude that from all the types of images that were examined the most commonly used types are the .png and the .jpg.

4.2.3 Data manipulation

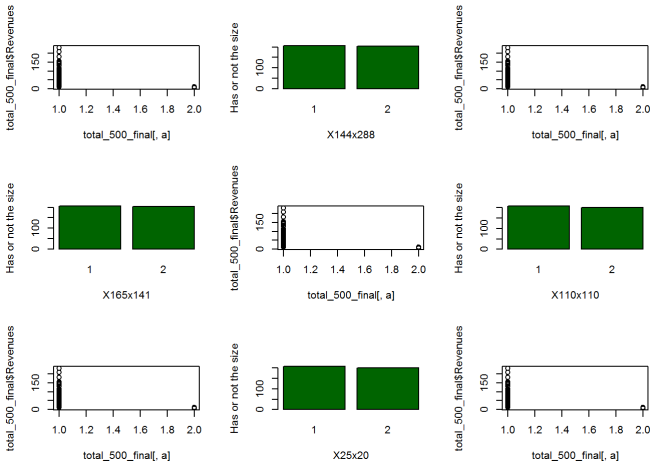
Even though we concluded the analysis part of the variables we still haven't examined the different image sizes we have gathered from the python scripts. Due to the very large amount of different sizes that we were able to gather we should find a way to reduce those variables in order to make it possible to run regression models. In this chapter we are going to examine those variables that we have created in order to see which ones are more meaningful to take further along on the analysis.

4.2.3.1 Image sizes variables reconstruction

After implementing the python code for the size images 3.4.10.3 we created almost 700 different variables. This number is almost prohibited when it comes to regression models especially when the actual number of the examined cases are a lot fewer (500 that became 408 after the cleansing that we performed).
The first step in order to examine those 700 different variables is by grouping them into two teams. In the first team we are keeping the ones that more of the half of the sites do not use and in the other one the ones that most of the sites are using. For example if we have a variable of the size 300x300 and from the 408 sites that we kept after the cleansing the 250 are using this size then the variable will be pointed to the second group.
After separating the two groups we visualized the results for each variable. In the tables that follows we can see the distribution of the variables and also the correlation with the revenues.[10]

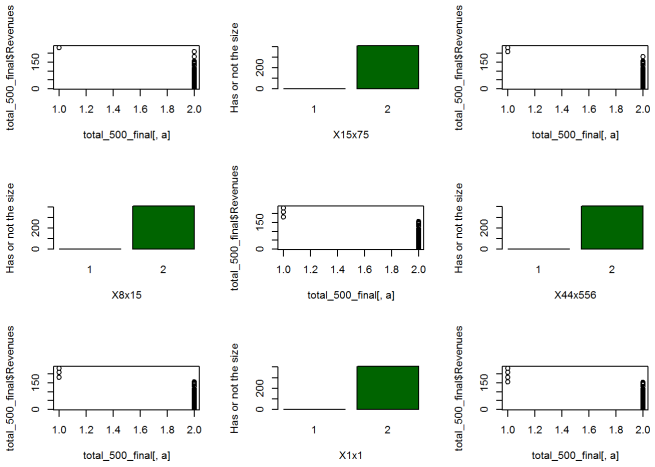
[10]Note: In the following tables we have only four examples from each group due to the vast number of variables. The codes is available in the AppendixA.3.3

Table 45: Sizes that are not used from the majority of the websites



The images are used from low revenue companies.

Table 46: Sizes that are used from the majority of the websites



The images are not used from high revenue companies.

From the two tables we can see that for the sizes that are not used from the majority of the companies there is not a specific pattern that can help us further along in the analysis. While in the table for the sizes that are being used from the majority of the companies we can observe a very interesting pattern.

Even though most of the companies are using those sizes, the ones that do not use them are those with the highest revenues. This could indicate a negative correlation between those sizes and the revenues of a company.
Moreover even though it is not quite clear in the example table above as we proceed in other variables the number of high revenue companies that are not using the sizes increase. So in order to restrict the sizes in a number of variables that can be manipulated and retrieve some useful information we decide to keep the first 24 sites of the second team (the one that contains the sizes that occur in more than half of the sites). Those sizes are the following ones:

Table 47: Sizes that will be examined

15x75	8x15	44x556
1x1	800x1200	100x100[11]
24pxx133px	21pxx173px	46x214
49x49	50x45	400x300
292pxx292px	200pxx200px	1279pxx984px
300pxx1500px	29x29	115x223
160x233	300x993	41x192
28x221	15x12	60x60

4.2.3.2 Remove variables

Now that we have created and analysed all the needed variables we can subtract from the data frame the variables that we have decide not to include in the regression analysis that will follow. Those are all the image sizes except from the 24 that were mentioned in the above section as well as the following ones:

1. Market Value
2. Assets
3. Ranking
4. Total Stockholder Equity
5. The page not opened variable

Those variables are being removed for the following reasons. The first four variables are data from the fortune 500 pages of the companies and we downloaded in order to determine which of those metrics we will use in order to see if the performance indicators of the websites effect the company's status. Since we have already reach to the conclusion that the variable we are going to use for this purpose is the Revenues there is no need keeping the other ones as well.
Regarding the fifth variable, the reason we will remove it is that the prices of the variable do not differ in any site. For all the sites they are the same so there is no point in further examining it.

4.2.4 Regression models

By completing the variable analysis we have ended up with 52 variables that are key performance indicators for the websites of the companies we are examining and 1 variable that shows the status of the companies. The names of the variables are available in the following table:

Table 48: Final Variables for Regression

1. Revenues	2. Non document error	3. Number of errors
4. Number of warnings	5. Facebook	6. Instagram
7. Linkedin	8. Pinterest	9. Twitter
10. Youtube	11. Flesh Measure	12. Readability
13. Sentences	14. Unique words	15. Words
16. external	17. internal	18. total links
19. 15x75	20. 8x15	21. 44x556
22. 1x1	23. 800x1200	24. 100x100[12]
25. 24pxx133px	26. 21pxx173px	27. 46x214
28. 49x49	29. 50x45	30. 400x300
31. 292pxx292px	32. 200pxx200px	33. 1279pxx984px
34. 300pxx1500px	35. 29x29	36. 115x223
37. 160x233	38. 300x993	39. 41x192
40. 28x221	41. 15x12	42. 60x60
43. bmp	44. dib	45. gif
46. jpe	47. jpeg	48. jpg
49. png	50. tif	51. tiff
52. total images	53. loading time	

As we can see from the above table the final variables are from almost every group of information that we manage to gather. There are variables concerning the html validation, the social media existence, the image types and sizes, the number of links, the content in terms of text and readability and the loading time of the websites.
In this chapter we will perform some regression models to the chosen data frame in order to determine the variables that effect most the price of the revenues of a company.The code that was created for the regression models is available in the AppendixA.3.4.
In statistics there are two main types of regressions the linear and the logistic regression. The linear regression is an approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.
The logistic regression, or logit regression, is a regression model where the dependent variable (DV) is categorical.Cases where the dependent variable has

more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. Since the variable Revenues which is the dependent variable we have chosen is a scalar numeric variable we will perform linear regression to the final data frame we have created.

4.2.4.1 Training and test set creation

Before we begin with the regression we have to divide the data frame to 2 data frames so as to be able to test the model we will create. We will make a training set and a test set:

- **Training set:** We will use this set in order for the model to learn how to fit the parameters of the classifier.
- **Test set:** This set will be used only to assess the performance of the models that we will create.

The sets will be separated in proportions: 15% -85% where the 15% will be the test set in which we will test our findings and the 85% will be the set we will use in order to create the regression models. After separating the two sets we ended up with 348 observations in the training set and 60 observations in the test set.

4.2.4.2 Null and Full model creation

The first step is by using the training set to create a null regression model and a full regression model.
The null regression model contains only the dependent variable y where in our case is the Revenues. The null model we created is the one in the following table:

Table 49: Null regression model

```
Call:
lm(formula = Revenues ~ 1, data = total_500_final_train)

Residuals:
   Min     1Q Median     3Q    Max
-17.16 -15.24 -11.17  -1.43 211.43

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   22.288      1.703   13.09   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 31.77 on 347 degrees of freedom
```

From the null regression model we can see that the intercept is positive. That means that even without any other variable the Revenues of a company start at 22.25 million dollars. This is normal if we consider that we are examining the most successful enterprises in the United States and even the ones that have low revenues in this examination do not seize to be considered high enough in comparison with other companies that are not included in the fortune 500 most successful ones.
The next step is to create a full model where we used the dependent variable Revenues against all 52 independent variables of the final data frame. The full model we created is available in the following table. From this model we can see that the intercept initial value has been increased a lot in comparison to the null model. It went from 22 to 438. Nevertheless it is obvious that some of the independent variables have negative values which means that based on the prices they get they can lower the final Revenues of the company.
Moreover we can see that some variables have NAs in the estimate. This could be happening because they are highly correlated to other variables. That is why we begin with the full model so as to see which variables would be wise to keep or to remove.
In the bottom of the regression model there are some prices that will help us evaluate the model.

- **Residual Standard Error**: is a measure of the quality of a linear regression fit. Theoretically, every linear model is assumed to contain an error term E. Due to the presence of this error term, we are not capable of perfectly predicting our response variable (Revenues) from the predictors (independent variables). The Residual Standard Error is the average amount that the dependent variable will deviate from the true regression line. In our example, the actual revenues of a company can deviate from the true regression line by approximately 14.37 million dollars, on average. In other words, given that the mean revenues for all companies is 438.83 million dollars (the intercept estimate of the full model) and that the Residual Standard Error is 14.38, we can say that the percentage error is (any prediction would still be off by) 3.27%. Its also worth noting that the Residual Standard Error was calculated with 309 degrees of freedom. Simplistically, degrees of freedom are the number of data points that went into the estimation of the parameters used after taking into account these parameters (restriction). In our case, we had 348 data points and 39 parameters (intercept and variables that do not have NA in the estimate column)so the degrees of freedom are 348 - 39 = 309.

- **Multiple R-squared:** is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.The definition of R-squared is the percentage of the response variable variation that is explained by a linear model:

$$R\text{-}squared = Explained\ variation\ /\ Total\ variation$$

R-squared is always between 0 and 100%:

  - 0% indicates that the model explains none of the variability of the response data around its mean.
  - 100% indicates that the model explains all the variability of the response data around its mean.

  In general, the higher the R-squared, the better the model fits your data. However, sometimes the R squared can be misleading that is why we should also take into account the next measure.

- **The adjusted R-squared**: is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance. As in the R squared so for the Adjusted R Squared the bigger the price the better the result.

- **The p-value**:is the probability that, using a given statistical model, the statistical summary (such as the sample mean difference between two compared groups) would be the same as or more extreme than the actual observed results.In other words it shows whether or not having this model could give better results from not having it at all and depend only to chance. If the price of p - value is lower than 0.05 the model could be characterized as significant or in other word the model fits the data well.

The four scenarios that we can meet regarding the relation between the Adjusted R square and the p value are the following:

1. low Adjusted R square and low p-value (p-value $<= 0.05$)
2. low Adjusted R square and high p-value (p-value > 0.05)
3. high Adjusted R square and low p-value
4. high Adjusted R square and high p-value

This scenarios are respectively interpreted as follows:

1. The model doesn't explain much of variation of the data but it is significant (better than not having a model)
2. The model doesn't explain much of variation of the data and it is not significant (worst scenario)
3. The model explains a lot of variation within the data and is significant (best scenario)
4. The model explains a lot of variation within the data but is not significant (model is worthless)

In our case the model has an Adjusted R Square of 79% and a p value < 0.05. This means that we are in the best case scenario.

Table 50: Full regression model

```
##
## Call:
## lm(formula = Revenues ~ ., data = total_500_final_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.199  -8.486  -3.461   2.223  65.721
##
## Coefficients: (14 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         438.837989  29.754450  14.749  < 2e-16 ***
## non.document.error   -0.973835   1.984634  -0.491 0.623996
## number_of_errors      0.012911   0.010948   1.179 0.239167
## number_of_warning    -0.020989   0.038979  -0.538 0.590653
## facebook             -3.004443   2.410328  -1.246 0.213530
## instagram             2.605986   2.234726   1.166 0.244460
## linkedin             -0.325461   2.191702  -0.148 0.882047
## pinterest             1.109936   3.221429   0.345 0.730669
## twitter               1.567012   2.451651   0.639 0.523189
## youtube               2.272481   2.072409   1.097 0.273696
## Flesh_Mesaure         0.001529   0.004208   0.363 0.716631
## Readability           1.613550   0.770141   2.095 0.036973 *
## Sentences            -0.006214   0.010664  -0.583 0.560540
## Unique.words         -0.005883   0.019639  -0.300 0.764725
## Words                 0.002363   0.004897   0.482 0.629833
## external              0.004634   0.017992   0.258 0.796924
## internal              0.001545   0.007942   0.195 0.845906
## total.links                NA         NA      NA       NA
## X15x75              -20.966921  20.586365  -1.018 0.309244
## X8x15               -30.814241  20.765759  -1.484 0.138856
## X44x556             -17.975428  20.815629  -0.864 0.388503
## X1x1                       NA         NA      NA       NA
## X800x1200            -9.979089  18.025681  -0.554 0.580250
## autox100.            -3.048954  18.426054  -0.165 0.868682
## X24pxx133px         -15.566911  18.548403  -0.839 0.401973
## X21pxx173px                NA         NA      NA       NA
## X46x214                    NA         NA      NA       NA
## X49x49                     NA         NA      NA       NA
## X50x45               -6.206753  18.046674  -0.344 0.731134
## X400x300            -12.942588  18.222831  -0.710 0.478091
## X292pxx292px         -6.790009  14.896169  -0.456 0.648838
## X200pxx200px               NA         NA      NA       NA
## X1279pxx984px              NA         NA      NA       NA
## X300pxx1500px              NA         NA      NA       NA
## X29x29               -7.177375  14.533766  -0.494 0.621769
## X115x223             -8.312448  18.150382  -0.458 0.647291
## X160x233                   NA         NA      NA       NA
## X300x993                   NA         NA      NA       NA
## X41x192                    NA         NA      NA       NA
## X28x221                    NA         NA      NA       NA
## X15x12                     NA         NA      NA       NA
## X60x60              -75.505052  14.921561  -5.060  7.2e-07 ***
## .bmp                  2.398250   0.633335   3.787 0.000183 ***
## .dib                  0.316324   1.107796   0.286 0.775419
## .gif                 -0.064718   0.066521  -0.973 0.331361
## .jpe                 -0.610764   4.148684  -0.147 0.883055
## .jpeg                 0.640994   4.149986   0.154 0.877350
## .jpg                  0.007073   0.024010   0.295 0.768499
## .png                 -0.011736   0.031687  -0.370 0.711358
## .tif                 -0.009165   0.042009  -0.218 0.827451
## .tiff                -3.166227   5.182089  -0.611 0.541652
## total.images               NA         NA      NA       NA
## loading.time         -5.750583   2.043199  -2.814 0.005199 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.37 on 309 degrees of freedom
## Multiple R-squared:  0.8179, Adjusted R-squared:  0.7955
## F-statistic: 36.52 on 38 and 309 DF,  p-value: < 2.2e-16
```

