Social Signal Sentiment-Based Prediction for Cryptocurrency Trading

Table of Contents

Start with the Documentation

Development Environment

Code Editor

The code editor used was Visual Studio Code with their tremendous amount of extensions. Particular helpful extensions were the Jupyter Notebook, the TabNine AI supported Autocomplete and the LTeX LanguageTool that checked grammar and spell Markdown files. This way it was possible to directly write the documentation inside VSCode.

Environment Setup

To ensure version control for used python libraries the package and environment management system Conda was used.

With conda environments it is possible to work with defined python and package versions.

Setting up an environment is easy:

1. Install conda with `pip install conda`
2. Create the environment with `conda create --name myenv`
3. See if environment was created `conda env list`
4. Activate environment with `conda activate myenv`

To create an environment from an existing `requirements.txt` file add this to the second step:

```
conda create --name myenv --file requirements.txt
```

The following packages were used:

- pandas
- matplotlib
- sentiment
- streamlit
- wordcloud
- numpy
- openpyxl
- regex==2022.3.2
- pyOpenSSL
- demoji
- python-dateutil
- python-dotenv
- tweepy
- SQLAlchemy
- vaderSentiment
- psycopg2-binary
- streamlit_autorefresh
- boto
- schedule

- postgres
- python-binance
- python-kucoin

Export to requirements.txt or requirements.yml with

- `conda env export > environment.yml`
- `pip freeze > requirements.txt`

Local Development

Tweepy Stream and local Export

1. Activate the conda environment with required packages (see above)
2. Get your own API-Keys from Twitter API and Kucoin Sandbox and add them to .env-file
3. Set up a Postgres Database and add DB_URL to .env
4. Edit runner.py :
    - Uncomment line 48 for local export
    - Either add the keywords for the coins in the last line:
      Runner(['btc','ada','eth']))

      or Uncomment lines 66 - 72
5. run script via terminal:
   `python3 runner.py -k "btc,eth,ada" -i 5`

Streamlit

```
1. cd streamlit
2. streamlit run 01_💬_Tweet-Sentiment.py
3. open http://localhost:8501
```

Folder-Structure

main - Folder

main - Folder	
config.py	File to get the Environment-Variabes
Procfile	A Heroku file for starting the processes
docs	Contains the ordered Documentation

sentiment - Folder	
filter.py	Functions to filter the tweets by checking for blacklisted words, duplicates and unnecessary symbols
keywords.py	Class to build a keyword list for coins
listener.py	Class to listen and filter tweets
runner.py	Main Class. Called in Procfile and starts Listener with Keywords
trade.py	Functions for Trading. Called every hour in Heroku Scheduler.
Logs-Folder	All Logs (Heroku, Tweepy, Excel, Json)

sentiment/database - Folder	
database.py	SQL-Alchemy Connection with Heroku database
exporter.py	Export Local Tweets to Json/Excel
Trade.py	Trade Class to declare the Format and Type of each Column in the Database
Tweet.py	Tweet Class to declare Format and Type of each Column in the Database

streamlit - Folder	
01_💬 _Tweet-Sentiment.py	Main File to visualise all the data from tweets, sentiment and trades with Streamlit
financial_data.py	Functions to get the data from Heroku Database, get prices from Binance and for building Signals
streamlit_data.py	Functions to edit the data from the databases: Splitting the Dataframe, calculate average and convert to signals.
visualise.py	Functions to visualise the price chart and words.