

Authors

Palak Sharma, sharma.pala@husky.neu.edu

Dhanisha Phadte, phadate.d@husky.neu.edu

Dharani Thirumalaisamy, thirumalaisamy.d@husky.neu.edu

HEADING : Data Science in FinTech Industries(Cryptocurrency)

The goal of this blog post is to provide a brief introduction of FinTech Industries and an easy way to cryptocurrency analysis using Python and R. We will walk through a simple Python and R script to retrieve, analyze, and visualize data on cryptocurrencies using data science feature like scrapping data set using Quandl API, exploratory data analysis in python and various graphical representation of real time bitcoin data in R.

What is FinTech?

FinTech stands for Financial Technologies, and in its broadest definition, that's exactly what it is: technologies used and applied in the financial services sector, chiefly used by financial institutions themselves on the back end of their businesses. Fintech is a portmanteau of financial technology that describes an emerging financial services sector in the 21st century. Originally, the term applied to technology applied to the back-end of established consumer and trade financial institutions. FinTech like crowdfunding, mobile payments, and money transfer services is revolutionizing the way small businesses start up, accept payments, and go global, and they are making it easier than ever to start and run a business.

2017 was still a busy year for blockchain. Venture capital investments topped \$1 billion for the first time. People are finding it easier to invest in the digital currency, thanks to the debut of firms such as Bitcoin Investment Trust. Big financial companies—Nasdaq, American Express and Visa—invested in blockchain startups, “a game changer in terms of attitude towards the technology”.

Cryptocurrency – the next new buzz word

The word crypto comes from the fact that most of the cryptocurrencies (Yes there's more than one!), use cryptography principles for generation, transfers etc. Cryptography is a branch of computer science that deals with encryption, decryption, passwords, and all that secret and safety stuff! Cryptocurrency is a form of digital money that is designed to be secure and, in many cases, anonymous. It is a currency associated with the internet that uses cryptography, the process of converting legible information into an almost uncrackable code, to track purchases and transfers.

Cryptography was born out of the need for secure communication in the Second World War. It has evolved in the digital era with elements of mathematical theory and computer science to become a way to secure communications, information and money online. The first cryptocurrency was bitcoin, which was created in 2009 and is still the best known. There has been a proliferation of cryptocurrencies in the past decade and there are now more than 1,000 available on the internet. Bitcoin soared as high as \$20,000 at the end of last year before crashing back to less than \$8000 now.

Data Science – ‘The New Oil’ for Cryptocurrency

WHY.....Required on first place????

Here are 7 ways how data science is at the core of the current transformation of the financial sector.

1. Payment and Transactions: Analysis and prediction of transaction volumes is key to enhance product value for customers. Data science enables better classification of payment records and thus allows banks to tailor additional services to their client's needs. This may vary from simple analytical features (How much did you spend on groceries last month?) to more advanced features such as the integration of payment records and personal data to allow for recommendation, loyalty rewards and other forms of proactive engagement.
2. Credit Risk Evaluation : With the vision to make “credit accessible for more people”, various FinTech startups are on the rush for clients and VC money. Their value proposition boils down to a “faster and more accurate credit risk evaluation” process than at traditional banks, which enables them to reach a broader client base and minimize credit default rates.
3. Revenue and Debt Collection : Data science enables the utilization of powerful predictive models in order to optimize revenue and debt collection. Already at the moment of purchase it is possible to predict a probability of timely payment, thus making revenue collection more transparent.
4. Customer Journey Attribution : Customer Acquisition Costs and Customer Lifetime-Value are – as in most business models – key metrics for banks and financial service providers. Thus, minimizing churn rates and optimizing conversion rates are crucial activities within most financial organizations.
5. Fraud Detection and Prevention : While they have been on the agenda even before data science became a recognized term, fraud detection and prevention are still among the top priorities of FinTech executives
6. Portfolio Optimization and Asset Management : While portfolio optimization and asset management are generally among the more mature fields of application of data science, there are some exciting new avenues. Impact analyses and correlation with asset price developments are now executed in a seamless manner and allow for more creativity and insight. Testing and

optimization of capital allocation strategies can be fully automated, which makes cuts on overhead costs feasible.

7. Corporate Compliance & Service Quality :Certainly not on the innovation end, but for larger institutions still of vital importance: implementation and tracking mechanisms for compliant behavior across the entire organization.

