Bitcoin metric report scripts

July 13, 2022

Install these libraries first for the script to work. Install them one by one, do not execute the below script all at once, it will give you an error.

[ ]:
```
pip install pandas
pip install numpy
pip install matplotlib
pip install seaborn
pip install datetime
pip install yfinance
pip install pytrends
pip install quandl #you need to get your own free API key by making an account␣
↪on Quandl, I cannot share mine.
```

This script calls the *Quandl API* and *Yahoo finance* library to get all our data metrics which we want in our BI report

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import quandl
import datetime as datetime
import yfinance as yf
from functools import reduce

#This function will get the bitcoin metrics from qunadl where metrics variable␣
↪is a 'list'
def Blockchain(metrics):
    quandl.ApiConfig.api_key = [redacted] # get the API key to get␣
↪datasets from quandl blockchain.com database
    lst_dataframe = []
    for x in metrics:
        df = quandl.get(x)
        lst_dataframe.append(df)

    #concatenate the different dataframes with an outerjoin, NaN values will be␣
↪initial so should be filled with 0
```

```
    df_final = reduce(lambda  left,right: pd.merge(left,right,on=['Date'],␣
 ↪how='outer'),  lst_dataframe).fillna(0)
    return df_final

# This function will get index data from yahoo finance like for gold, S&P-500␣
 ↪etc.
def yahoo(index):
    df_index = yf.download(index, start='2009-01-02',end=datetime.date.today() -␣
 ↪datetime.timedelta(days=1),
                           progress=False) #index data from yahoo finance
    df_index= df_index.drop(['Open','High','Low','Adj Close','Volume'], 1)
    return df_index

# First get data from quandl
'''BCHAIN/MKPRU #bitcoin daily closing price
BCHAIN/TRVOU # daily exchange trade volume in USD
BCHAIN/DIFF # daily measure of difficutly of mining a block
BCHAIN/HRATE #daily measure of hashrate
BCHAIN/ETRAV #daily number of transactional volume
BCHAIN/NTRBL # daily number of average transactions per block
BCHAIN/TRFUS #daily number of total transactions
BCHAIN/MKTCP #bitcoin market capitalization by day'''

data = ['BCHAIN/MKPRU', 'BCHAIN/TRVOU', 'BCHAIN/DIFF', 'BCHAIN/HRATE', 'BCHAIN/
 ↪ETRAV', 'BCHAIN/NTRBL', 'BCHAIN/TRFUS','BCHAIN/MKTCP']
df = Blockchain(metrics = data)

#Rename the columns
df.columns = ['BTC Price USD', 'Trade Vol USD', 'Difficulty', 'Hash Rate',␣
 ↪'Transaction Vol', 'Transactions per block', 'Total Transactions','Market␣
 ↪Capitalization USD']

#Now adding the tech stocks accounted in NASDAQ 100
df_NASDAQ = yahoo(index = '^OEX')
df_NASDAQ = df_NASDAQ.rename(columns={'Close': 'NASDAQ-100',})

#Now for S&P 500
df_SandP500 = yahoo(index = '^GSPC')
df_SandP500 = df_SandP500.rename(columns={'Close': 'S&P 500',})

# For gold and silver, using the philadelphia gold and silver index
df_gold = yahoo(index = '^XAU')
df_gold = df_gold.rename(columns={'Close': 'Gold & Silver',})

#For brent crude Oil price
df_oil = yahoo(index = 'CL=F')
df_oil = df_oil.rename(columns={'Close': 'Oil',})
```

```
#Now combine the dataframes into one
df12 = [df_NASDAQ, df_SandP500, df_gold, df_oil]
df_yahoo = reduce(lambda  left,right: pd.merge(left,right,on=['Date'],␣
 ↪how='outer'), df12).fillna(method ='bfill').fillna(method='ffill')

df13 = [df, df_yahoo]
df_final = reduce(lambda  left,right: pd.merge(left,right,on=['Date'],␣
 ↪how='outer'), df13).fillna(method ='bfill').fillna(method='ffill')

df_final.reset_index(inplace=True)
```

This script prints the BTC correlation Line plot against the highest correlated feature

[47]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
import seaborn as sns
import numpy as np

# This snippet selects the highest correlated feature
correlations = dataset.corr()
correlations = correlations.reset_index()
x = correlations.nlargest(2, ['BTC Price ($)'])
highest_correlated = list(x.loc[x['BTC Price ($)'] < 1]['index'].
 ↪astype('string'))[0] # gives highest correlated field to BTC price

# These lines generate the data to be plotted

fig, ax1 = plt.subplots(figsize= (12.5,3.2)) # initializes figure and plots

ax2 = ax1.twinx() # applies twinx to ax2, which is the second y axis.

sns.lineplot(x = dataset['Date'], y = dataset['BTC Price ($)'], ax = ax1, color␣
 ↪= 'green') # plots the first set of data, and sets it to ax1.
sns.lineplot(x = dataset['Date'], y = dataset[highest_correlated], color =␣
 ↪'blue', ax = ax2) # plots the second set, and sets to ax2.

# these lines add the annotations for the plot.
ax1.set_xlabel('Date')
ax1.set_ylabel('BTC Price ($)', color='g')
for label in ax1.get_yticklabels():
    label.set_color("green")

ax2.set_ylabel(highest_correlated, color='b')
for label in ax2.get_yticklabels():
```

```
    label.set_color("blue")


ax1.xaxis.set_major_locator(mdates.WeekdayLocator(interval=5))
ax1.xaxis.set_major_formatter(DateFormatter("%d-%m"))
ax1.set_title('BTC Correlation Line Plot By Date')
plt.tight_layout()
plt.show(); # shows the plot.
```

This script prints the correlation heatmap

[47]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

plt.figure(figsize= (8, 4.1))
sns.set(font_scale=1.1)
heatmap = sns.heatmap(dataset.corr(), vmin=-1, vmax=1, annot=True,␣
 ↪cmap='coolwarm_r', linewidths=1)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12)
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```

To get google trends data, use below script

[ ]:
```
from pytrends import dailydata
from pytrends.request import TrendReq

#set date range to yout liking. Format is Year then Month number.
df_interest = dailydata.get_daily_data('bitcoin', 2009, 1, 2022, 3)

df_interest = df_interest.
 ↪drop(['bitcoin_unscaled','bitcoin_monthly','isPartial','scale'], 1) # drop␣
 ↪unnecessary columns
df_interest = df_interest.rename(columns={'bitcoin': 'Interest_Index'}) # rename␣
 ↪column for keyword interest
df_interest.index.names = ['Date'] # set index to Date
```