4.2.4.3 Lasso Method

Even though the results of the full model indicates that the model is quite good we shouldn't be contained with these results. Since this is the full model we could also try some other methods in order to see if we can make it even better. The first method we are going to implement is the Lasso (least absolute shrinkage and selection operator). The Lasso method is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpret-ability of the statistical model it produces.
In the regression model, we were looking to find the that minimizes the following relationship:

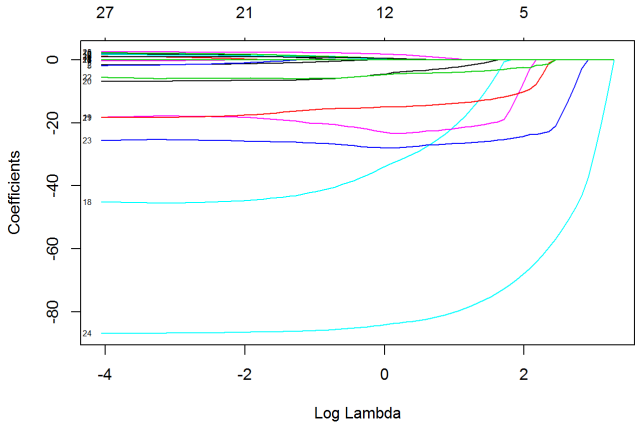
$$(Y\text{-}X1\beta1\text{-}X2\beta2...)^2$$

While LASSO applies a penalty term to the problem:

$$(Y\text{-}X1\beta1\text{-}X2\beta2...)^2 + \alpha\Sigma i|\beta i|$$

So when α is zero, there is no penalization, and you have the ordinary least squares (OLS[13]) solution. As the penalization α increases, $\Sigma|\beta i|$ is pulled towards zero, with the less important parameters being pulled to zero earlier. At some level of α, all the i have been pulled to zero.
So in order to choose the variables we are going to include in the second model we would advice the next graph. Where we can see when a variable has been pulled to zero. Of course firstly we remove the variables that were NA in the full model so as to focus on the ones that play a part on the outcome.

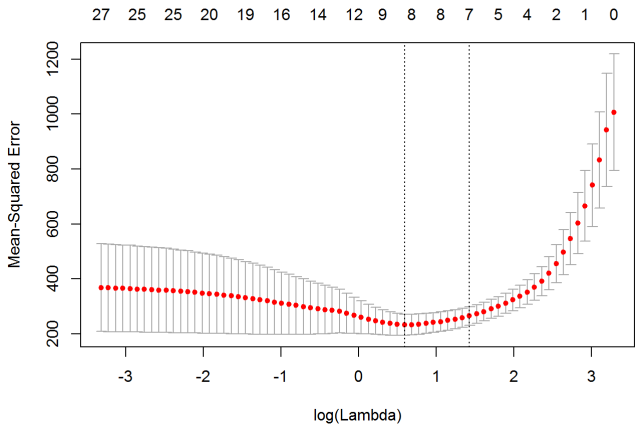
Table 51: Lasso variables



[13](OLS is a method for estimating the unknown parameters in a linear regression model, with the goal of minimizing the sum of the squares of the differences between the observed responses (values of the variable being predicted) in the given dataset and those predicted by a linear function of a set of explanatory variables.

From the graph we can see that there are some variables that make the make the longest to be pulled to zero and those are the variables that we should keep for this model. Another graph that we should take into account before choosing the variables for the model is the following one:

Table 52: Lasso Mean square error



Includes the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the λ sequence - error bars. Two selected λs are indicated by the vertical dotted lines.

The two selected λs that are indicated by the vertical dotted lines are the following:

- **lambda.min** is the value of λ that gives minimum mean cross-validated error
- **lambda.1se** is the value of λ which gives the most regularized model such that error is within one standard error of the minimum.

We can use either one in order to see which variables we can keep. We used both of them and we created 2 models. The analytical models are available in the Appendix but you can see the actual variables and their estimates below as well as the measures that indicate how well performed are the models.

- **Lasso Min model**:
  Revenues = 222.268 - 41.027 (8x15) - 24.134 (44x556) - 5.37 (800x1200)- 15.7302 (24px x 133px) - 4.8865 (50x45) - 29.5016 (400x300) - 85.9726 (60x60) + 2.5989 (bmp) + 0.9183 (dib)

- Residuals standard error : 14.28 on 338 degrees of freedom (any prediction will be off by 3.33%)
- Adjusted R-squared : 0.7979 (79,79%)
- p-value < 0.05

- **Lasso One standard deviation model**:
  Revenues = 222.268 - 41.027 (8x15) - 24.134 (44x556) - 5.372 (800x1200)- 15.730 (24px x 133px) - 4.887 (50x45) - 28 (400x300) - 87.077 (60x60)
  - Residuals standard error : 14.73 on 340 degrees of freedom (any prediction will be off by 3.4%)
  - Adjusted R-squared : 0.7851 (78,51%)
  - p-value < 0.05

From those two models we can see that the one for the min λ is better regarding the Adjusted R-squared and also the prediction accuracy is a little higher from the full model that we have already created. This means that we were able to improve the model a little.

4.2.4.4 Both Method

Nevertheless we still can check if the model can be improved even more. This time we will apply the both method to the full model (after we subtract the variables that were NAs) in order to see if the variables that will be kept give the same or better results than the models we have already created.
This method takes as inputs the null model and the full model and it goes back and forth and checks which combinations of the variables brings the best results. The model that was created after applying the method is the following one:

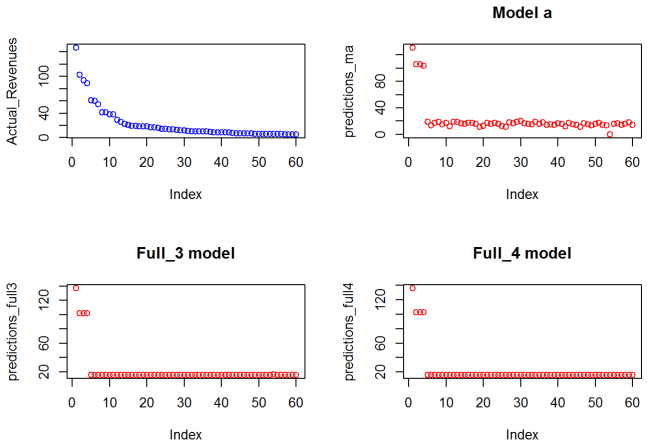
- **Both method model**:
  Revenues = 216.04 - 88.85 (60x60) - 25.93 (44x556) - 43.04 (8x15)- 21.05 (24px x 133px) - 27.14 (400x300) + 2.53 (bmp) - 5.41 (loading time) + 0.039 (jpeg)+ 1.257 (readability) + 2.956 (instagram)
  - Residuals standard error : 14 on 337 degrees of freedom (any prediction will be off by 3.31%)
  - Adjusted R-squared : 0.8059 (80,59%)
  - p-value < 0.05

As it is easily observed the model is better than the one we found with the Lasso method and as a result even better from the full one we created in the begging.

4.2.4.5 Predictions and comparison of models

Now that we have created three models we should check in actual data which of them works better. Based on the measures, we have already seen that the model of the both method has better perspectives to be the most accurate one. Now we will try to predict the revenue prices of the test set by looking only at the independent values and by implementing the models. Then we will compare those results with the actual revenues to see how well did the models perform.In the following table we can see the results of the predictions:

Table 53: Lasso variables



From the above table we can see that the highest price of the actual revenues in close to 100. The only model that goes close but do not supersedes this number is the one from the both method (model a). The other two while they are doing quite well predictions we can see that in the high revenue sites they reach the 120 million dollars which are bigger than the actual ones and also in the low prices they do not differentiate at all all the predictions are close to zero, unlike the model of the method both where the predictions, while they seem to differ a little from the actuals they seem to have a better capture of the correct revenues.
As a result we can conclude that from the models we created the best one is the one created by the method both. The variables that were used in the model are the following:

1. Image size: 60 x 60 (negative correlation with Revenues)
2. Image size: 44 x 556 (negative correlation with Revenues)
3. Image size: 8 x 15 (negative correlation with Revenues)

4. Image size: 24px x 133px (negative correlation with Revenues)
5. Image size: 400 x 300 (negative correlation with Revenues)
6. Image type: bmp (positive correlation with Revenues)
7. Loading time (negative correlation with Revenues)
8. Image type: jpeg (positive correlation with Revenues)
9. Readability (positive correlation with Revenues)
10. Instagram (positive correlation with Revenues)

Before we explain the results more elaborately we should first test some other methods as well just to make sure that we cannot improve even more the results of our model.

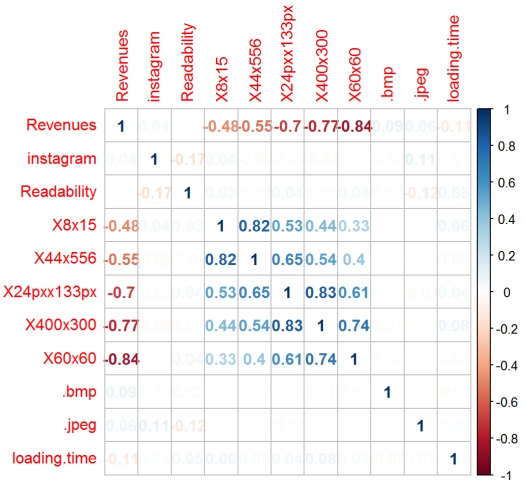
4.2.5 Comparisons and other methods

Now that we have analysed the metrics that are considered more important it would be wise to use other methods to double check the results. The code that was created for those models is available in the AppendixA.3.5.

4.2.5.1 Correlation testing

The next step of the analysis is to create a correlation table with the variables that are included in the model that had the best accuracy.

Table 54: Correlation graph of the best models' variables



From the table we can see that as expected the revenues are quite correlated with all the variables. But another observation that we can make is that some variables are highly correlated with each other as well. This means that we can try subtracting some of the variables (that are highly correlated with other variables) of the model and test whether or not it makes a difference in the final accuracy of it.
The highest correlations are between the variables 8x15 and 44x556 (82% correlated positive) and also between the variables 24px x 133px and 400x300 (83% positive correlation). From those four variables we will subtract one of each pair in order to see if the model that will be created will perform the same, better or worse from the initial one. We decided to remove from the model the variables 8x15 and 24px x 133px and the new model that was created is the following one:

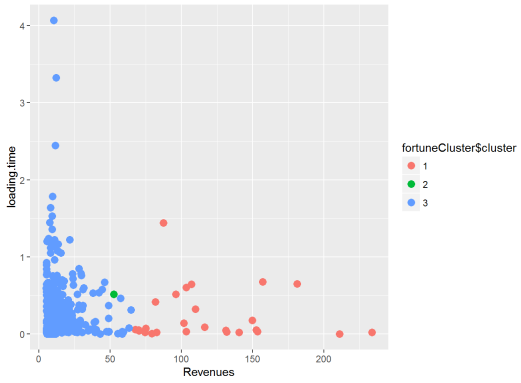
- **New model (from the both method) without the 2 variables**:
  Revenues = 203.12 - 88.71 (60x60) - 63.78 (44x556) - 39.25 (400x300) + 2.53 (bmp) - 5.54 (loading time) + 0.039 (jpeg)+ 1.04 (readability) + 2.3284 (instagram)
  - Residuals standard error : 14.16 on 339 degrees of freedom (any prediction will be off by 4.17%)
  - Adjusted R-squared : 0.8014 (80,14%)
  - p-value < 0.05

From the results of the new model we can see that by this subtraction the model became a little worse. The Adjusted R squared decreased while the percentage of not accuracy went up. So we can conclude that this change wasn't the best solution and even thought the variables are correlated they are also contribute to the better description of the dependent variable.

4.2.5.2 Clustering testing

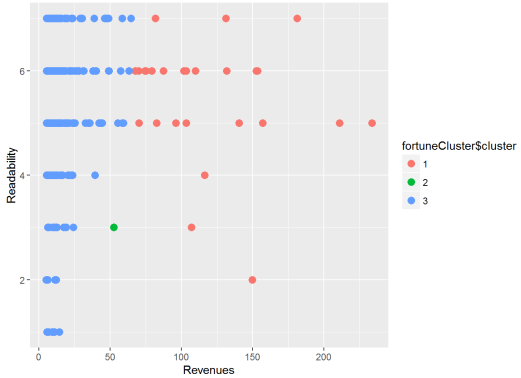
The next step is to create three clusters of the variables that are included in the model we created and to see if the way that they are grouping can also explain the revenues. If this hypothesis can be confirmed this means that the model we examined is indeed quite accurate.
The observations of the variables should have a clear pattern regarding the revenues in the way that they are grouped together. In order to check that we created the following plots:

Table 55: Loading time vs Revenues Clustering



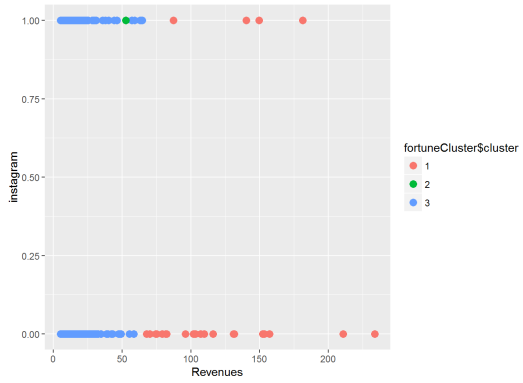
In the graph regarding the loading time we can see that the clusters are divided very clearly. The first cluster contains all the companies that have revenues more than 75 million dollars while the 3rd cluster contains all the companies that have less than 75 million dollars in revenues. The second cluster contains only one observation. It is not clear from this graph why this observation has been clustered alone but by continuing with the further analysis of the clusters with the other variables it would be more clear why the groups were divided in this way.