WHAT...are we dealing with???

Different types of datasets available for cryptocurrency :

a. Cross-sectional dataset

It groups the available data into groups based on the similarities in the data and builds the model. In statistical terms , it is a dataset that is taken at a particular time. The error between 2 observations will be zero. The predicted output will be numeric. Example : the consumption expenditure of individuals for a period of one month.

b. Time-series dataset

It refers to the data that is taken over a period of time at specific and equal-intervals of time. The observations made are independent of one another. The predicted output is numeric. Usually regression method is used to create a model. This mainly has 4 subtypes : Date , Date Time ,Time, Timestamp . Example : Stock market closing rate for a time span of one year.

c. Panel dataset

In statistics and econometrics, panel data or longitudinal data are multi-dimensional data involving measurements over time. Panel data contain observations of multiple phenomena obtained over multiple time periods for the same firms or individuals. Time series and cross-sectional data can be thought of as special cases of panel data that are in one dimension only (one panel member or individual for the former, one time point for the latter).

WHERE....to find them???

A number of data sources are available on internet, where data will be available freely to perform analysis and predict.

- www.Kaggle.com/datasets ::: Formats Available : csv , JSON,SQLite
- <http://archive.ics.uci.edu/ml/datasets> ::: Formats Available : csv
- <https://elitedatascience.com/datasets> ::: Formats Available : csv
- <https://www.quandl.com/> ::: Formats Available : csv, JSON,xml
- <https://www.census.gov/data.html> ::: Formats Available : csv, sas
- <https://relational.fit.cvut.cz/> ::: Formats Available : MySQL database

CSV format :

```
import urllib2
import csv
url = ("website from where the data is taken")
data = urllib2.urlopen(url)
reader = csv.DictReader(data)
for record in reader:
    print record
```

We can use ScraperWiki library instead of urllib .

```
import csv
import scraperwiki
url = ("website from where the data is taken")
data = scraperwiki.scrape(url)
data = data.splitlines()
reader = csv.DictReader(data)
for record in reader:
    print record
    #for scraperwiki only:
    scraperwiki.sqlite.save(['Value'], record)
```

JSON format :

1. Create a base class to handle the website :

```
import re, json
import mechanize
import urlparse
import urlutil
from bs4 import BeautifulSoup

class BrassringScraper(object):
    def __init__(self, url):
        self.br = mechanize.Browser()
        self.url = url
        self.soup = None
        self.data = []
        self.numdataretrieved = 0
```

2. Add a scrape method :

```
def scrape(self):
    self.open_search_openings_page()
    self.submit_search_form()
    self.scrape_jobs()
```

3. Submit the search form :

```
def submit_search_form(self):
    self.soupify_form(soup=None, form_name='aspnetForm')
    self.br.select_form('aspnetForm')
    self.br.submit()
    self.soup = BeautifulSoup(self.br.response().read())
```

4. It runs the form through BeautifulSoup and sets the current page in HTML form . No using the attributes we can load the JSON data in to it :

```
def scrape_data(self):
    while not self.seen_all_data():
        t = 'ctl00_MainContent_GridFormatter_json_tabledata'
        i = self.soup.find('input', id=t)
        j = json.loads(i['value'])

        for x in j:
            # debug
            print '\n'.join('%s\t%s' % (y,z) for y,z in
x.items())
            print '\n'

            data = {}
            data['date'] = self.get_date(x)
            data['high'] = self.get_high(x)
            data['low'] = self.get_low(x)

            self.data.append(data)

        self.numdataseen += len(j)
```

XML format :

PARSING :

- Parsing the XML package has 2 basic models - DOM & SAX.
- Document Object Model (DOM) in which tree is stored internally as C, or as regular R objects
- Uses XPath to query nodes of interest, to extract info.
- It writes recursive functions to "visit" nodes which extracts information as it descends tree.
- It extracts information to R data structures via handler functions that are called for particular XML elements by matching XML name.

- For processing very large XML files with low-level state machine via R handler functions closures are used.

IMPLEMENTATION :

The preferred approach is using DOM.

