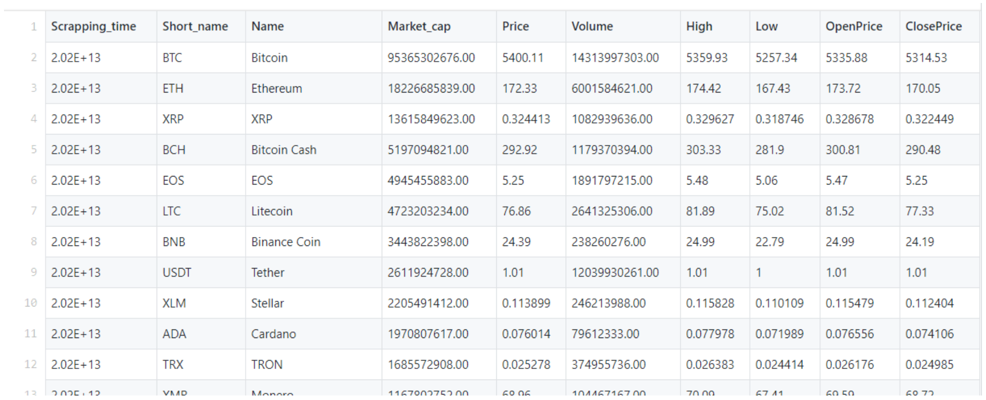
ECON900_ps1

Shih Yu Sung

4/23/2019

■ Web scrapping part:

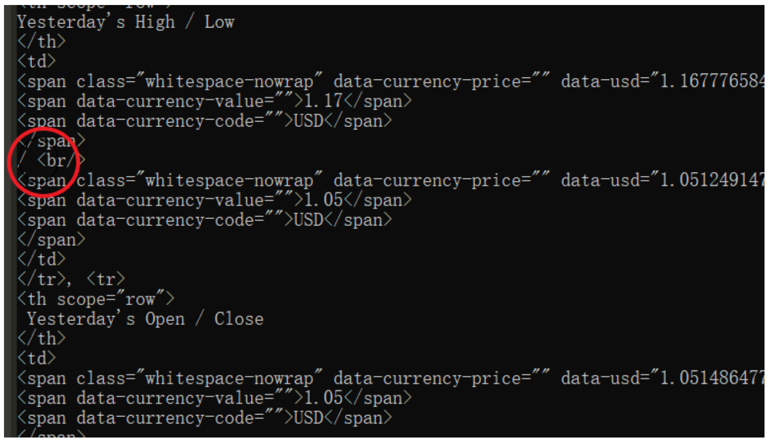
In this part, I scrape the data all the currencies from the coinmarketcap.com and collect historical data of opening price, closing price, day high, day low, volume, market Cap, price, currency short name, and currency name.



Scrapping_time	Short_name	Name	Market_cap	Price	Volume	High	Low	OpenPrice	ClosePrice
2.02E+13	BTC	Bitcoin	95365302676.00	5400.11	14313997303.00	5359.93	5257.34	5335.88	5314.53
2.02E+13	ETH	Ethereum	18226685839.00	172.33	6001584621.00	174.42	167.43	173.72	170.05
2.02E+13	XRP	XRP	13615849623.00	0.324413	1082939636.00	0.329627	0.318746	0.328678	0.322449
2.02E+13	BCH	Bitcoin Cash	5197094821.00	292.92	1179370394.00	303.33	281.9	300.81	290.48
2.02E+13	EOS	EOS	4945455883.00	5.25	1891797215.00	5.48	5.06	5.47	5.25
2.02E+13	LTC	Litecoin	4723203234.00	76.86	2641325306.00	81.89	75.02	81.52	77.33
2.02E+13	BNB	Binance Coin	3443822398.00	24.39	238260276.00	24.99	22.79	24.99	24.19
2.02E+13	USDT	Tether	2611924728.00	1.01	12039930261.00	1.01	1	1.01	1.01
2.02E+13	XLM	Stellar	2205491412.00	0.113899	246213988.00	0.115828	0.110109	0.115479	0.112404
2.02E+13	ADA	Cardano	1970807617.00	0.076014	79612333.00	0.077978	0.071989	0.076556	0.074106
2.02E+13	TRX	TRON	1685572908.00	0.025278	374955736.00	0.026383	0.024414	0.026176	0.024985
2.02E+13	XMR	Monero	1167802752.00	68.96	104467167.00	70.09	67.41	69.59	68.72

A glance of the data I scraped.

Since the historical data of each currency is hidden in each currency website. The method I use is to request the main page first, which contains the list of all the currency name, currency short name, and the currency link, and then I use links to directly scrape each currency website online. Another method is to request each currency website and download the HTML files and then parse the historical data of each currency locally. But I didn't use this method because in that way I would have two thousand more saved HTML files after I request, which is too much.



```
<th scope="row">
Yesterday's High / Low
</th>
<td>
<span class="whitespace-nowrap" data-currency-price="" data-usd="1.167776584
<span data-currency-value="">1.17</span>
<span data-currency-code="">USD</span>
</span>
/ <br/>
<span class="whitespace-nowrap" data-currency-price="" data-usd="1.051249147
<span data-currency-value="">1.05</span>
<span data-currency-code="">USD</span>
</span>
</td>
</tr>, <tr>
<th scope="row">
 Yesterday's Open / Close
</th>
<td>
<span class="whitespace-nowrap" data-currency-price="" data-usd="1.051486477
<span data-currency-value="">1.05</span>
<span data-currency-code="">USD</span>
</span>
```

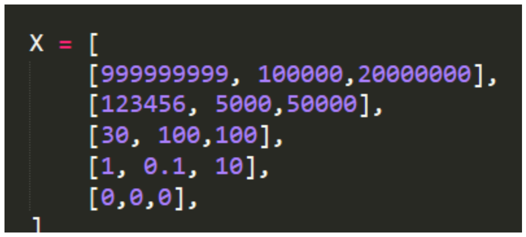
Here is an example of the bug I encountered when I parsed the website:
A simple blank before the <br> can cause the whole codes to break down.