Table 56: Readability vs Revenues Clustering



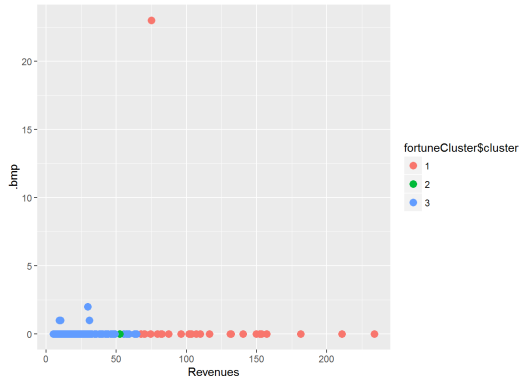
Again here the deviation happened in the revenue values of 75 million dollars except from the one observation of the second cluster.

Table 57: Instagram vs Revenues Clustering



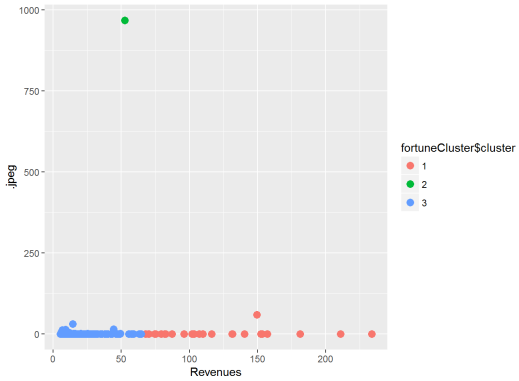
The same pattern also appears in the instagram variable graph. Here we can see that the variables of the cluster 1 that have the biggest revenues do not have instagram.

Table 58: Bmp vs Revenues Clustering



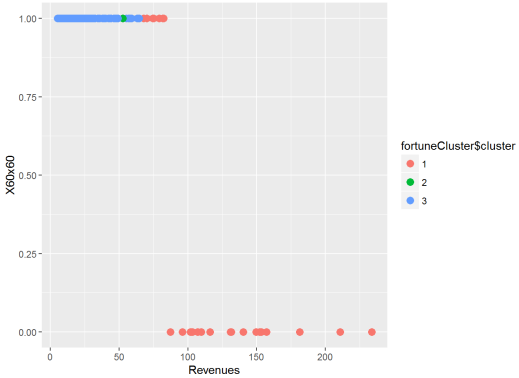
By the same logic is the graph for the bmp image type divided. Here we can see that the only site that have many bmp images is part of the high revenues cluster.

Table 59: Jpeg vs Revenues Clustering



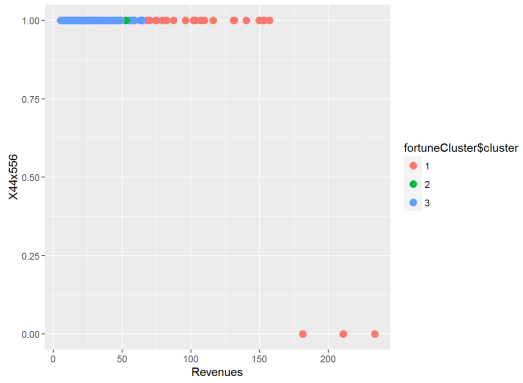
From the jpeg graph we can finally understand which variable was responsible for the second cluster. The number of jpeg images in this specific site were extremely high and that lead the clustering algorithm to create a unique group for the specific observation.
The next graphs are from the image sizes that are included in the model. As we have already seen in the analysis before we expect that the observations with the highest revenues won't have those type of images. And we expect the clustering to show this clearly.

Table 60: Image size: 60 x 60 vs Revenues Clustering



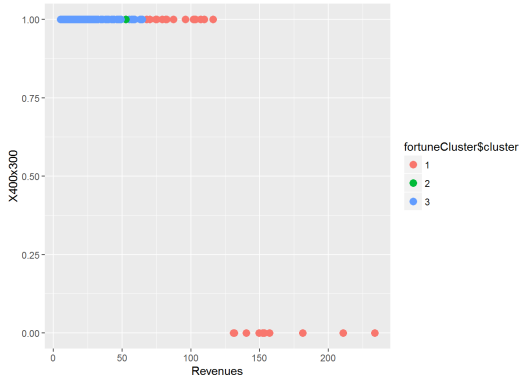
From the table for the image size 60 x 60 we can see that the 1st cluster has only zero prices which means that the companies with the highest revenues under consideration do not have this image size in their site's homepage.

Table 61: Image size: 44 x 556 vs Revenues Clustering



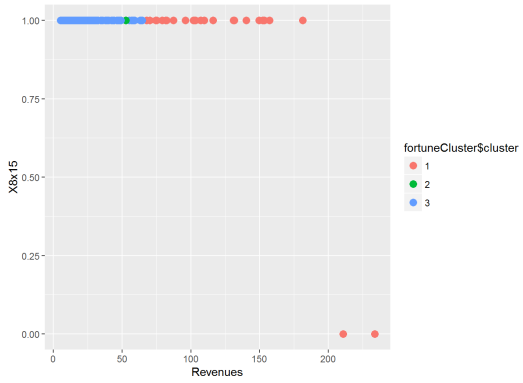
For the image size 44 x 556 we can see that only the 3 sites with the biggest revenues do not use this size of image in their initial pages on their websites.

Table 62: Image size: 400 x 300 vs Revenues Clustering



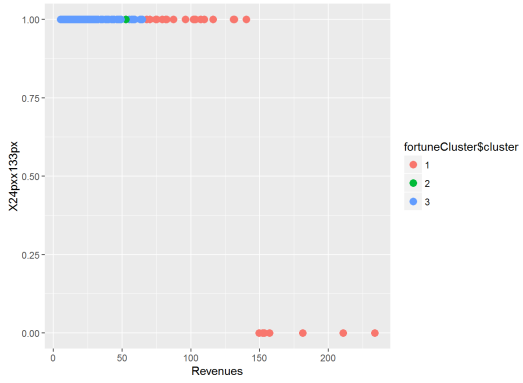
Almost half of the sites that belong to the first cluster which is the one with the highest revenues do not use this size on the images they use on their site.

Table 63: Image size: 8 x 15 vs Revenues Clustering



With the exception of the two most successful companies that we are examining all the other sites, regardless of the cluster they belong to do use the image size 8 x 15.

Table 64: Image size: 24px x 133px vs Revenues Clustering



Finally the only size of image that we are using in our analysis in pixels (24px x 133px) is used from the majority of the sites at hand with the exception of the 6 most successful ones.
From those graphs we can see that the image sizes have a negative relation regarding the revenues as the most successful ones do not use those dimensions even though the specific sizes are being used from the majority of the companies we are examining.
The variables that are not very clear on the correlation with the revenues is the readability and the instagram. While the bmp and the jpeg while they are

not being used from the majority of the companies the ones that do use them belong in the cluster with the high revenues.

4.2.5.3 Final testing

After clustering the variables in order to see if they can explain indeed the revenues we can also try a final different approach to see if we can improve our model. Based on the model we created the variable that has the highest value regarding the price it gets is the image size (60 x 60) which is (-88.85). So we will extract this variable from the model in order to see how this change effect the accuracy of the model that will be created. The new model is the following one:

- **New model without the 60 x 60 variable**:
  Revenues = 206.82 - 63.96 (44x556) - 126.11 (400x300) + 2.44 (bmp) - 4.19 (loading time) + 0.036 (jpeg)+ 0.42 (readability) + 0.84 (instagram)
  - Residuals standard error : 19.48 on 340 degrees of freedom (any prediction will be off by 5.72%)
  - Adjusted R-squared : 0.624 (80,14%)
  - p-value < 0.05

We can see from the model that the weight of image size 400 x 300 has changed dramatically. But the final result is a lot worst than the original model we have created. This means that the variable 60 x 60 is very important and we should keep it in the model.
After completing the analysis we can conclude that there are specific images dimensions that are correlated negative with a firm's revenues. Also there are two specific image types that has positive correlation with the revenues if they are used by the websites. More over the loading time of a site has a negative correlation with the revenues which means the most time it takes for a web site to upload it's pages the less revenues the company at hand will have. Also the only social media that seems to have a positive correlation with a company's revenues worth mentioning is instagram. That means that whether or not a company chooses to participate in the specific social medium is crucial for it's future success. Finally the ease of readability seems to have a positive correlation with the revenues. If we take into consideration that the way the readability variable was ranked this means that the more difficult a websites content is the more revenues will this firm have. That can be confirm if we consider that the users want to gain as much information as possible and they do not mind the difficulty of the text if the actual content is in fact informative.

5 Conclusions

With the completion of all the stages of the paper's analysis we showed the correlation between some of the most important metrics of a website and the revenues of the firm that owns the specific website. By gathering the elements we wanted to examine ,with the help of the programming language Python, we were able to perform a statistical analysis on them, with the use of the programming language R in the environment of the R-Studio, and moreover create regression models and predictive models that can provide us with a better understanding of the variables relationship with the revenues. Based on those results we can answer the questions that were raised in the beginning of this paper:

1. Do the revenues of a company relate to specific metrics of the company's website?
   There were specific metrics that seems to be correlated in a high level with the revenues of the company. These metrics are the loading time of the web page, the readability index, the existence of an instagram account link in the home page of the site, the usage of specific image types and the non usage of specific image dimensions.

2. Which of the metrics under examination are correlated the most with the revenues of each company of the Fortune 500 ones?

3. Can there be a predictive model that, based on the prices of the metrics that will be characterized as the more important ones from the findings, could forecast the revenues of an enterprise?

In the following sectors there will be a summary of the findings of this research along with an explanation on how those findings can contribute to the already existent studies.

5.1 Findings summary

With the completion of the analysis we can reach to the conclusion that the most important indicators that can be related to the actual success of a company are the following ones:

- **The loading time of the home page:**
  Page loading time is obviously an important part of any website's user experience. And in many cases companies let it slide to accommodate better aesthetic design, new nifty functionality or to add more content to web pages. Unfortunately, website visitors tend to care more about speed than the rest of the spectaculars that a website can offer. Additionally, page loading time is becoming a more important factor when it comes to search engine rankings.
  Search engines like Google use page loading time in algorithms that determine search engine rankings, meaning that they are more likely to guide

users to sites that load quickly. This can explain the fact that the loading time has a negative correlation with the company's status (in terms of revenues). As a result the findings of this paper comes to confirm the already established theory that loading time is an important factor of how a user is perceiving a website. The user perception has to do with his future endeavours with the firm and if he has an ill formed opinion he most likely won't use the company's product again. Based on these facts it is obvious that this parameter can actually interfere with the company's future success.

- **How comprehensible is the text on the site:**
  While a websites visual design is how a company looks online, the sites writing is how it sounds online. Tone of voice and great content are crucial for communicating on the Internet. Nevertheless, the best copywriting is for nothing if users dont read the text. As with other areas of user experience, content has to survive a costbenefit analysis on the part of users: Cost: how much hassle and pain do I have to suffer on this website? Benefits: Whats in it for me, what will I gain if I read this information? For online copy, the barriers to use fall into 3 categories: legibility, readability, and comprehension, each of which is defined and discussed below. Readability measures the complexity of the words and sentence structure in a piece of content. The assumption behind this metric is that complex sentences are harder to parse and read than simpler ones. Its usually reported as the reading level (stated as years of formal education) needed to easily read the text. For example, a 12th grade reading level means that somebody with a good high-school diploma will be able to read the text without difficulty.

- **Whether or not the company choose to have an instagram account**:
  For social media natives, an Instagram account is a stamp of authority and authenticity. These connected customers will often search for an Instagram profile *before* searching for a website. The assumption is that a website is a static PR/Advertising fabrication, as authentic as a classified ad. Instagram shows and tells more, and more is what todays consumers expect.

- **Whether or not a company uses specific image sizes**:
  The fact that the sizes of the images are extremely important on the status of a company is not a surprise. There has been several studies that shows that when a user is visiting a site he usually focus on specific parts of it. The content/ images the company chooses to use can be crucial to how the user will perceive the website an whether or not he will choose to continue looking around on it.

- **Whether or not the images that are being used are of specific types**:

The type of images are correlated with the loading time of the website in a very high level.

5.2 Findings contribution

5.2.1 Theoretical level

5.2.2 Managerial level

6 Further Research

7 Bibliography

References

[1] Abels E. G.,White M. D. ,Hahn K. (1997). Identifying userbased criteria for Web pages. Internet Research, 7:4, 252-262

[2] Agarwal R. and Venkatesh V. (2002). Assessing a Firm's Web Presence: A Heuristic Evaluation Procedure for the Measurement of Usability. Information Systems Research, 13:2,168-186

[3] Bailey 1. and Pearson S.W.(1983). Development of a Tool for Measuring and Analyzing Computer User Satisfaction. Management Science, 29:5, 530-545

[4] Bruce H. (1998). User Satisfaction with Information Seeking on the Internet. Journal of the American Society of Information Sciences, 49:6, 541-556

[5] Cohen J.B. (1960). A Coefficient of Agreement for Nominal Scales. Educational and Psychology Measurement, 20, 37-46

[6] Davis F.D., Bagozzi RP., Warshaw P.R. (1989). User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. Management Science, 35:8, 982-1003

[7] Evans P. and Wurster T.(2000). Blown to Bits. Boston: Harvard Business School Press

[8] Fan X., Salvendy G. (2003). Customer-centered rules for design of e-commerce Web sites. Communications of the ACM, 46(12), 332-336

[9] Forte R. M.(2015). Mastering Predictive Analytics with R. Packt Publishing Ltd.