Here are a list of packages to be used:

- xmlName() - element name (w/w.o. namespace prefix) xmlNamespace()
- xmlAttrs() - all attributes
- xmlGetAttr() - particular value
- xmlValue() - get text content.
- xmlChildren()
- xmlSApply()
- xmlNamespaceDefinitions()

HTML format :

```
from lxml import html
import requests
```

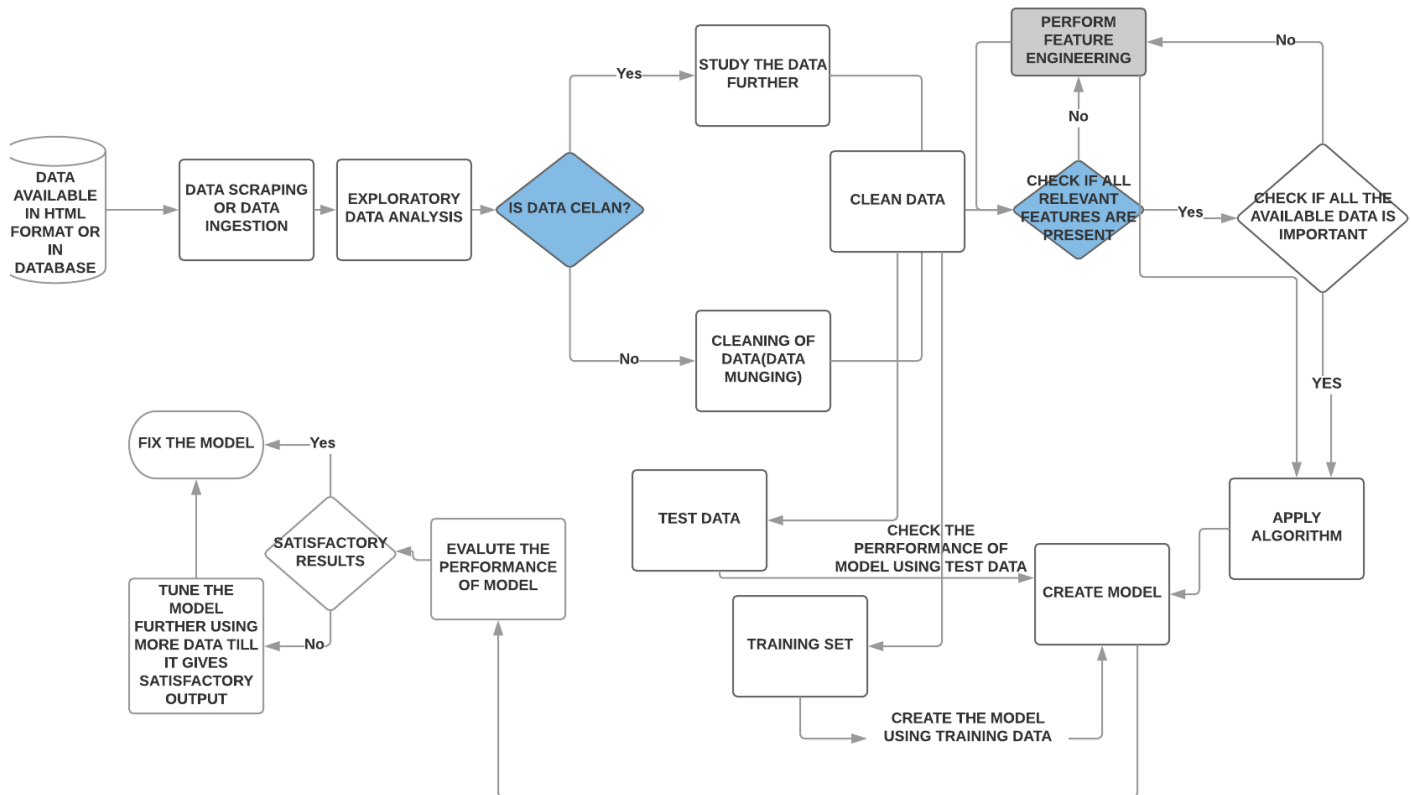
```
page=requests.get('http://econpy.pythonanywhere.com/ex/001.html')
tree=html.fromstring(page.content)
```

```
<divtitle="buyer-name">Carson Busses</div>
<spanclass="item-price">$29.95</span>
buyers=tree.xpath('//div[@title="buyer-name"]/text()')
prices=tree.xpath('//span[@class="item-price"]/text()')
```

```
print'Buyers: ',buyers
```

```
print'Prices: ',prices
```

FLOW DIAGRAM OF DATA ANALYSIS



HOW....to access???