■ Machine Learning Part:

In this part, I use four different models to run machine learning. The first two models are Ordinal Least Squares and K Neighbors Classifier, which are both supervised learning. The last two models are Train Test Split using Sklearn and Kfold using Sklearn, too. Both are unsupervised learning.

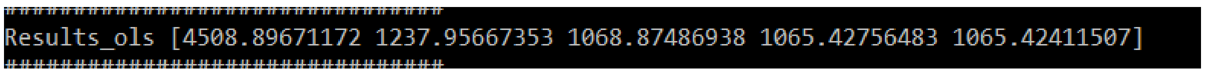
1. Ordinal Least Squares:

   In this part, I use the dataset I got from the web scrapping part. When I run the OLS model, I want to know the relation of the rank of each currency and the market cap, the price, and the volume of each currency. That is, the dependent variable Y is rank, the independent X are the market cap, the price, and the volume. I want to know if it can predict a currency's rank given random values of market cap, price, and volume. Note that I didn't scrape the rank variable. But since the order of each currency is fixed, I just create a new variable rank based on the order.



```
X = [
    [999999999, 100000,20000000],
    [123456, 5000,50000],
    [30, 100,100],
    [1, 0.1, 10],
    [0,0,0],
]
```

   Here is a list of 5 different sets of variables. The variables represent the market cap, the price, and the volume respectively. The OLS results I get is shown below.

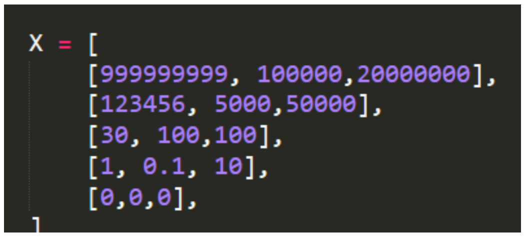


```
##################################
Results_ols [4508.89671172 1237.95667353 1068.87486938 1065.42756483 1065.42411507]
##################################
```

   By a glance at the data, we can tell that the rank is probably based on the market cap and volume. So we can guess that the first set should have the best rank while the last set should get the least rank. But the OLS result we get is quite different from what we are guessing.

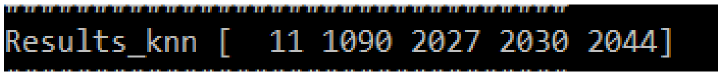
2. K Neighbors Classifier:

   In this part, I use the same dataset. When I run the knn model, I want to know the relation of the rank of each currency and the market cap, the price, and the volume of each currency. That is, the dependent variable Y is rank, the independent X are the market cap, the price, and the volume. I want to know if K Neighbors Classifier can predict a currency's rank given random values of market cap, price, and volume.



```
X = [
    [999999999, 100000,20000000],
    [123456, 5000,50000],
    [30, 100,100],
    [1, 0.1, 10],
    [0,0,0],
]
```

Here I use the same list of 5 different sets of variables. The variables represent the market cap, price, and volume respectively. The knn results I get is shown below.

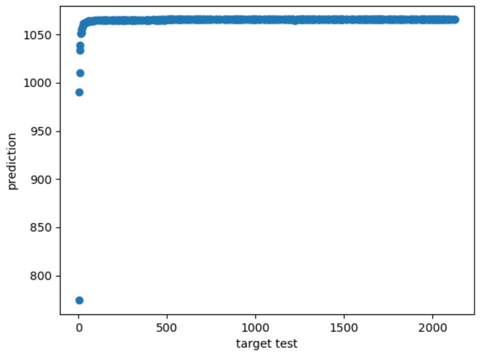


```
Results_knn [  11 1090 2027 2030 2044]
```

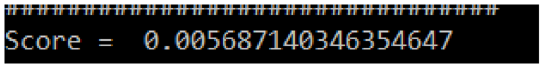
The result we get is quite the same as we thought. The more the market cap and the volume are, the better the rank it gets.

3. Train Test Split

In this part, I set the market as X and the rank as Y. I want to know whether the rank of each currency is affected by its market cap. I use the test and train to predict the result. The graph of results is shown below.



And the R-squared score I got from this model is shown below.

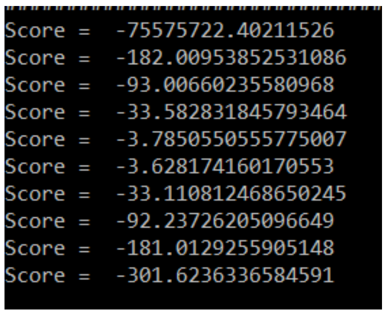


```
Score =  0.005687140346354647
```

The R-squared score is quite low, which tells us that very little information is explained by the variable is this model.

4. Kfold

In this part, I use the Kfold model to test ten splits and then predict each R-squared score.



```
Score =  -75575722.40211526
Score =  -182.00953852531086
Score =  -93.00660235580968
Score =  -33.582831845793464
Score =  -3.7850550555775007
Score =  -3.628174160170553
Score =  -33.110812468650245
Score =  -92.23726205096649
Score =  -181.0129255905148
Score =  -301.6236336584591
```