[10] Hoffman D. L., Novak T.P.(2000). How to acquire customers on the Web. Harvard Business Review, 78(3), 179-188

[11] Kaplan A.M., Haenlein M.(2010). Users of the word united! The challenges and opportunities of social media. Business Horizons, 53, 59-68

[12] Katerattanukul P. and Siau K. (1999). Measuring Information Quality of Web Sites: Development of an Instrument. International Conference Information Systems, 279-285

[13] Liao C., To P., Shih M.(2006). Web site practices: A comparison between the top 1000 companies in the US and Taiwan. Information Systems Research, 26, 196-211

[14] Liu, C., Arnett, K.P. (2000). Exploring the factors associated with Web site success in the context of electronic commerce. Information Management, 38(1), 2333

[15] Liu, C., Arnett, K. P., Capella, L. M., Beatty, R. C. (1997). Web sites of the Fortune 500 companies: Facing customers through home pages. Information and Management, 31(6), 335345

[16] McKinney V., Yoon K., Zahedi F.M. (2002). The Measurement of Web-Customer Satisfaction: An Expectation and Dis confirmation Approach. Information Systems Research, 13:3, 296-315

[17] Monideepa T., Jie Z.(2005) Analysis of Critical Website Characteristics: A Cross-Category Study of Successful Websites, Journal of Computer Information Systems, 46:2, 14-24

[18] Nielsen J. (2000). Designing Web Usability. Indianapolis, IN:New Riders

[19] Palmer J. W. (2002). Web site usability, design, and performance metrics. Information Systems Research, 13(2), 151-167

[20] Rose G.,Khoo H.,Straub D. (1999). Current Technological Impediments to Business-to-Consumer Electronic Commerce. Communications of the AIS, 1:16, 1-74

[21] Shneiderman B. (1998). Designing the User Interface: Strategies for Effective Human-Computer Interaction. MA: Addison-Wesley

[22] VonDran G.M, Zhang P.,Small R. (2002). Quality Websites: An Application of the Kano Model to Website Design. Proceedings of the Americas Conference on Information Systems, 898-900

[23] Wei-Shang Fan, Ming-Chun Tsai (2010). Factors driving website success: the key role of Internet customisation and the influence of website design quality and Internet marketing strategy. Total Quality Management and Business Excellence, 21:11, 1141-1159

[24] Wilkerson G.L., Bennett L.T. , Oliver K.M. (1997). Evaluation Criteria and Indicators of Quality for Internet Resources. Educational Technology, 37, 52-59

[25] Young D., Benamati J. (1999). Differences in Public Web sites: The Current State of Large U.S. Firms. Journal of Electronic Commerce Research, 94-105

[26] Statista (2017). Retrieved February 26, 2017, from $https : //www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/$

[27] Statista (2017). Retrieved February 26, 2017, from *https* : *//www.statista.com/statistics/*282087*/number−of−monthly−active − twitter − users/*

[28] Statista (2017). Retrieved February 26, 2017, from *https* : *//www.statista.com/statistics/*253577*/number−of−monthly−active − instagram − users/*

[29] Statista (2017). Retrieved February 26, 2017, from *https* : *//www.statista.com/statistics/*274050*/quarterly − numbers − of − linkedin − members/*

[30] Forbes (2016). Retrieved February 26, 2017, from *https* : *//www.forbes.com/sites/kathleenchaykowski/*2016*/*10*/*13*/pinterest−reaches* − 150 − *million − monthly − users/*#269*f*39*e*0732*e*

[31] Fortune Lords (2017). Retrieved February 26, 2017, from *https* : *//fortunelords.com/youtube − statistics/*

[32] Zyxware (2017). Retrieved January 29, 2017, from *http* : *//www.zyxware.com/articles/*4344*/list − of − fortune* − 500 − *companies − and − their − websites*

[33] Forbes (2015). Retrieved February 26, 2017, from *https* : *//www.forbes.com/sites/kathleenchaykowski/*2015*/*12*/*08*/facebook−business − pages − climb − to* − 50 − *million − with − new − messaging − tools/*#4*b*75*d*1*d*46991

[34] Forbes (2016). Retrieved February 26, 2017, from *https* : *//www.forbes.com/sites/kathleenchaykowski/*2016*/*10*/*13*/pinterest−reaches* − 150 − *million − monthly − users/*#2*a*5*f*63*f*4732*e*

[35] Fortune 500 (2017). Retrieved January 29, 2017, from *http* : *//beta.fortune.com/fortune*500

[36] Readability test tool (2017). Retrieved January 29, 2017, from *http* : *//www.webpagefx.com/tools/read − able/*

[37] Gaur L., Singh G., Kumar S. (2016). Google Analytics: A tool to make website more Robust. ICTCS

[38] Tables Generator (2017). Retrieved January 29, 2017, from *http* : *//www.tablesgenerator.com/*

[39] Continuum (2017). Retrieved January 29, 2017, from *https* : *//www.continuum.io/downloads*

[40] Validator W3 (2017). Retrieved January 29, 2017, from *https* : *//validator.w*3*.org/*

[41] The Python Tutorial (2016). Available on line from *https* : *//docs.python.org/*3*/tutorial/index.html*

A Appendix

A.1 Appendix A: Fortune 500 Companies

Table 65: Fortune 500 - 50 first companies

1. Walmart	2. Exxon Mobil	3. Apple
4. Berkshire Hathaway	5. McKesson	6. UnitedHealth Group
7. CVS Health	8. General Motors	9. Ford Motor
10. AT&T	11. General Electric	12. AmerisourceBergen
13. Verizon	14. Chevron	15. Costco
16. Fannie Mae	17. Kroger	18. Amazon.com
19. Walgreens Boots Alliance	20. HP	21. Cardinal Health
22. Express Scripts Holding	23. J.P. Morgan Chase	24. Boeing
25. Microsoft	26. Bank of America Corp.	27. Wells Fargo
28. Home Depot	29. Citigroup	30. Phillips 66
31. IBM	32. Valero Energy	33. Anthem
34. Procter & Gamble	35. State Farm Insurance Cos.	36. Alphabet
37. Comcast	38. Target	39. Johnson & Johnson
40. MetLife	41. Archer Daniels Midland	42. Marathon Petroleum
43. Freddie Mac	44. PepsiCo	45. United Technologies
46. Aetna	47. Lowe's	48. UPS
49. AIG	50. Prudential Financial	

Table 66: Fortune 500 - Companies Ranked: 51 - 100

51. Intel	52. Humana	53. Disney
54. Cisco Systems	55. Pfizer	56. Dow Chemical
57. Sysco	58. FedEx	59. Caterpillar
60. Lockheed Martin	61. N.Y. Life Insurance	62. Coca-Cola
63. HCA Holdings	64. Ingram Micro	65. Energy Transfer Equity
66. Tyson Foods	67. American Airlines Group	68. Delta Air Lines
69. Nationwide	70. Johnson Controls	71. Best Buy
72. Merck	73. Liberty Mutual I.G.	74. Goldman Sachs Group
75. Honeywell International	76. Massachusetts Mutual L.I.	77. Oracle
78. Morgan Stanley	79. Cigna	80. U.C. Holdings
81. Allstate	82. TIAA	83. INTL FCStone
84. CHS	85. American Express	86. Gilead Sciences
87. Publix Super Markets	88. General Dynamics	89. TJX
90. ConocoPhillips	91. Nike	92. World Fuel Services
93. 3M	94. Mondelez International	95. Exelon
96. Twenty-First Century Fox	97. Deere	98. Tesoro
99. Time Warner	100. Northwestern Mutual	

Table 67: Fortune 500 - Companies Ranked: 101 - 150

101. DuPont	102. Avnet	103. Macy's
104. Enterprise Products Partners	105. Travelers Cos.	106. Philip Morris International
107. Rite Aid	108. Tech Data	109. McDonald's
110. Qualcomm	111. Sears Holdings	112. Capital One Financial
113. EMC	114. USAA	115. Duke Energy
116. Time Warner Cable	117. Halliburton	118. Northrop Grumman
119. Arrow Electronics	120. Raytheon	121. Plains GP Holdings
122. US Foods Holding	123. AbbVie	124. Centene
125. Community Health Systems	126. Alcoa	127. International Paper
128. Emerson Electric	129. Union Pacific	130. Amgen
131. U.S. Bancorp	132. Staples	133. Danaher
134. Whirlpool	135. Aflac	136. AutoNation
137. Progressive	138. Abbott Laboratories	139. Dollar General
140. Tenet Healthcare	141. Eli Lilly	142. Southwest Airlines
143. Penske Automotive Group	144. ManpowerGroup	145. Kohl's
146. Starbucks	147. Paccar	148. Cummins
149. Altria Group	150. Xerox	

Table 68: Fortune 500 - Companies Ranked: 151 - 200

151. Kimberly-Clark	152. Hartford F.S.G.	153. Kraft Heinz
154. Lear	155. Fluor	156. AECOM
157. Facebook	158. Jabil Circuit	159. CenturyLink
160. Supervalu	161. General Mills	162. Southern
163. NextEra Energy	164. Thermo Fisher Scientific	165. American Electric Power
166. PG&E Corp.	167. NGL Energy Partners	168. Bristol-Myers Squibb
169. Goodyear Tire & Rubber	170. Nucor	171. PNC F.S.G.
172. Health Net	173. Micron Technology	174. Colgate-Palmolive
175. Freeport-McMoRan	176. ConAgra Foods	177. Gap
178. Baker Hughes	179. Bank of N.Y. Mellon C.	180. Dollar Tree
181. Whole Foods Market	182. PPG Industries	183. Genuine Parts
184. Icahn Enterprises	185. Performance Food Group	186. Omnicom Group
187. DISH Network	188. FirstEnergy	189. Monsanto
190. AES	191. CarMax	192. National Oilwell Varco
193. NRG Energy	194. Western Digital	195. Marriott International
196. Office Depot	197. Nordstrom	198. Kinder Morgan
199. Aramark	200. DaVita HealthCare Partners	

Table 69: Fortune 500 - Companies Ranked: 201 - 250

201. Molina Healthcare	202. WellCare Health Plans	203. CBS
204. Visa	205. Lincoln National	206. Ecolab
207. Kellogg	208. C.H. Robinson Worldwide	209. Textron
210. Loews	211. Illinois Tool Works	212. Synnex
213. Viacom	214. HollyFrontier	215. Land O'Lakes
216. Devon Energy	217. PBF Energy	218. Yum Brands
219. Texas Instruments	220. CDW	221. Waste Management
222. Marsh & McLennan	223. Chesapeake Energy	224. Parker-Hannifin
225. Occidental Petroleum	226. Guardian Life I.C.A.	227. Farmers Ins. Exchange
228. J.C. Penney	229. Consolidated Edison	230. Cognizant Tech. Solutions
231. VF	232. Ameriprise Financial	233. Computer Sciences
234. L Brands	235. Jacobs Engineering Group	236. Principal Financial
237. Ross Stores	238. Bed Bath & Beyond	239. CSX
240. Toys R Us	241. Las Vegas Sands	242. Leucadia National
243. Dominion Resources	244. United States Steel	245. L-3 Communications
246. Edison International	247. Entergy	248. ADP
249. First Data	250. BlackRock	

Table 70: Fortune 500 - Companies Ranked: 251 - 300

251. WestRock	252. Voya Financial	253. Sherwin-Williams
254. Hilton Worldwide Holdings	255. R.R. Donnelley & Sons	256. Stanley Black & Decker
257. Xcel Energy	258. Murphy USA	259. CBRE Group
260. D.R. Horton	261. Estee Lauder	262. Praxair
263. Biogen	264. State Street Corp.	265. Unum Group
266. Reynolds American	267. Group 1 Automotive	268. Henry Schein
269. Hertz Global Holdings	270. Norfolk Southern	271. Reinsurance G. of America
272. Public Service E. G.	273. BB&T Corp.	274. DTE Energy
275. Assurant	276. Global Partners	277. Huntsman
278. Becton Dickinson	279. Sempra Energy	280. AutoZone
281. Navistar International	282. Precision Castparts	283. Discover F. S.
284. Liberty Interactive	285. W.W. Grainger	286. Baxter International
287. Stryker	288. Air Products & Chemicals	289. Western Refining
290. Universal Health Services	291. Owens & Minor	292. Charter Communications
293. Advance Auto Parts	294. MasterCard	295. Applied Materials
296. Eastman Chemical	297. Sonic Automotive	298. Ally Financial
299. CST Brands	300. eBay	

Table 71: Fortune 500 - Companies Ranked: 301 - 350

301. Lennar	302. GameStop	303. Reliance Steel & Aluminum
304. Hormel Foods	305. Celgene	306. Genworth Financial
307. PayPal Holdings	308. Priceline Group	309. MGM Resorts International
310. Autoliv	311. Fidelity National Financial	312. Republic Services
313. Corning	314. Peter Kiewit Sons'	315. Univar
316. Mosaic	317. Core-Mark Holding	318. Thrivent F. for Lutherans
319. Cameron International	320. HD Supply Holdings	321. Crown Holdings
322. EOG Resources	323. Veritiv	324. Anadarko Petroleum
325. Laboratory C. of A.	326. Pacific Life	327. News Corp.
328. Jarden	329. SunTrust Banks	330. Avis Budget Group
331. Broadcom	332. American Family I. G.	333. Level 3 Communications
334. Tenneco	335. United Natural Foods	336. Dean Foods
337. Campbell Soup	338. Mohawk Industries	339. BorgWarner
340. PVH	341. Ball	342. O'Reilly Automotive
343. Eversource Energy	344. Franklin Resources	345. Masco
346. Lithia Motors	347. KKR	348. Oneok
349. Newmont Mining	350. PPL	

Table 72: Fortune 500 - Companies Ranked: 351 - 400

351. SpartanNash	352. Quanta Services	353. XPO Logistics
354. Ralph Lauren	355. Interpublic Group	356. Steel Dynamics
357. WESCO International	358. Quest Diagnostics	359. Boston Scientific
360. AGCO	361. Foot Locker	362. Hershey
363. CenterPoint Energy	364. Williams	365. Dick's Sporting Goods
366. Live Nation Entertainment	367. Mutual of Omaha Ins.	368. W.R. Berkley
369. LKQ	370. Avon Products	371. Darden Restaurants
372. Kindred Healthcare	373. Weyerhaeuser	374. Casey's General Stores
375. Sealed Air	376. Fifth Third Bancorp	377. Dover
378. Huntington Ingalls Industries	379. Netflix	380. Dillard's
381. EMCOR Group	382. Jones Financial	383. AK Steel Holding
384. UGI	385. Expedia	386. salesforce.com
387. Targa Resources	388. Apache	389. Spirit AeroSystems H.
390. Expeditors Inter. of Washington	391. Anixter International	392. Fidelity N. Inf. S.
393. Asbury Automotive Group	394. Hess	395. Ryder System
396. Terex	397. Coca-Cola Eur. P.	398. Auto-Owners Insurance
399. Cablevision Systems	400. Symantec	

Table 73: Fortune 500 - Companies Ranked: 401 - 450

401. Charles Schwab	402. Calpine	403. CMS Energy
404. Alliance Data Systems	405. JetBlue Airways	406. Discovery Communic.
407. Trinity Industries	408. Sanmina	409. NCR
410. FMC Technologies	411. Erie Insurance Group	412. Rockwell Automation
413. Dr Pepper Snapple Group	414. iHeartMedia	415. Tractor Supply
416. J.B. Hunt Transport Services	417. Commercial Metals	418. Owens-Illinois
419. Harman Inter. Ind.	420. Baxalta	421. American F. G.
422. NetApp	423. Graybar Electric	424. Oshkosh
425. Ameren	426. A-Mark Precious Metals	427. Barnes & Noble
428. Dana Holding	429. Constellation Brands	430. LifePoint Health
431. Zimmer Biomet H.	432. Harley-Davidson	433. PulteGroup
434. Newell Brands	435. Avery Dennison	436. Jones Lang LaSalle
437. WEC Energy Group	438. Marathon Oil	439. TravelCenters of A.
440. United Rentals	441. HRG Group	442. Old Republic Inter.
443. Windstream Holdings	444. Starwood Hotels & Resorts	445. Delek US Holdings
446. Packaging Corp. of A.	447. Quintiles Transnational H.	448. Hanesbrands
449. Realogy Holdings	450. Mattel	