1. Retrieve Bitcoin data using Quandl API

Quandl offers free Bitcoin exchange rates for 30+ currencies from a variety of exchanges. Quandl's simple API gives access to Bitcoin exchanges and daily Bitcoin values. Accessing Bitcoin data via the API is no different than the mechanism for all data on Quandl.

Import the Dependencies at the Top of the Notebook

```
In [13]: #to extract the data using QuadAPI
import os # OS module provides function that allows you to interface with the operating system that Pyt
import numpy as np #provides a high-performance multidimensional array object,
import pandas as pd
import pickle # Pickling is a way to convert a python object (list, dict, etc.) into a character stream
#serializing and de-serializing a Python object structure
import quandl #Quandl requires NumPy (v1.8 or above) and pandas (v0.14 or above) to work.
#Package for quandl API access
#Basic wrapper to return datasets from the Quandl website as Pandas dataframe objects with a timeseries
```

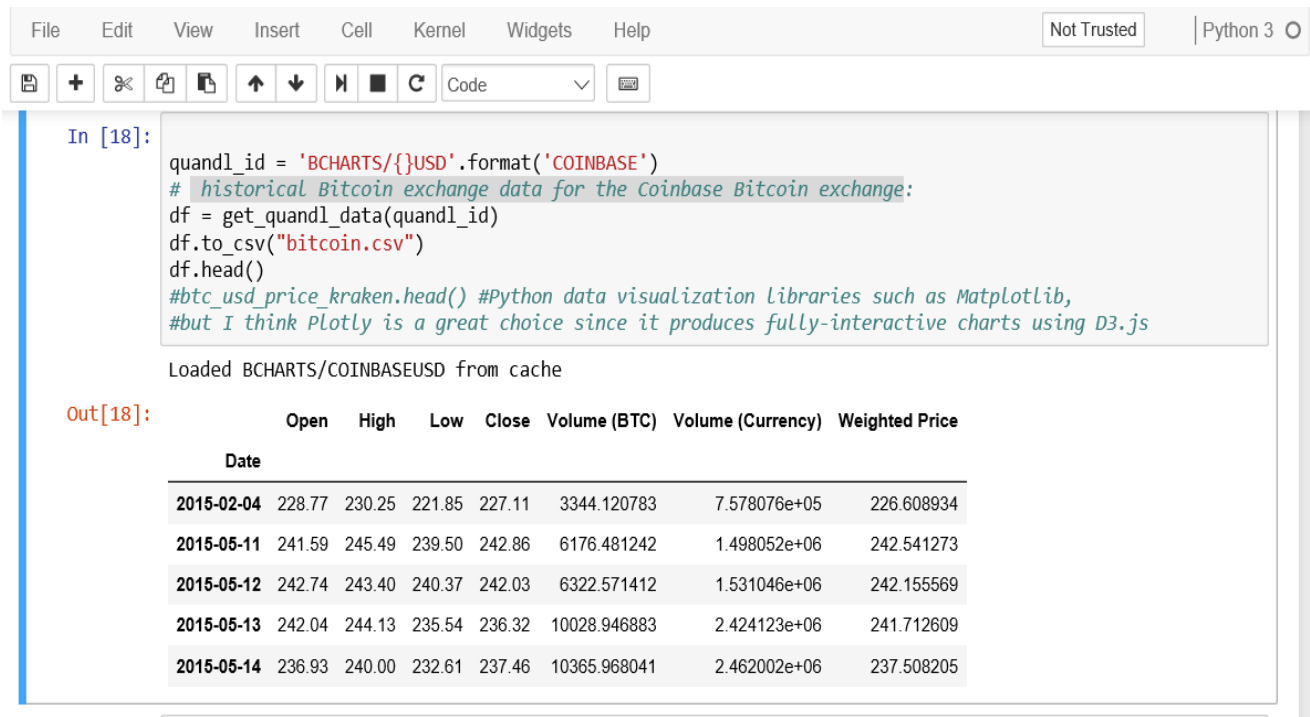
To get Bitcoin pricing data using Quandl's free Bitcoin API.