Table 74: Fortune 500 - Companies Ranked: 451 - 500

451. Motorola Solutions	452. J.M. Smucker	453. Regions Financial
454. Celanese	455. Clorox	456. Ingredion
457. Genesis Healthcare	458. Peabody Energy	459. Alaska Air Group
460. Seaboard	461. Frontier Communic.	462. Amphenol
463. Lansing Trade Group	464. SanDisk	465. St. Jude Medical
466. Wyndham Worldwide	467. Kelly Services	468. Western Union
469. Envision Healthcare H.	470. Visteon	471. Arthur J. Gallagher
472. Host Hotels & Resorts	473. Ashland	474. Insight Enterprises
475. Energy Future Holdings	476. Markel	477. Essendant
478. CH2M Hill	479. Western & Southern F.G.	480. Owens Corning
481. S&P Global	482. Raymond James Financial	483. NiSource
484. Airgas	485. ABM Industries	486. Citizens F.G.
487. Booz Allen Hamilton H.	488. Simon Property Group	489. Domtar
490. Rockwell Collins	491. Lam Research	492. Fiserv
493. Spectra Energy	494. Navient	495. Big Lots
496. Telephone & Data Systems	497. First American Financial	498. NVR
499. Cincinnati Financial	500. Burlington Stores	

A.2 Appendix B: Python Scripts

Script 1: Initial lists

```
list_company_number =[]
list_company_name = []
list_company_website = []
```

Script 2: Python Libraries

```
import urllib
import urllib2
import time
import os
from bs4 import BeautifulSoup
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Script 3: Companies ranking, names and url

```
def websites (url):
    from time import time
    start = time ()
    browser = urllib2.build_opener()
    browser.addheaders = [('User-agent', 'Mozilla/5.0')]
    response = browser.open(url)
    myHTML = response.read()
    soup = BeautifulSoup(myHTML,"lxml")
    o = 0
    td_list =[]
    for row2 in soup.html.body.findAll('td'):
        td_list.insert(o, row2)
        o = o + 1
    a = 0
    b = 1
    c = 2
    list_numbering = 0
    for i in range (0,500):
        num = str(td_list[a])
        company = str(td_list[b])
        site = str(td_list[c])
        c_num = re.findall('>(.+?)</td>',num)
        c_num = str(c_num[0])
```

```
        c_name = re.findall('>(.+?)</td>',company)
        c_name = str(c_name[0])
        c_site = re.findall('">(.+?)</a>',site)
        c_site = str(c_site[0])
        list_company_number.insert(list_numbering,c_num)
        list_company_name.insert(list_numbering,c_name)
        list_company_website.insert(list_numbering,c_site)
        a = a + 3
        b = b + 3
        c = c + 3
        list_numbering =  list_numbering + 1
    end = time ()
    duration = round (end - start, 1)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', duration, ' seconds'
    print 'The lists are ready in ', minutes, ' minutes'
```

Script 4: URL Validation

```
nv = 0
for num in range(len(list_company_website)):
    line = 'http://' + str(list_company_website[num])
    x = validators.url(line)
    if x != True:
        nv = nv +1
print "The validation is complete! There were" , nv,
 "not valid pages"
```

Script 5: Download sites HTML

```
import time
browser2 = urllib2.build_opener()
browser2.addheaders = [('User-agent', 'Mozilla/5.0')]
for i in range (0,500):
    k = str(i + 1)
    lc = str(list_company_website[i])
    lc = lc.replace("'","")
    lc = lc.replace("[","")
    lc = lc.replace("]","")
    lcn = str(list_company_name[i])
    lcn = lcn.replace("'","")
    lcn = lcn.replace("[","")
    lcn = lcn.replace("]","")
    url2= 'http://' + lc
    list500_names.insert(i,lcn)
```

```
    list500_url.insert(i,lc)
    list500_num.insert(i,k)
    if i == 118 or i == 464 or i == 70:
        list500_sites.insert(i,0)
        print ("The site " + str(i)
        + " has NOT been downloaded!")
    else:
        try:
            response2=browser2.open(url2)
            print ("The site " + str(i)
            + " has been downloaded!")
        except Exception:
            list500_sites.insert(i,0)
            print ("The site " + str(i)
            + " has NOT been downloaded from exception!")
            continue
        myHTML2=response2.read()
        list500_sites.insert(i,myHTML2)
        time.sleep(2)
```

Script 6: Not downloadable HTML

```
not_d = []
not_d_n = []
num = []
def not_downloadables (list500_names,list500_sites):
    met = 0
    for i in range(len(list500_names)):
        if list500_sites[i] == 0:
            ct = list500_names[i]
            not_d.insert(met,ct)
            not_d_n.insert(met,str(i))
            num.insert(met,met)
            met = met + 1

not_downloadables (list500_names,list500_sites)

d = {'company' : pd.Series(not_d, index=[num]),
     'number' : pd.Series(not_d_n, index=[num])}
nd = pd.DataFrame(d)
nd
```

Script 7: Content

```
flesch = []
sentence = []
word = []
unique_w =[]
empty =[]

import time
for num in range(0,500):
    site = list500_sites[num]
    line = list500_url[num]
    url_check = "http://www.webpagefx.com/tools/
____read-able/check.php?tab=Test+By+Url&uri=http://"+ line
    browser = urllib2.build_opener()
    browser.addheaders = [('User-agent', 'Mozilla/5.0')]
    if site == 0 or num == 107:
        print("Site", str(num), "is not validated from sites")
        flesch.insert(num,"n/a")
        sentence.insert(num,"n/a")
        word.insert(num,"n/a")
        unique_w.insert(num,"n/a")
    else:
        try:
            response = browser.open(url_check)
        except Exception:
            flesch.insert(num,"n/a")
            sentence.insert(num,"n/a")
            word.insert(num,"n/a")
            unique_w.insert(num,"n/a")
            print("Site", str(num), "is not validated from check")
            continue
        html_r = response.read()
        check = str(html_r)
        if check != empty:
                soup = BeautifulSoup(check,"lxml")
                o = 0
                keyf = []
                for row in soup.html.body.findAll('tr'):
                    keyf.insert(o,row)
                    o = o + 1
                if keyf != empty:
                        print("Site", str(num), "is validated")
                        #Flesh measurement
                        if keyf[0] != empty:
                            readability = str(keyf[0])
                            split1 = readability.split('>')
                            readability2 = str(split1[4])
```

```
        split2 = readability2.split('<')
        readability3 = str(split2[0])
        flesch.insert(num,readability3)
    else:
        flesch.insert(num,"n/a")
        sentence.insert(num,"n/a")
        word.insert(num,"n/a")
        unique_w.insert(num,"n/a")
    #Number of sentences
    if keyf[6] != empty:
        sentences = str(keyf[6])
        spli1 = sentences.split('>')
        sentences2 = str(spli1[4])
        spli2 = sentences2.split('<')
        sentences3 = str(spli2[0])
        sentence.insert(num,sentences3)
    else:
        flesch.insert(num,"n/a")
        sentence.insert(num,"n/a")
        word.insert(num,"n/a")
        unique_w.insert(num,"n/a")
    #Number of words
    if keyf[7] != empty:
        words = str(keyf[7])
        spl1 = words.split('>')
        words2 = str(spl1[4])
        spl2 = words2.split('<')
        words3 = str(spl2[0])
        word.insert(num,words3)
    else:
        flesch.insert(num,"n/a")
        sentence.insert(num,"n/a")
        word.insert(num,"n/a")
        unique_w.insert(num,"n/a")
    #No. of complex words
    if keyf[7] != empty:
        unique_ws = str(keyf[8])
        sp1 = unique_ws.split('>')
        unique_ws2 = str(sp1[4])
        sp2 = unique_ws2.split('<')
        unique_ws3 = str(sp2[0])
        unique_w.insert(num,unique_ws3)
    else:
        flesch.insert(num,"n/a")
        sentence.insert(num,"n/a")
        word.insert(num,"n/a")
```

```
                                unique_w.insert(num,"n/a")
                else:
                        print("Site", str(num), "is not validated from check 2")
                        flesch.insert(num,"n/a")
                        sentence.insert(num,"n/a")
                        word.insert(num,"n/a")
                        unique_w.insert(num,"n/a")
    time.sleep(2)
```

Script 8: Readability variable

```
readability = []
def readable (flesch):
    for i in range (len(flesch)):
            f_n = flesch[i]
            if f_n == "n/a":
                readability.insert(i,"n/a")
            else:
                a = int(float(f_n))
                if a > 90:
                    readability.insert(i,"Very easy")
                elif a > 80:
                    readability.insert(i,"Easy")
                elif a > 70:
                    readability.insert(i,"Fairly easy")
                elif a > 60:
                    readability.insert(i,"Standard")
                elif a > 50:
                    readability.insert(i,"Fairly difficult")
                elif a > 30:
                    readability.insert(i,"Difficult")
                else:
                    readability.insert(i,"Very Confusing")
    print "The function is completed!"

readable (flesch)

d1 = {'company' : pd.Series(list500_names, index=[list500_num]),
      'url' : pd.Series(list500_url, index=[list500_num]),
      'Readability' : pd.Series(readability, index=[list500_num]),
      'Flesh_Mesaure' : pd.Series(flesch,index=[list500_num]),
'Sentences' : pd.Series(sentence, index=[list500_num]),
'Words' : pd.Series(word, index=[list500_num]),
'Unique_words' : pd.Series(unique_w, index=[list500_num])}
fre = pd.DataFrame(d1)
```

Script 9: HTML Validation

```
num_errors = []
num_info_warnings = []
num_non_doc = []
nm = []
num_open_page = []
empty = ""

def html_validation (list500_url,list500_names):
    from time import time # I used it to see how much time it does to run the fu
    start = time ()
    for num in range(len(list500_names)):
        line = list500_url[num]
        url_check = "https://validator.w3.org/nu/?doc=https://" + line
        browser = urllib2.build_opener()
        browser.addheaders = [('User-agent', 'Mozilla/5.0')]
        response = browser.open(url_check)
        html_check = response.read()
        html_check
        check = str(html_check)
        er = 0
        err = 0
        errr = 0
        e = False
        if check != empty:
            e = True
            soup = BeautifulSoup(check,"lxml")
            o = 0
            keyf = []
            for row in soup.html.body.findAll('div'):
                keyf.insert(o,row)
                o = o + 1
            if len(keyf) != 0:
                    keyfin = str(keyf[2])
                    dol= re.findall('class="error"',keyfin)
                    er = er + len(dol)
                    doll= re.findall('class="info_warning"'
                                     ,keyfin)
                    err = err + len(doll)
                    dolll= re.findall('class="non-document-error_io"'
                                      ,keyfin)
                    errr = errr + len(dolll)
        num_errors.insert(num,er)
        num_info_warnings.insert(num,err)
        num_non_doc.insert(num,errr)
```

```
            nm.insert(num,num)
            num_open_page.insert(num,e)
        end = time ()
        duration = round (end - start, 3)
        minutes = round (duration /60, 1)
        print 'The lists are ready in ', minutes, ' minutes'


html_validation (list500_url,list500_names)


d8 = {'company' : pd.Series(list500_names, index=[nm]),
      'The_page_opened' : pd.Series(num_open_page, index=[nm])
      ,'number_of_errors' : pd.Series(num_errors, index=[nm]),
      'number_of_warning' : pd.Series(num_info_warnings, index=[nm])
      ,'non-document-error' : pd.Series(num_non_doc, index=[nm])}
html_val = pd.DataFrame(d8)
html_val.head(3)
```

Script 10: Social media variables

```
sm_f = []
sm_t = []
sm_i = []
sm_p = []
sm_y = []
sm_l = []
sm_nm = []
nm = []
sm_url = []

def socialmedia (list500_sites,list500_names,list500_url):
    from time import time
    start = time ()
    for i in range(len(list500_names)):
            myHTML = list500_sites[i]
            sm = ['facebook.com','twitter.com',
                  'instagram.com','pinterest.com',
                  'youtube.com','linkedin.com']
            if myHTML == 0:
                sm_nm.insert(i,list500_names[i])
                nm.insert(i,i)
                sm_url.insert(i,list500_url[i])
                sm_f.insert(i,'n/a')
                sm_t.insert(i,'n/a')
```

```
            sm_i.insert(i,'n/a')
            sm_p.insert(i,'n/a')
            sm_y.insert(i,'n/a')
            sm_l.insert(i,'n/a')
        else:
            for index in range(len(sm)):
                x = sm[index]
                social = re.findall(x,myHTML)
                if (len(social) > 0):
                    if x == 'facebook.com':
                        answerf = 'TRUE'
                    if x == 'twitter.com':
                        answert = 'TRUE'
                    if x == 'instagram.com':
                        answeri = 'TRUE'
                    if x == 'pinterest.com':
                        answerp = 'TRUE'
                    if x == 'youtube.com':
                        answery = 'TRUE'
                    if x =='linkedin.com':
                        answerl = 'TRUE'
                else:
                     if x == 'facebook.com':
                        answerf = 'FALSE'
                     if x == 'twitter.com':
                        answert = 'FALSE'
                     if x == 'instagram.com':
                        answeri = 'FALSE'
                     if x == 'pinterest.com':
                        answerp = 'FALSE'
                     if x == 'youtube.com':
                        answery = 'FALSE'
                     if x =='linkedin.com':
                        answerl = 'FALSE'
                sm_nm.insert(i,list500_names[i])
                nm.insert(i,i)
                sm_url.insert(i,list500_url[i])
                sm_f.insert(i,answerf)
                sm_t.insert(i,answert)
                sm_i.insert(i,answeri)
                sm_p.insert(i,answerp)
                sm_y.insert(i,answery)
                sm_l.insert(i,answerl)
end = time ()
duration = round (end - start, 3)
minutes = round (duration /60, 1)
```

```
    print 'The lists are completed in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

socialmedia (list500_sites,list500_names,list500_url)

d2 = {'company' : pd.Series(sm_nm, index=[nm]),
      'facebook' : pd.Series(sm_f, index=[nm]),
       'twitter' : pd.Series(sm_t, index=[nm]),
      'instagram' : pd.Series(sm_i, index=[nm]),
       'pinterest' : pd.Series(sm_p, index=[nm]),
      'youtube' : pd.Series(sm_y, index=[nm]),
       'linkedin' : pd.Series(sm_l, index=[nm]),}
social_media = pd.DataFrame(d2)
```

Script 11: Links Internal and External variables