Define a function to download and cache datasets from Quandl.

```
In [15]: def get_quandl_data(quandl_id):
        '''Download and cache Quandl dataseries'''
        #taking data series from cached path
        cache_path = '{}.pkl'.format(quandl_id).replace('/', '-') # creat pkl file as quandl_id.pkl and repl
        #cache_path = BCHARTS-KRAKENUSD.pkl
        try:
            f = open(cache_path, 'rb') #open the pkl file to read
            df = pickle.load(f) # load the data from file into df object
            print('Loaded {} from cache'.format(quandl_id)) # loaded from the pkl file which was cached
        except (OSError, IOError) as e:
            print('Downloading {} from Quandl'.format(quandl_id))
            # getting data from quandl website using quandl module and API as pandas dataframe
            df = quandl.get(quandl_id, returns="pandas")
            df.to_pickle(cache_path)# storing the data from df to pkl file
            print('Cached {} at {}'.format(quandl_id, cache_path))
        return df
```


Using pickle to serialize and save the downloaded data as a file, which will prevent our script from re-downloading the same data each time we run the script. The function will return the data as a Pandas dataframe.

Pull Historical Bitcoin exchange data for the Coinbase Bitcoin exchange



The screenshot shows a Jupyter Notebook interface. The top bar includes a menu (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a 'Not Trusted' status indicator, and 'Python 3' as the selected kernel. Below the menu is a toolbar with icons for saving, adding, deleting, and other notebook actions. The main area contains a code cell labeled 'In [18]:' with the following Python code:

```
quandl_id = 'BCHARTS/{}USD'.format('COINBASE')
# historical Bitcoin exchange data for the Coinbase Bitcoin exchange:
df = get_quandl_data(quandl_id)
df.to_csv("bitcoin.csv")
df.head()
#btc_usd_price_kraken.head() #Python data visualization libraries such as Matplotlib,
#but I think Plotly is a great choice since it produces fully-interactive charts using D3.js
```

Below the code cell, the output is displayed. It starts with the text 'Loaded BCHARTS/COINBASEUSD from cache'. This is followed by 'Out[18]:' and a Pandas DataFrame table. The table has columns: Date, Open, High, Low, Close, Volume (BTC), Volume (Currency), and Weighted Price. The data rows represent historical Bitcoin prices from February 4, 2015, to May 14, 2015.

Date	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
2015-02-04	228.77	230.25	221.85	227.11	3344.120783	7.578076e+05	226.608934
2015-05-11	241.59	245.49	239.50	242.86	6176.481242	1.498052e+06	242.541273
2015-05-12	242.74	243.40	240.37	242.03	6322.571412	1.531046e+06	242.155569
2015-05-13	242.04	244.13	235.54	236.32	10028.946883	2.424123e+06	241.712609
2015-05-14	236.93	240.00	232.61	237.46	10365.968041	2.462002e+06	237.508205

2. Fetching data using Web Scrapping

- When performing data science tasks, it's common to want to use data found on the internet. You'll usually be able to access this data in csv format, or via an Application Programming Interface (API). However, there are times when the data you want can only be accessed as part of a web page. In cases like this, you'll want to use a technique called web scraping to get the data from the web page into a format you can work with in your analysis.
- It is very necessary to be familiar with the HTML structure before you start to scrap the data. Scrapping can be done with Python using BeautifulSoup and requests libraries.pandas, and matplotlib library is used to perform some simple analysis using.
- Use urllib library to request the web base from server by the passing the url page.

Is the Data Suitable for Modelling???

Once upon a time there was a boy named Data. Throughout his life, he was always trying to understand what his purpose was. What values do I have? What impact can I make on this world? Where does Data come from? These questions were always in his mind and fortunately, through sheer luck, Data finally came across a solution and went through a great transformation. It all started as Data was walking down

the rows when he came across a weird, yet interesting, pipe. On one end was a pipe with an entrance and at the other end an exit. The pipe was also labeled with five distinct letters: “**O.S.E.M.N.**”.

OSEMN Pipeline

O — Obtaining our data

S — Scrubbing / Cleaning our data

E — Exploring / Visualizing our data will allow us to find patterns and trends

M — Modeling our data will give us our predictive power as a wizard

N — Interpreting our data. At this stage, we already have our data and even we have scraped some data on cryptocurrency, the focus of this blog will be on the “**E**” of pipeline : Exploring and Visualizing data via EDA and Feature Engineer.

EXPLORATORY DATA ANALYSIS on the data we scraped via API (Python)

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to :

1. maximize insight into a data set;
2. uncover underlying structure;
3. extract important variables;
4. detect outliers and anomalies;
5. test underlying assumptions;
6. develop parsimonious models; and
7. determine optimal factor settings.

Lets get started with some python codes leading to EDA on the data we scraped using APIs in the previous section :

- A. Know the datatypes you are dealing with and club them into one category

```
In [1]: import pandas as pd
```

```
In [2]: data= pd.read_csv(r"C:\Users\palak\Documents\Python\bitcoin.csv")
```

```
In [3]: data.dtypes
```

```
Out[3]: Date                object
Open                float64
High                float64
Low                 float64
Close               float64
Volume (BTC)        float64
Volume (Currency)   float64
Weighted Price       float64
dtype: object
```

```
In [7]: def get_var_category(series):
        unique_count = series.nunique(dropna = False)
        total_count = len(series)
        if pd.api.types.is_numeric_dtype(series):
            return 'Numerical'
        elif pd.api.types.is_datetime64_dtype(series):
            return 'Date'
        elif unique_count == total_count:
            return 'Text (Unique)'
        else:
            return 'Categorical'
    def print_categories(df):

        for column_name in df.columns:
            print(column_name, " :", get_var_category(df[column_name]))

    print_categories(data)

Date : Text (Unique)
Open : Numerical
High : Numerical
Low : Numerical
Close : Numerical
Volume (BTC) : Numerical
Volume (Currency) : Numerical
```

B. Calculate if any missing values

```
In [11]: #data['Volume'].astype(str).astype(int)
lengtho = len(data['Open'])
print(lengtho)
```

659360

```
In [13]: count = data['Open'].count()
print(count)
```

658996

```
In [14]: missing_rows = length-count
perc_miss_data = float(missing_rows / length)
perc_miss_data = "{0:.1f}%".format(perc_miss_data*100)
print(perc_miss_data)
```

0.1%

C. Computing Summary Metrics for the overall data

```
In [5]: print("Minimim Value : ", data["Open"].min())
```

Minimim Value : 1e-10

```
In [6]: print("Maximum Value : ", data["Open"].max())
```

Maximum Value : 99917.8

```
In [7]: print(data["Open"].mode())
```

0 0.000002
dtype: float64

```
In [9]: mean = data["Open"].mean()
print(mean)
```

35.58682048554945

```
In [11]: median = data["Open"].median()
print(median)
```

0.006664

D. Data Profiling

```
In [9]: import pandas_profiling
```

```
In [10]: pandas_profiling.ProfileReport(data)
```

Overview

Dataset info

Number of variables	8
Number of observations	882
Total Missing (%)	0.0%
Total size in memory	55.2 KiB
Average record size in memory	64.1 B

Variables types

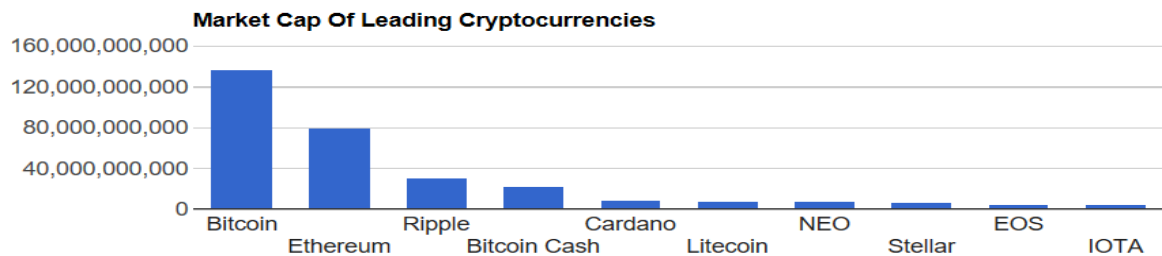
Numeric	3
Categorical	0
Boolean	0
Date	0
Text (Unique)	1
Rejected	4
Unsupported	0

Warnings

- Close is highly correlated with Low ($p = 0.99792$) Rejected
- High is highly correlated with Open ($p = 0.99787$) Rejected