```
l_nm = []
l_ex = []
l_in = []
l_t = []
nm = []
l_url = []


def links (list500_sites,list500_names,list500_url):
    from time import time
    start = time ()
    for num in range(len(list500_names)):
            myHTML = list500_sites[num]
            if myHTML == 0:
                l_nm.insert(num,list500_names[num])
                l_ex.insert(num,'n/a')
                l_t.insert(num,'n/a')
                l_in.insert(num,'n/a')
                nm.insert(num,num)
            else:
                href = re.findall('href',myHTML)
                external = re.findall('href="https:',myHTML)
                ex = (len(external))
                alllinks = (len(href))
                internal =  (len(href) - len(external))
                l_nm.insert(num,list500_names[num])
                l_ex.insert(num,ex)
                l_t.insert(num,alllinks)
                l_in.insert(num,internal)
```

```
                nm.insert(num,num)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

links (list500_sites,list500_names,list500_url)

d3 = {'company' : pd.Series(l_nm, index=[nm]),
      'external' : pd.Series(l_ex, index=[nm]),
      'internal' : pd.Series(l_in, index=[nm]),
     'total_links' : pd.Series(l_t, index=[nm])}
sites_links = pd.DataFrame(d3)
```

Script 12: Loading time variable

```
lt_nm = []
lt_time = []
nm = []
lt_url = []

def loadtime (list_company_website,list500_names,list500_url):
    from time import time
    browser2 = urllib2.build_opener()
    browser2.addheaders = [('User-agent', 'Mozilla/5.0')]
    for num in range(len(list500_names)):
        lc = str(list_company_website[num])
        lc = lc.replace("'","")
        lc = lc.replace("[","")
        lc = lc.replace("]","")
        url2 = 'http://' + lc
        if num == 118 or num == 464:
            lt_nm.insert(num,list500_names[num])
            lt_time.insert(num,'n/a')
            nm.insert(num,num)
            lt_url.insert(num,list500_url[num])
        else:
            try:
                response2 = browser2.open(url2)
            except Exception:
                lt_time.insert(num,'n/a')
                lt_nm.insert(num,list500_names[num])
                nm.insert(num,num)
                print ("The site " + str(num)+ " has NOT been loaded!")
```

```
                continue
            start_time = time()
            myHTML2 = response2.read()
            end_time = time()
            response2.close()
            l_t = round(end_time-start_time, 3)
            #in order to be more readable we rounded the time
            loadt = str(l_t)
            lt_nm.insert(num,list500_names[num])
            lt_time.insert(num,loadt)
            nm.insert(num,num)
            lt_url.insert(num,list500_url[num])
            #print ("The site " + str(num) + " has been loaded!")
    print "The function is completed!"

loadtime (list_company_website,list500_names,list500_url)

d4 = {'company' : pd.Series(lt_nm, index=[nm]),
      'loading_time' : pd.Series(lt_time, index=[nm])}
loading_time = pd.DataFrame(d4)
```

Script 13: Types and Number of Images variables

```
p_p = []
p_d = []
p_jpg = []
p_jpeg = []
p_gif = []
p_tif = []
p_tiff = []
p_bmp = []
p_jpe = []
p_nm = []
p_tt =[]
nm = []
p_url = []

def images (list500_sites,list500_names,list500_url):
    from time import time
    start = time ()
    for num in range(len(list500_names)):
            myHTML = list500_sites[num]
            image = ['.png','.dib','.jpg','.jpeg',
                     '.bmp','.jpe','.gif','.tif','.tiff']
            totalnumber = 0
```

```
if myHTML == 0:
    p_nm.insert(num,list500_names[num])
    p_p.insert(num,'n/a')
    p_d.insert(num,'n/a')
    p_jpg.insert(num,'n/a')
    p_jpeg.insert(num,'n/a')
    p_gif.insert(num,'n/a')
    p_tif.insert(num,'n/a')
    p_tiff.insert(num,'n/a')
    p_bmp.insert(num,'n/a')
    p_jpe.insert(num,'n/a')
    p_tt.insert(num,'n/a')
    nm.insert(num,num)
    p_url.insert(num,list500_url[num])
else:
    for index in range(len(image)):
        x = image[index]
        photo = re.findall(x,myHTML)
        if x == '.png':
            p = str (len(photo))
        if x == '.dib':
            d = str (len(photo))
        if x == '.jpg':
            jpg = str (len(photo))
        if x == '.jpeg':
            jpeg = str (len(photo))
        if x == '.gif':
            gif = str (len(photo))
        if x == '.tif':
            tif = str (len(photo))
        if x == '.tiff':
            tiff = str (len(photo))
        if x == '.bmp':
            bmp = str (len(photo))
        if x == '.jpe':
            jpe = str (len(photo))
        totalnumber = len(photo) + totalnumber
    total = str (totalnumber)
    p_nm.insert(num,list500_names[num])
    p_p.insert(num,p)
    p_d.insert(num,d)
    p_jpg.insert(num,jpg)
    p_jpeg.insert(num,jpeg)
    p_gif.insert(num,gif)
    p_tif.insert(num,tif)
    p_tiff.insert(num,tiff)
```

```
                    p_bmp.insert(num,bmp)
                    p_jpe.insert(num,jpe)
                    p_tt.insert(num,total)
                    nm.insert(num,num)
                    p_url.insert(num,list500_url[num])
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

images (list500_sites,list500_names,list500_url)

d5 = {'company' : pd.Series(p_nm, index=[nm]),
      '.png' : pd.Series(p_p, index=[nm]),
      '.dib' : pd.Series(p_d, index=[nm]),
      '.jpg' : pd.Series(p_jpg, index=[nm]),
      '.jpeg' : pd.Series(p_jpeg, index=[nm]),
      '.bmp' : pd.Series(p_bmp, index=[nm]),
      '.jpe' : pd.Series(p_jpe, index=[nm]),
      '.gif' : pd.Series(p_gif, index=[nm]),
      '.tif' : pd.Series(p_tif, index=[nm]),
      '.tiff' : pd.Series(p_tiff, index=[nm]),
      'total images' : pd.Series(p_tt, index=[nm])}
images_types = pd.DataFrame(d5)
```

Script 14: Different image sizes per company function

```
nm = []
s_comp = []
s_dimensions = []
s_times = []
s_tt_dif_dim = []
ht = [] #list of different heights in each case
wt = [] #list of different widths in each case
h_w = [] # combinations of height and width
s_url = []

def find_dif_sizes (list_company_website,list500_names,list500_url):
    from time import time
    start = time ()
    for num in range(len(list500_names)):
            nm.insert(num,num)
            s_comp.insert(num,list500_names[num])
            s_url.insert(num,list500_url[num])
```

```
myHTML = list500_sites[num]
if myHTML == 0:
    s_dimensions.insert(num,0)
    s_times.insert(num,0)
else:
    soup = BeautifulSoup(myHTML, "lxml")
    s_dimensions_local = []
    s_times_local = []
    hw = 0
    for tag in soup.find_all('img'):
        h = tag.attrs.get('height', None)
        w = tag.attrs.get('width', None)
        if h != None:
            if w != None:
                ht.insert(hw,h)
                wt.insert(hw,w)
                hw = hw + 1
    hw2 = 0
    for l in range(len(ht)):
        h_w_c = ht[l] + 'x' + wt[l]
        h_w.insert(hw2,h_w_c)
        hw2 = hw2 + 1
    if h_w == []:
        nm.insert(num,num)
        s_comp.insert(num,list500_names[num])
        s_dimensions.insert(num,0)
        s_times.insert(num,0)
    if h_w != []:
        hw_unique = Counter(h_w)
        hw_unique2 = str(hw_unique)
        split1 = hw_unique2.split('{')
        a = split1[1]
        split2 = a.split('}')
        b = split2[0]
        split3 = b.split(',')
        finalsplit = []
        fs = []
        z = 0
        m = 1
        j = 0
        z1 = 0
        m1 = 1
        for numb in split3:
            oldstring = numb
            newstring = oldstring.replace("'", "")
            new = newstring.replace(" '","")
```

```
                        string = new.replace(" ","")
                        finalstring = string.split(':')
                        for xx in range(len(finalstring)):
                            ax = finalstring[xx]
                            if 'x' in ax:
                                s_dimensions_local.insert(z1,finalstring[xx])
                                z1 = z1 + 1
                            else:
                                s_times_local.insert(m1,finalstring[xx])
                                m1 = m1 + 1
                    s_dimensions.insert(num,s_dimensions_local)
                    s_times.insert(num,s_times_local)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

find_dif_sizes (list500_sites,list500_names,list500_url)
```

Script 15: Unique Image Sizes across all companies function

```
dif_size = []
un_size = []

def unique_dif_sizes (s_dimensions,list500_names):
    ds = 0
    for num in range(len(list500_names)):
        asw = s_dimensions[num]
        if asw != 0 :
            for s in range(len(asw)):
                ss = asw[s]
                dif_size.insert(ds,ss)
                ds = ds + 1
    dsu = 0
    for i in dif_size:
        if i not in un_size:
            un_size.insert(dsu,i)
            dsu = dsu + 1
    print(un_size)


unique_dif_sizes (s_dimensions,list500_names)
```

Script 16: The unique dimensions per company True or False function

```
t_f_s = []
ttf = []
nm = []
com = []

def dimensions_per_company (un_size,list500_names):
    from time import time
    start = time ()
    for num in range(len(list500_names)):
        s1a = s_dimensions[num]
        where = [] #empty list
        wh = 0
        haveornot = []
        for er in range (len(un_size)):
            if s1a != 0 :
                for sizea in s1a:
                    if sizea == un_size[er]:
                        where.insert(wh,str(er))
                        wh = wh +1
                        break
            if str(er) in where:
                haveornot.insert(er,True)
            else:
                haveornot.insert(er,False)

        t_f_s.insert(num,haveornot)
        ttf.insert(num,t_f_s)
        nm.insert(num,num)
        com.insert(num,list500_names[num])
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

dimensions_per_company (un_size,list500_names)
```

Script 17: Final data frame

```
d6 = {'company' : pd.Series(com, index=[nm])}
sizess = pd.DataFrame(d6)

def final_dimensions_dataframe (un_size,t_f_s,list500_names):
    from time import time
    start = time ()
```

```
    for q in range(len(un_size)):
        names = un_size[q]
        var = []
        for num in range(len(list500_names)):
            a = t_f_s[num]
            var.insert(num,a[q])
        sizess[names] = pd.Series(var, index=sizess.index)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

final_dimensions_dataframe (un_size,t_f_s,list500_names)
```

Script 18: Names corrections

```
list_company_name_new = []

for num in range (0,500):
    cn = list_company_name[num]
    cn = cn.replace(" ", "-")
    cn = cn.replace("&", "")
    cn = cn.replace("          ", "")
    cn = cn.replace(".", "-")
    cn = cn.replace("amp;", "")
    company = cn.lower()
    list_company_name_new.insert(num,cn)
```

Script 19: Fortune 500 - Pages Download

```
fortune_pages = []

def fortune500 (list_company_name_new):
    from time import time
    start = time ()
    for num3 in range (0,500):
        i = str (num3 +1)
        companyname =  list_company_name_new[num3]
        browser = urllib2.build_opener()
        browser.addheaders = [('User-agent', 'Mozilla/5.0')]
        site_fortune = "http://beta.fortune.com/fortune500/"+companyname+"-"+ i
        page_fortune = browser.open(site_fortune)
        html_fortune = page_fortune.read()
        #print("fortune page for company: ", list_company_name_new[num3], i)
```

```
        fortune_pages.insert(num3, html_fortune)
    end = time ()
    duration = round (end - start, 3)
    minutes = round (duration /60, 1)
    print 'The lists are ready in ', minutes, ' minutes'
    print 'The lists are ready in ', duration, ' seconds'

fortune500 (list_company_name_new)
```

Script 20: Fortune 500 - Initial variables

```
keyf =[]
per =[]
rev_dol = []
rev_per = []
prof_dol = []
prof_per = []
assets_dol = []
assets_per = []
tse_dol = []
tse_per = []
mar_dol = []
mar_per = []
market = []
nm = []
ln = []
urln = []
empty = []
```

Script 21: Fortune 500 - Pages Download

```
def fortune_metrics (list_company_name,list_company_website):
    x = 0
    for n in range (0,500):
        nm.insert(x,x)
        ln.insert(x,list_company_name[n])
        urln.insert(x,list_company_website[n])
        files = fortune_pages[x]
        soup = BeautifulSoup(files,"lxml")
        o=0
        for row in soup.html.body.findAll('tbody'):
            keyf.insert(o,row)
            o=o+1
        keyfin = keyf[0]
        data = keyfin.findAll('td')
```

```
two = str(data[1]) # revenue
revdol= re.findall('>\$(.+?)</td>',two)
if revdol[0] != empty:
    w = revdol[0]
    a = w.replace("[", "")
    r = a.replace("]","")
    rev_dol.insert(x,r)
else:
    rev_dol.insert(x,'not available')
tria = str(data[2])# revenue in percentage
revper= re.findall('>(.+?)%</td>',tria)
if revper != empty:
    w = revper[0]
    a = w.replace("[", "")
    r1 = a.replace("]","")
    rev_per.insert(x,r1)
else:
    rev_per.insert(x,'not available')
eight = str(data[7]) #assets in dollars
assetsdol= re.findall('>\$(.+?)</td>',eight)
if assetsdol != empty:
    w = assetsdol[0]
    a = w.replace("[", "")
    ass = a.replace("]","")
    assets_dol.insert(x,ass)
else:
    assets_dol.insert(x,'not available')
ten = str(data[9]) #Total Stockholder Equity
eleven = str(data[10])
tsedol= re.findall('>\$(.+?)</td>',eleven)
if tsedol != empty:
    w = tsedol[0]
    a = w.replace("[", "")
    ts = a.replace("]","")
    tse_dol.insert(x,ts)
else:
    tse_dol.insert(x,'not available')
thirteen = str(data[12]) # market value
fourteen = str(data[13])
mardol= re.findall('>\$(.+?)</td>',fourteen)
if mardol != empty:
    w = mardol[0]
    a = w.replace("[", "")
    mar = a.replace("]","")
    mar_dol.insert(x,mar)
else:
```

```
            mar_dol.insert(x,'not available')
        x = x + 1
    print "The function is complete!"

fortune_metrics (list_company_name,list_company_website)
```

Script 22: Fortune 500 - Data frame

```
d9 = {'company' : pd.Series(ln, index=[nm]),
      'Revenues $' : pd.Series(rev_dol, index=[nm]),
      'Revenues %' : pd.Series(rev_per, index=[nm]),
      'Assets $' : pd.Series(assets_dol, index=[nm]),
      'Total Stockholder Equity $' : pd.Series(tse_dol, index=[nm]),
      'Market value $' : pd.Series(mar_dol, index=[nm])}
fort500 = pd.DataFrame(d9)
```

Script 23: Final Data Frame and csv file