EXPLORATORY DATA ANALYSIS on real time data (R Studio)

```
1 library(coinmarketcapr) #to get the cryptocurrencies market cap prices from "coin market cap"
2 library(formatR) # to reformat R code to increase readability
3 library(yaml) #convert R data int yaml(to create ordered maps)
4 library(googlevis) #R interface to google chart
5 library(knitr) #packet for dynamic report generation
6 data_sample <- get_marketcap_ticker_all()
7 mapply(class,data_sample)
8
9
10 for(i in c(4:ncol(data_sample))) {
11   data_sample[,i] <- as.double(data_sample[,i])
12 }
13
14 mark <- gviscolumnchart(data_sample[1:15, ], "name", "market_cap_usd", options = list(title = "Market cap of Leading cryptocurrencies",
15                                           legend = "left"))
16 plot(mark)
```



Data: data • Chart ID: [ColumnChartIDfb867882161](#) • [googleVis-0.6.2](#)
R version 3.4.3 (2017-11-30) • [Google Terms of Use](#) • [Documentation and Data Policy](#)

```

library(httr) #
library(Rcurl)
library(crypto)
library(googlevis)
installed.packages(googlevis)

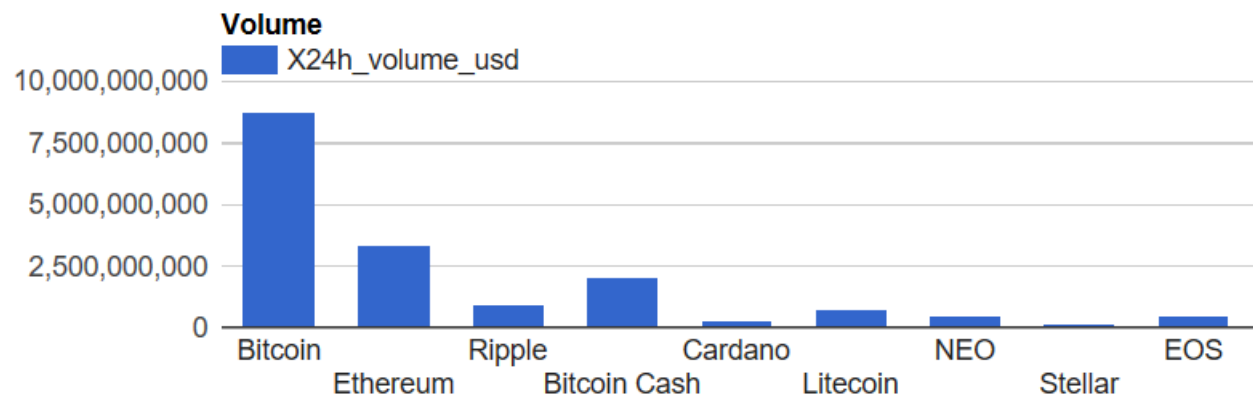
sample_data <- crypto::listcoins()
data1 <- crypto::getCoins(coin = c("Bitcoin", "Ethereum", "Ripple", "Bitcoin Cash",
                                   "Cardano"))

print(data1)
p = gvisAnnotationChart(data1, idvar = "name", "date", "market", options = list(title = "Market Cap Trend",
                                         legend = "top"))

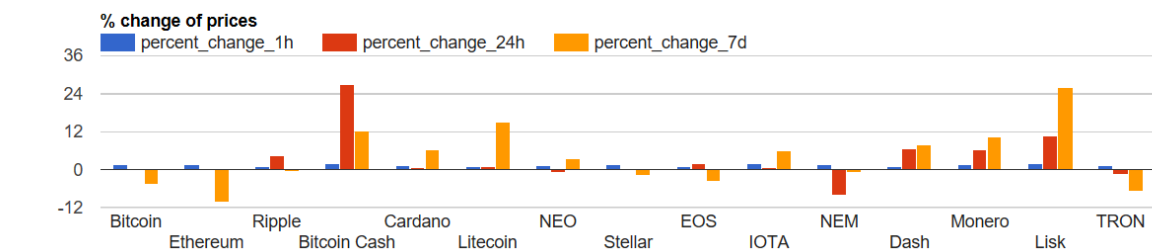
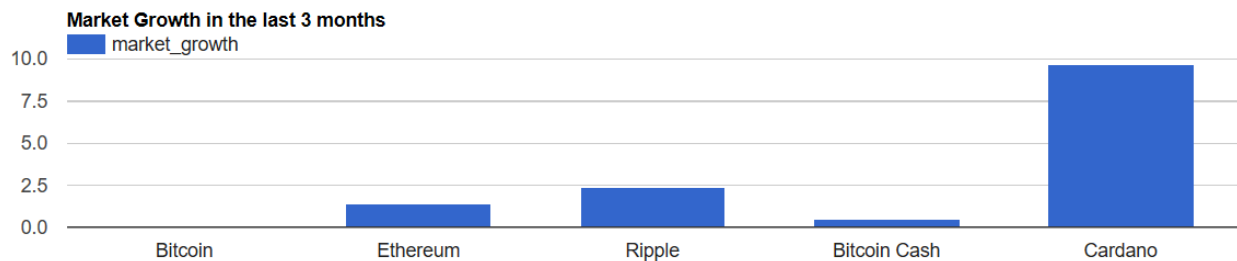
plot(p)
ohl1c = gviscandlestickchart(data1[data1$name == "Bitcoin" & data1$date < "2017-01-31" &
                                   data1$date > "2017-01-01", ], "date", "low", "open", "close", "high", options = list(title = "OHLC chart of Bitcoin in Jan",
                                                                 legend = "top", height = 600))

plot(ohl1c)
install.packages("quantmod")
library(quantmod)
coins <- unique(data1$name)
s_date = Sys.Date() - 1
data1$market_growth <- NA
for (i in coins) {
  data1[data1$name == i, ]$market_growth = Delt(data1[data1$name == i, ]$market,
                                                type = "arithmetic", k = 90)
}
c_data = data1[complete.cases(data1), ] #complete.cases is used to get rwcords without NA
mg_3 = gviscolumnchart(c_data[c_data$date == s_date, ], "name", "market_growth",
                      options = list(title = "Market Growth in the last 3 months", scaletype = "allfixed",
                                      legend = "top"))
plot(mg_3)