```
result = pd.merge(fort500, html_val, how='inner',
 on=['company', 'company'])
result2 = pd.merge(social_media, fre, how='inner',
 on=['company', 'company'])
result3 = pd.merge(sites_links, sizess, how='inner',
on=['company', 'company'])
result4 = pd.merge(images_types, loading_time, how='inner',
 on=['company', 'company'])
result5 = pd.merge(result,result2 , how='inner',
 on=['company', 'company'])
result6 = pd.merge(result3, result4, how='inner',
 on=['company', 'company'])
final = pd.merge(result5, result6, how='inner',
 on=['company', 'company'])
final.head(3)

final.to_csv('total_500_new.csv', sep=';')
```

A.3 Appendix C: R Scripts

A.3.1 Data cleansing

```
#we upload the dataset
total_500 <- read.csv("~/GitHub/thesis_msc_business_analytics/
Python/total_500_new.csv", sep=";", na.strings="n/a")
#we see how many observations and how many variables we have
dim(total_500)
#We create a subset to make some changes to the data
total_500_sub <- total_500
#Change the decimal point for the 4 variables
total_500_sub$Assets.. <- gsub(",", ".",
 total_500_sub$Assets.. )
total_500_sub$Market.value.. <- gsub(",", ".",
 total_500_sub$Market.value.. )
total_500_sub$Revenues.. <- gsub(",", ".",
 total_500_sub$Revenues.. )
total_500_sub$Total.Stockholder.Equity.. <- gsub(",", ".",
 total_500_sub$Total.Stockholder.Equity.. )
#Make the variables numeric
for(i in 1:18){
 total_500_sub[,i] <- as.numeric(total_500_sub[,i])}
for(i in 20:730){
 total_500_sub[,i] <- as.numeric(total_500_sub[,i])}
#We omit the nas from the analysis
total_500_final <- na.omit(total_500_sub)
#We rename variable X as Ranking
colnames(total_500_final)[1] <- "Ranking"
#Change the names of some variables to be more easily readable
colnames(total_500_final)[2] <- "Assets"
colnames(total_500_final)[3] <- "Market_Value"
colnames(total_500_final)[4] <- "Revenues"
colnames(total_500_final)[6] <- "Total_SH_Equity"
#Delete the variables we will not need
total_500_final$Revenues...1 <- NULL #Revenues %
total_500_final$company <- NULL #company name
total_500_final$url<- NULL # company url
#we upload the libraries beneath that we will use in the analysis
library(ggplot2)
library(reshape2)
library(DAAG)
#Final number of observation and variables we will use
dim(total_500_final)
```

A.3.2 Variable analysis and correlation

A.3.2.1 Fortune variables correlation

```
#we first see the summary of the Fortune variables and then we create their histo
#good grasp of how they are distributed
ggplot(data=total_500_final,aes(x=Revenues))
+geom_histogram(binwidth=50, colour = "green", fill ="darkgreen")
ggplot(data=total_500_final,aes(x=Assets))
+geom_histogram(binwidth=100, colour = "red", fill ="darkred")
ggplot(data=total_500_final,aes(x=Market_Value))
+geom_histogram(binwidth=100, colour = "blue", fill ="darkblue")
ggplot(data=total_500_final,aes(x=Total_SH_Equity))
+geom_histogram(binwidth=100, colour = "purple", fill ="pink")

#We make plots to see how the variables we got from Fortune 500 are related with
ggplot(total_500_final, aes(Assets,Ranking))
+ geom_point(colour = "red")
ggplot(total_500_final, aes(Market_Value, Ranking))
+ geom_point(colour = "blue")
ggplot(total_500_final, aes(Total_SH_Equity, Ranking))
 + geom_point(colour = "purple")
ggplot(total_500_final, aes(Revenues, Ranking))
 + geom_point(colour = "green")
#We can see that the Ranking has a linear relationship with the Revenues so we u
#In order to have a more clear look we also create a correlation diagram
total_500_fortune <- total_500_final[,c(1:5)]
library(corrplot)
library(caret)
sm <- cor(total_500_fortune)
sm
corrplot(cor(total_500_fortune),method="number")
#From this plot we understand that the Ranking and the Revenues have very high c
```

A.3.2.2 Social media analysis

```
#Facebook
social_media_facebook <- round(table(total_500_final$facebook)/408,3)
social_media_facebook
slicelable <- c(paste(35.3,"% no"),paste(64.7,"% yes"))
pie(social_media_facebook,label = slicelable,
main="Share of companies with Facebook",
col=rainbow(length(social_media_facebook)))
ggplot(total_500_final, aes(Revenues, facebook))
+ geom_point(size=3, colour = "darkblue")

#Twitter
```

```
social_media_twitter <- round(table(total_500_final$twitter)/408,3)
social_media_twitter
slicelable <- c(paste(31.4,"% no"),paste(68.6,"% yes"))
pie(social_media_twitter,label = slicelable,
main="Share of companies with Twitter",
col=rainbow(length(social_media_twitter)))
ggplot(total_500_final, aes(Revenues, twitter))
+ geom_point(size=3, colour = "darkgreen")

#Instagram
social_media_instagram <- round(table(total_500_final$instagram)/408,3)
social_media_instagram
slicelable <- c(paste(77.7,"% no"),paste(22.3,"% yes"))
pie(social_media_instagram,label = slicelable,
main="Share of companies with Instagram",
col=rainbow(length(social_media_instagram)))
ggplot(total_500_final, aes(Revenues, instagram))
+ geom_point(size=3, colour = "pink")

#Pinterest
social_media_pinterest <- round(table(total_500_final$pinterest)/408,3)
social_media_pinterest
slicelable <- c(paste(90.2,"% no"),paste(9.8,"% yes"))
pie(social_media_pinterest,label = slicelable,
main="Share of companies with Pinterest",
col=rainbow(length(social_media_pinterest)))
ggplot(total_500_final, aes(Revenues, pinterest))
+ geom_point(size=3, colour = "darkred")

#Youtube
social_media_youtube <- round(table(total_500_final$youtube)/408,3)
social_media_youtube
slicelable <- c(paste(41.7,"% no"),paste(58.3,"% yes"))
pie(social_media_youtube,label = slicelable,
main="Share of companies with Youtube",
col=rainbow(length(social_media_youtube)))
ggplot(total_500_final, aes(Revenues, youtube))
+ geom_point(size=3, colour = "red")

#LinkedIn
social_media_linkedin <- round(table(total_500_final$linkedin)/408,3)
social_media_linkedin
slicelable <- c(paste(42.9,"% no"),paste(57.1,"% yes"))
pie(social_media_linkedin,label = slicelable,
main="Share of companies with Linkedin",
col=rainbow(length(social_media_linkedin)))
```

```
ggplot(total_500_final, aes(Revenues, linkedin))
+ geom_point(size=3, colour = "blue")

#And we can also see for correlations
total_500_social_media <- total_500_final[,c(4,10:15)]
library(corrplot)
library(caret)
sm <- cor(total_500_social_media)
sm
corrplot(cor(total_500_social_media),method="number")
```

A.3.2.3 Links analysis

```
par(mfrow=c(1,1))
library(ggplot2)
ggplot(data=total_500_final,aes(x=total.links))
+geom_histogram(binwidth=50, colour = "darkblue", fill ="blue")
ggplot(total_500_final, aes(Revenues, total.links))
+ geom_point(size=3, colour = "darkblue")
ggplot(data=total_500_final,aes(x=external))
+geom_histogram(binwidth=50, colour = "darkred", fill ="red")
ggplot(total_500_final, aes(Revenues, external))
+ geom_point(size=3, colour = "darkred")
ggplot(data=total_500_final,aes(x=internal))
+geom_histogram(binwidth=50, colour = "darkgreen", fill ="green")
ggplot(total_500_final, aes(Revenues, internal))
+ geom_point(size=3, colour = "darkgreen")

#And we can also see for correlations
total_500_links <- total_500_final[,c(4,21:23)]
library(corrplot)
library(caret)
tl <- cor(total_500_links)
tl
corrplot(cor(total_500_links),method="number")
```

A.3.2.4 Loading time analysis

```
ggplot(data=total_500_final,aes(x=loading.time))
+geom_histogram(binwidth=1, colour = "pink", fill ="purple")
ggplot(total_500_final, aes(Revenues, loading.time))
+ geom_point(size=3, colour = "purple")
```

A.3.2.5 Content analysis

```
```{r}
##############################################################################
```
```

```
#Now we will see the total words, the unique words and the sentences how are dis
ggplot(data=total_500_final,aes(x=Sentences))+geom_histogram(binwidth=50, colour
ggplot(total_500_final, aes(Revenues, Sentences)) + geom_point(size=3, colour =
##########################
ggplot(data=total_500_final,aes(x=Unique.words))+geom_histogram(binwidth=50, col
ggplot(total_500_final, aes(Revenues, Unique.words)) + geom_point(size=3, colour
##########################
ggplot(data=total_500_final,aes(x=Words))+geom_histogram(binwidth=50, colour = "
ggplot(total_500_final, aes(Revenues, Words)) + geom_point(size=3, colour = "pur
##############################
```
```
```{r}
#And we can also see for correlations
total_500_lt_w <- total_500_final[,c(4,18:20,727)]
library(corrplot)
library(caret)
tl <- cor(total_500_lt_w)
tl
corrplot(cor(total_500_lt_w),method="number")
```

```{r}
################################

#Next we will check the Flesh Measure alone and in relationship with revenues
ggplot(data=total_500_final,aes(x=Flesh_Mesaure))+geom_histogram(binwidth=50, co
ggplot(total_500_final, aes(Revenues, Flesh_Mesaure)) + geom_point(size=3, colou
############################
```

```{r}
total_500_final$Readability <- gsub("Very easy", "01_VE", total_500_final$Readab
total_500_final$Readability <- gsub("Easy", "02_E", total_500_final$Readability
total_500_final$Readability <- gsub("Fairly easy", "03_FE", total_500_final$Read
total_500_final$Readability <- gsub("Standard", "04_St", total_500_final$Readab
total_500_final$Readability <- gsub("Fairly difficult", "05_FD", total_500_final
total_500_final$Readability <- gsub("Difficult", "06_D", total_500_final$Readab
total_500_final$Readability <- gsub("Very Confusing", "07_VC", total_500_final$R
barplot(table(total_500_final$Readability),col ="dark red")
```

```{r}
total_500_final$Readability <- gsub("01_VE","1", total_500_final$Readability )
total_500_final$Readability <- gsub("02_E", "2", total_500_final$Readability )
total_500_final$Readability <- gsub("03_FE", "3", total_500_final$Readability )
total_500_final$Readability <- gsub("04_St", "4", total_500_final$Readability )
```
```

```
total_500_final$Readability <- gsub("05_FD", "5", total_500_final$Readability )
total_500_final$Readability <- gsub("06_D", "6" ,total_500_final$Readability )
total_500_final$Readability <- gsub("07_VC", "7",total_500_final$Readability )
total_500_final$Readability <- as.numeric(total_500_final$Readability )
ggplot(data=total_500_final,aes(x=Readability))+geom_bar(binwidth=1, colour = "d
ggplot(total_500_final, aes(Revenues, Readability)) + geom_point(size=3, colour =
#And we can also see for correlations
total_500_r <- total_500_final[,c(4,16,17)]
library(corrplot)
library(caret)
tl <- cor(total_500_r)
tl
corrplot(cor(total_500_r),method="number")
```

```
```

A.3.2.6 HTML validation analysis

```
```{r}
#############################################################################
#Now we will see the number of errors and warnings alone and in relationship wit
ggplot(data=total_500_final,aes(x=number_of_errors))+geom_histogram(binwidth=50,
ggplot(total_500_final, aes(Revenues, number_of_errors)) + geom_point(size=3, co
ggplot(data=total_500_final,aes(x=number_of_warning))+geom_histogram(binwidth=20
ggplot(total_500_final, aes(Revenues, number_of_warning)) + geom_point(size=3, c
#############################################################################
#############################################################################
#Now we will see the non.document.error and the page not opened variables alone
ggplot(data=total_500_final,aes(x=non.document.error))+geom_histogram(binwidth=1
ggplot(total_500_final, aes(Revenues, non.document.error)) + geom_point(size=1,
ggplot(data=total_500_final,aes(x=The_page_opened))+geom_histogram(binwidth=1, c
ggplot(total_500_final, aes(Revenues, The_page_opened)) + geom_point(size=3, col
#In the page not opened we can see that the variable has only the price 1 that m
#############################################################################
#And we can also see for correlations
total_500_html <- total_500_final[,c(4,7:9)]
library(corrplot)
library(caret)
tl <- cor(total_500_html)
tl
corrplot(cor(total_500_html),method="number")
```
```

A.3.2.7 Number, types and image sizes analysis

```
```{r}
#Now we will see the total images alone and in relationship with the revenues
```
```

```
ggplot(data=total_500_final,aes(x=total.images))+geom_histogram(binwidth=100, co
```

```{r}
ggplot(total_500_final, aes(Revenues, total.images)) + geom_point(size=3, colour
```

```{r}
##############################################################################
#We will see now the frequency of image types that is being used

par(mfrow=c(1,1))
k = c(717:725)
for(i in 1:9){
  a <- k[i]
  image_type<- round(table(total_500_final[,a])/408,3)
  barplot(image_type,xlab=names(total_500_final)[a],ylab = "Shares of images per
```

```{r}
#It is obvious that the most common images type are .jpg, gif and .png
#We will check now the types in relationship with the revenues
ggplot(total_500_final, aes(Revenues, .bmp)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .dib)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .gif)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .jpe)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .jpeg)) + geom_point(size=3, colour = "dar
ggplot(total_500_final, aes(Revenues, .jpg)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .png)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .tif)) + geom_point(size=3, colour = "dark
ggplot(total_500_final, aes(Revenues, .tiff)) + geom_point(size=3, colour = "dar
```

```{r}
#And we can also see for correlations
total_500_im<- total_500_final[,c(4,717:726)]
library(corrplot)
library(caret)
tl <- cor(total_500_im)
tl
corrplot(cor(total_500_im),method="number")
```

```{r}
#We will see now the frequency of image sizes that is being used
```

```
k = c()
#Check for sizes that are half and half divided in existing and not
for(i in 24:716){
  image_size<- round(table(total_500_final[,i]))
  if ((image_size[[1]]==408)==TRUE){
    k <- union(k, c(i))
  }}
######################
#Number 24 is all onw price so we want use it
names(total_500_final)[24]
total_500_final$X144x144 <- NULL
```

```{r}
false_not_existing = c()
#Check for sizes that are less than half divided in existing and not
for(i in 24:715){
  image_size<- round(table(total_500_final[,i]))
  if ((image_size[[2]]<204)==TRUE){
    false_not_existing <- union(false_not_existing, c(i))
  }}
########################
```

```{r}
#Now we will take the sizes that exist in less than half the instances and check
par(mfrow=c(3,3))
for(i in 1:416){
  a = false_not_existing[i]
  plot(total_500_final[,a],total_500_final$Revenues)
  image_size<- round(table(total_500_final[,a]))
  barplot(image_size,xlab=names(total_500_final)[a],ylab = "Has or not the size"
```

```{r}
true_existing = c()
#Check for sizes that are more than half divided in existing and not
for(i in 24:715){
  image_size<- round(table(total_500_final[,i]))
  if ((image_size[[2]]>204)==TRUE){
    true_existing <- union(true_existing, c(i))
  }}
```

```{r}
#Now we will take the sizes that exist in more than half the instances and check
```

```
par(mfrow=c(3,3))
for(i in 1:276){
  a = true_existing[i]
  image_size<- round(table(total_500_final[,a]))
  plot(total_500_final[,a],total_500_final$Revenues)
  barplot(image_size,xlab=names(total_500_final)[a],ylab = "Has or not the size"
```

```{r}
#By checking the above plots we can see that the 24 first sizes do appear to hav
par(mfrow=c(3,3))
keep = c()
for(i in 1:24){
  a = true_existing[i]
  keep = union (keep, c(a))}
keep
#As we can see they are the variables from 24 to 47 and these are the only sizes
total_500_final <- total_500_final[,-c(48:715)]
```

A.3.3 Data manipulation

```{r}
#Also we remove the other Fortune 500 variables since they will interfer in the
total_500_final$Market_Value <- NULL
total_500_final$Assets <- NULL
total_500_final$Ranking <- NULL
total_500_final$Total_SH_Equity <- NULL
total_500_final$The_page_opened <- NULL
```

```{R}
summary(total_500_final)
```

```{r}
names(total_500_final)
```

```{R}
total_500_final$X15x12<- gsub("1","0", total_500_final$X15x12)
total_500_final$X15x12 <- gsub("2", "1", total_500_final$X15x12 )

total_500_final$X60x60<- gsub("1","0", total_500_final$X60x60)
total_500_final$X60x60 <- gsub("2", "1", total_500_final$X60x60 )

total_500_final$X15x75<- gsub("1","0", total_500_final$X15x75)
total_500_final$X15x75 <- gsub("2", "1", total_500_final$X15x75 )
```

```
total_500_final$X28x221<- gsub("1","0", total_500_final$X28x221)
total_500_final$X28x221 <- gsub("2", "1", total_500_final$X28x221 )

total_500_final$X41x192 <- gsub("1","0", total_500_final$X41x192 )
total_500_final$X41x192 <- gsub("2", "1", total_500_final$X41x192 )

total_500_final$X300x993 <- gsub("1","0", total_500_final$X300x993 )
total_500_final$X300x993 <- gsub("2", "1", total_500_final$X300x993 )

total_500_final$X160x233 <- gsub("1","0", total_500_final$X160x233 )
total_500_final$X160x233 <- gsub("2", "1", total_500_final$X160x233 )

total_500_final$X29x29 <- gsub("1","0", total_500_final$X29x29 )
total_500_final$X29x29 <- gsub("2", "1", total_500_final$X29x29 )

total_500_final$X300pxx1500px <- gsub("1","0", total_500_final$X300pxx1500px )
total_500_final$X300pxx1500px <- gsub("2", "1", total_500_final$X300pxx1500px )

total_500_final$X200pxx200px<- gsub("1","0", total_500_final$X200pxx200px )
total_500_final$X200pxx200px <- gsub("2", "1", total_500_final$X200pxx200px )

total_500_final$X292pxx292px <- gsub("1","0", total_500_final$X292pxx292px )
total_500_final$X292pxx292px <- gsub("2", "1", total_500_final$X292pxx292px )

total_500_final$X400x300 <- gsub("1","0", total_500_final$X400x300 )
total_500_final$X400x300 <- gsub("2", "1", total_500_final$X400x300 )

total_500_final$X115x223 <- gsub("1","0", total_500_final$X115x223 )
total_500_final$X115x223 <- gsub("2", "1", total_500_final$X115x223 )

total_500_final$X1279pxx984px <- gsub("1","0", total_500_final$X1279pxx984px )
total_500_final$X1279pxx984px<- gsub("2", "1", total_500_final$X1279pxx984px )

total_500_final$X8x15 <- gsub("1","0", total_500_final$X8x15 )
total_500_final$X8x15 <- gsub("2", "1", total_500_final$X8x15 )

total_500_final$X44x556 <- gsub("1","0", total_500_final$X44x556 )
total_500_final$X44x556 <- gsub("2", "1", total_500_final$X44x556 )

total_500_final$X1x1 <- gsub("1","0", total_500_final$X1x1 )
total_500_final$X1x1 <- gsub("2", "1", total_500_final$X1x1 )

total_500_final$autox100. <- gsub("1","0", total_500_final$autox100. )
total_500_final$autox100. <- gsub("2", "1", total_500_final$autox100. )
colnames(total_500_final)[24] <- "X100x100"
```

```
total_500_final$X800x1200 <- gsub("1","0", total_500_final$X800x1200 )
total_500_final$X800x1200 <- gsub("2", "1", total_500_final$X800x1200 )

total_500_final$X24pxx133px <- gsub("1","0", total_500_final$X24pxx133px )
total_500_final$X24pxx133px <- gsub("2", "1", total_500_final$X24pxx133px )

total_500_final$X21pxx173px <- gsub("1","0", total_500_final$X21pxx173px )
total_500_final$X21pxx173px <- gsub("2", "1", total_500_final$X21pxx173px )

total_500_final$X46x214 <- gsub("1","0", total_500_final$X46x214)
total_500_final$X46x214 <- gsub("2", "1", total_500_final$X46x214 )

total_500_final$X49x49 <- gsub("1","0", total_500_final$X49x49)
total_500_final$X49x49 <- gsub("2", "1", total_500_final$X49x49 )

total_500_final$X50x45 <- gsub("1","0", total_500_final$X50x45)
total_500_final$X50x45 <- gsub("2", "1", total_500_final$X50x45 )

```

```{r}
for(i in 19:42){
 total_500_final[,i] <- as.numeric(total_500_final[,i])}
```

A.3.4 Regression models

```{r}
#We split the set to training and test set
library(caret)
set.seed(20)
sampling_vector <- createDataPartition(total_500_final$Revenues, p = 0.85, list =
total_500_final_train <- total_500_final[sampling_vector,]
total_500_final_test <- total_500_final[-sampling_vector,]
```

```{r}
#We will try to create a regression model to see which of the variables of the u
#We create the empty lm model
model_null = lm(Revenues~1,data=total_500_final_train)
summary(model_null)
```

```{r}
################################################################################
#LASSO and Logistic Regression models
```

```
library(glmnet)
#We create a full model for the variable Ranking
full <- lm(Revenues~.,data=total_500_final_train)
summary(full)
```

```{r}
x <- model.matrix(full) [,-1]
dim(x)
lasso <- glmnet (x, total_500_final_train$Revenues)
par(mfrow=c(1,1),no.readonly = TRUE)
plot(lasso, xvar='lambda', label=T)
```
```{r}
lassob <- cv.glmnet(x,total_500_final_train$Revenues)
lassob$lambda.min
lassob$lambda.1se
```


```{r}
plot(lassob)
```

```{r}
#We see the coefficients for lamda min
blasso <- coef(lassob, s="lambda.min")
blasso
dim(blasso)
zblasso <- blasso[-1] * apply(x,2,sd)
zbolt <- coef (full) [-1] * apply (x,2,sd)
azbolt <- abs(zbolt)
sum(azbolt)
#since the sum is NA that means we have to substract some variables
# in order to find which variables to substract we run the coefficients and we s
coef(full)
```

```{r}
#Now we create a new model with only the variables with coef different from NA
full_2 <- lm(Revenues~. - total.images - total.links - X1x1 - X21pxx173px - X46x
X300x993 - X41x192 - X28x221 - X15x12,data=total_500_final_train)
summary(full_2)
```

```{r}
```

```
x <- model.matrix(full_2) [,-c(18,22,28,26,27,34,32,33,41,37,38,39,40,52)]
dim(x)
lasso <- glmnet (x, total_500_final_train$Revenues)
```

```{r}
plot(lasso, xvar='lambda', label=T)
```

```{r}
lassob <- cv.glmnet(x,total_500_final_train$Revenues)
lassob$lambda.min
lassob$lambda.1se
```

```{r}
plot(lassob)
```

```{r}
#coefiecinets for lammda min
blasso <- coef(lassob, s="lambda.min")
blasso
dim(blasso)
zblasso <- blasso[-1] * apply(x,2,sd)
zbolt <- coef (full_2) [-1] * apply (x,2,sd)
azbolt <- abs(zbolt)
sum(azbolt)
s <- sum(abs(zblasso))/sum(abs(azbolt))
s
```

```{r}
full_3 <- lm(Revenues~1 +X8x15  +X44x556 +X800x1200 +X24pxx133px +X50x45 +X400x3
+.bmp +.dib ,data=total_500_final_train)
summary(full_3)
ad_r_sq_f3 <- summary(full_3)$adj.r.squared
aic_f3 <- AIC(full_3)
```

```{r}
plot(full_3,which=1:3)
```

```{r}
##############################################
```

```
blassob <- coef(lassob, s="lambda.1se")
blassob
zblassob <- blassob[-1] * apply(x,2,sd)
zboltb <- coef (full_2) [-1] * apply (x,2,sd)
s <- sum(abs(zblassob))/sum(abs(zboltb))
s
#The model based on the lasso method by taking the lambda.1se is the null model
```

```{r}
full_4 <- lm(Revenues~1 +X8x15  +X44x556 +X800x1200 +X24pxx133px +X50x45 +X400x3
summary(full_4)
ad_r_sq_f4 <- summary(full_4)$adj.r.squared
aic_f4 <- AIC(full_4)
```

```{r}
plot(full_4,which=1:3)
```

```{r}
##################################################
#We use the "both" method to compare the full_3 model with the null model to see
model_a <- step(model_null, scope = list(lower = model_null, upper=full_2), dire
summary(model_a)
ad_r_sq_ma <- summary(model_a)$adj.r.squared
aic_ma <- AIC(model_a)
```

```{r}
plot(model_a,which=1:3)
```

```{r}
################
#We compare the Adjusted R squares of the models and also the AIC of the models
ad_r_sq_f3
ad_r_sq_f4
ad_r_sq_ma
#The best Adkusted R square is the one in full 3 (the closer to 1 the better)
aic_f3
aic_f4
aic_ma
#The best AIC and the best Adjusted R square is for model ma
```

```r
##########################################################################################
par(mfrow=c(2,2))
Actual_Revenues<- total_500_final_test$Revenues
plot (Actual_Revenues, col = "blue")
#########################################
predictions_ma <- predict(model_a,total_500_final_test)
plot (predictions_ma, col = "Red",main = "Model_a")
#########################################
predictions_full3 <- predict(full_3,total_500_final_test)
plot (predictions_full3 , col = "Red",main = "Full_3_model")
#########################################
predictions_full4 <- predict(full_4,total_500_final_test)
plot (predictions_full4 , col = "Red",main = "Full_4_model")
#########################################
#From the plots above we can see that the actual Revenues have a more smooth way
#The prediction model that is more smooth is the model a which has as we said be
```

```r
par(mfrow=c(1,1))
total_500_final_reg <- total_500_final_train[,c(1,6,12,20,21,25,30,42,43,47,53)]
corrplot(cor(total_500_final_reg),method="number")
#We can see here that the variable x8x15 has a very high correlation with the va
```

```r
#So we can try creating a new model excluding the 2 variables that are correlate
full_5 <- lm(Revenues~1 +X60x60 +X44x556 +X400x300 + .bmp +loading.time + .jpeg +
summary(full_5)
adj_r_square_full5 <- summary(full_5)$adj.r.squared
aic_full5 <- AIC(full_5)
```

```r
#We create the 2 basic plots so as to be able to explain the regression model
plot(full_5,which=1:3)
```

```r
ad_r_sq_ma
adj_r_square_full5
aic_ma
aic_full5
#The adjusted R square and the aic are a little worse than before
```

A.3.5 Comparisons and other methods

```
#Clustering
#Based on those results we will try to cluster the companies based on the result
set.seed(220)
fortuneCluster <- kmeans(total_500_final_reg[, 1:11], 3, iter.max = 100,nstart =
cluster <- table(fortuneCluster$cluster)
fortuneCluster$cluster <- as.factor(fortuneCluster$cluster)
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, loading.time, color = fortuneCluster$c
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, Readability, color = fortuneCluster$cl
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, instagram, color = fortuneCluster$clus
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, .bmp, color = fortuneCluster$cluster))
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, .jpeg, color = fortuneCluster$cluster)
```
```{r}
ggplot(total_500_final_reg, aes(Revenues, X60x60, color = fortuneCluster$cluster
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, X44x556, color = fortuneCluster$cluste
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, X400x300, color = fortuneCluster$clust
```
```{r}
ggplot(total_500_final_reg, aes(Revenues, X8x15, color = fortuneCluster$cluster)
```

```{r}
ggplot(total_500_final_reg, aes(Revenues, X24pxx133px, color = fortuneCluster$cl
```

```
```

```{r}
#From the clustering we can see that the variables do indeed devide the most hig
summary(model_a)
```

```{r}
#We can see from the model that the basic variable that effect a companys rankin
#We will try to make a model that we will not take into consideration this varia
full_6 <- lm(Revenues~1 +X44x556 +X400x300 + .bmp +loading.time + .jpeg + Readal
summary(full_6)
adj_r_square_full6 <- summary(full_6)$adj.r.squared
aic_full6 <- AIC(full_6)
```

```{r}
#We create the 2 basic plots so as to be able to explain the regression model
plot(full_6,which=1:3)
```

```{r}
predictions_ma <- predict(model_a,total_500_final_test)
Actual_Revenues<- total_500_final_test$Revenues
```

```{r}
par(mfrow=c(2,2))
plot (Actual_Revenues, col = "blue")
plot (predictions_ma, col = "Red",main = "Model A")
########################################
predictions_full_6 <- predict(full_6,total_500_final_test)
plot (predictions_full_6, col = "Red",main = "Full_6 model")
##########################################################
```

```{r}
#We can see that here the prediction of the new model is not as good as the prev
summary(model_a)
```
```
```