```



Data: data • Chart ID: [ColumnChartIDfb811fa3cdb](#) • [googleVis-0.6.2](#)

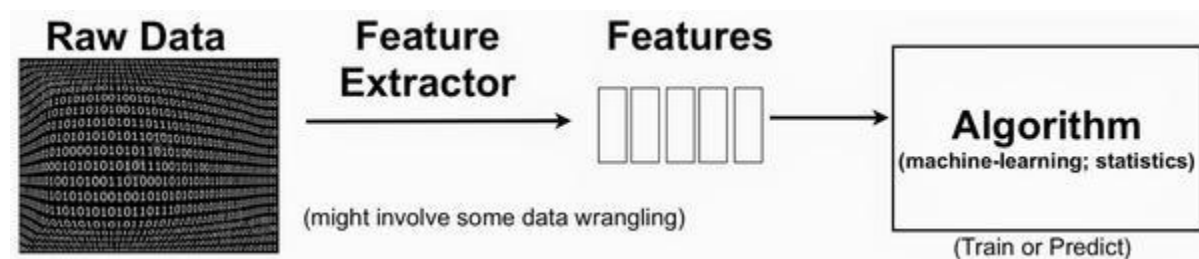


FEATURE ENGINEERING :

This is one of the method that is agreed to be the key success of Machine learning.

For a Machine Learning algorithm to be successful , it is important how well the data fits in the predictive model , i.e, it is important to choose the proper feature to make this happen. This is where feature engineering comes into picture.

Feature engineering is defined as a process of transforming raw data into features that represent the problems to models that one creates , which results in a better model with better accuracy for future predictions with new set of data.



Steps involved :

Feature extraction - reducing the dimensionality of the available data set so that it gives better result .

Out of this not all features will be of importance to us. So we have to select features that are more relevant to the analysis that one is performing. So , the next step is Feature Selection.

Sometimes , the feature which is at most important to the analysis will not be available. In that case one has to develop a feature. It is called Feature Construction. This feature should be related to the other existing features.

Once the data is with us it is equally important to know about the features - Feature Learning.

Process involved :

Brainstorming - coming up with a list of features which might fit best into the model.

Selection of feature - picking up the most relevant one

Evaluation of models - checking if the new data fits in the model that is already created .

Example :

This is a simple example that explains feature engineering.

Let's consider we have item_colour as one feature which has red , yellow, green and no colour as attributes.

In this case we can create another feature called value_of_colour and set it to 1 if the item has colour and to 0 if it has no colour. This will give us a better idea about the data that we have.