BLOXVERSE BACK-END USER'S GUIDE AND CODE FLOW

1. ico_scrap

a) unlisted_icos.py

This code will gather all the ICO names, their token names and their website (URL) keywords from ICOBench, ICOBazaar and ICORating. The final output will be the outdata/ico_unlisted.csv file that contains the information to be used by scrap_icos_full.py.
-Needs to be the first thing to run (every 4 months)
-Run with: caffeinate -i python3 unlisted_icos.py

b) scrap_icos_full.py

This is the main code that loops around entire unique collected ICO dataset (including companies listed on CMC and unlisted ICOs from Coindesk and returns the main final ICO datasets that are used for ML post-processing (8420 data points as of Q4 of 2018).
First it selects the data that have at least the "End Date" information available (reducing to 4054 data points in Q4 as of Q4 of 2018). Then it appends the "Google News" information from Google but gives the value "0" to the data that cannot be collected. This data is not used for ML/Rating purposes so that does not have any effect.
Produces:

- outdata/ico_data_full.txt: (that contains all data for all 8420 ICOs)
- outdata/ico_data_reduced.csv: (contains the reduced, 4054 (Q4, 2018) dataset with incomplete feature information marked as 'N/A'.
- outdata/ico_data_reduced2.csv: Incomplete or non-existent market performance information in ico_data_reduced.csv is marked with -1 day 1 return, 0 volume and -100 for the rest of the market indices. ***This is then copied back to outdata/ico_data_reduced.csv so this is the final file for further processing. This is the main file used for rating analytics on ICOs.***
- outdata/ico_data_complete.csv: This is the reduced, complete dataset with no "N/A" occurrences for any features. For future ML predictive analytics and also for show in the website.
- outdata/ico_data_rate_cluster.csv: This is the reduced dataset with complete information on the main 11 features (team, hardcap, price, raised, success, region, industry, twitter, telegram, N_daily_views and N_daily_time). Original dataset is now reduced to 567 data-points ***used for clustering analytics on ICOs***.

-Run with: caffeinate -i python3 scrap_icos_full.py
Finally the outdata directory is copied to outdata2

c) scrap_icos_main_func.py
This is the core of the data collection and hierarchical reduction algorithm. One call to the ico_data_collector routine given an input_vector = [ico_name,ico_token,ico_web], rbtc (bitcoin returns) and r10 (top ten tokens on CMC average returns) returns all the information for an ICO (features and market performance from CMC).

d) html_tester.py
This script can be used to test the correctness/efficiency of each scrapping routine and the ico_data_collector before running the main scrap_icos_full.py code.

e) icobazaar.py, icobench.py, icodrops.py, icomarks.py, icorating.py,tokenmarket.py
These scripts scrap the corresponding ICO databases and collect the ICO feature data using Python's bs4 and the requests package.

f) googletwitter.py
This script collects the social/web stats for each ICO (Twitter/Telegram follower count, Alexa Website stats: daily website visits, daily website engagement).

g) bitcoin_returns.py, top10_returns.py,coin_returns.py
These scripts survey CMC and collect the return arrays for Bitcoin, the weighted, top 10 listed coins as well as an individual (input) coin. All used within the ico_data_collector call to compute market performance statistics.

2. ico_rank (contents to be verified and merged with ico_scrapper)

First copy entire outdata directory from ico_scrapper into ico_rank directory (unless merged).

a) keep_social.py
This small script reduces the ico_data_rate_cluster.csv dataset into a smaller sub-set that only contains the social (twitter, telegram, alexa numbers) for the ICOs with available success rates. *Needs to be run before doing any analytics.*

b) rank_ico.py
This script computes the overall (for the entire ICO space and history) ranking for an input ICO under investigation. Only input by the user is the ICO name and the token name for the ICO. If the ico_data_collector (called within the plot_icos.py code) does not find some features it will complete them by asking the user to manually input them.
-First a rating is computed with statistical comparison to past ICOs. Histograms are made for each available feature in the outdata/ico_data_reduced.csv dataset and then split into 5 “bars” each. Grading is done depending on which (more successful ICO) bar the input ICO belongs to. Threshold set to 0.7 (70% success) by default but 0 may not make big difference. The final grades for each feature are then averaged to get a BloxVerse rating in the 0-1, 0-5 and “ICO Rating Hype Score” scale.
-Second clustering is done. The code splits the entire ICO dataset into clusters that were optimized with the use of Silhouette analysis. Then it assigns the ICO under investigation to a cluster and computes statistics within that cluster. Main output here is the RADAR diagram that shows graphically how the input ICO compares with other similar ICOs in the cluster for each feature (SWOT analysis).
Clustering is done both for the entire 7 features (team, hardcap, price, twitter, telegram, N_daily_views, N_daily_time) and for just the social/web features (twitter, telegram, N_daily_views, N_daily_time).
Standard output can be passed to a file for inclusion into the report.

c) rank_ico_q_sa.py
Same as plot_icos.py but does the same analysis both on a per-quarter (same as current ICO) and per-half-year basis to ensure that the ICO under investigation is compared against a dataset that correspond to current cryptocurrency market conditions (the “time aspect”).
This code produces 4 RADAR diagrams (two for each time range compared, one for all features and one for just social features). This code provides different

rankings that the original, full dataset comparison method that seem to better predict the ICO performance.

Troubleshooting: This code can fail for Quarterly clustering because of the small dataset and the inability to perform reliable clustering. However one should comment out those sections temporarily and run further to get the half-year comparison. Also the ratings hold well regardless of the clustering failures.

d) ico_market_stats.py
This script scans through the outdata/ico_data_cluster.csv database and collects quarterly and annual information on global features and cryptocurrency market trends as to provide a picture of where the space is heading (i.e. cumulative funding per quarter, mean success per quarter, industry/region share per quarter etc). It also computes the total number of funds raised for ICOs.

e) ico_ratings_performance.py
This script is to be run one time to illustrate the capacity of BloxVerse to outperform traditional rating (case study-type) companies. Produces a plot of rating versus success for past ICOs and compares against other sources. Also produces a ranked list of ICOs and compares with those of other sources to illustrate better agreement with ICO success rates. Calculates correlations between ratings and final success rates. Shows our quarterly bloxverse score is better proxy to ICO success than all other rating agencies.

f) ratings_scale.py
This script generates all .csv and .npy files that serve as a comparison bases for the full, semi-annual and quarterly databases to streamline the calculation of the rank/score within ico_ratings_performance.py. The output of ratings_scale.py is stored within the *scaling_dataset directory* and then called from the bloxverse_rating.py function to calculate the final rating.

h) bloxverse_rating.py
This routine contains the ico_rank_full, ico_rank_quarter and ico_rank_semiannual functions that scan through the dataset produced by ratings_scale.py to quickly calculate the bloxverse rating score for each category (full ICO dataset, semi-annual and quarterly subset comparisons). These functions are called within ico_ratings_performance to speed up the comparison analysis between Bloxverse and all other ratings agencies.

<u>Recommended Execution Sequence (Every ~4 months to allow for new ICO/STO data)</u>

i) python3 html_tester.py: Verify for a few (5-10 ICOs) that information is still collected correctly and that no changes need to be done any scrapping scripts or ico_data_collector and the way CMC formats their time series.
ii) caffeinate -i unlisted_icos.py: Collect ICO names, token names and web handles
iii) caffeinate -i scrap_icos_main.py: Compile datasets to be used for ML (ico_data_reduced.csv, ico_data_rate_cluster.csv and ico_data_complete.csv).
iv) python3 keep_social.py (create social stats file for processing)
v) Edit ratings_scale.py to include new year/quarters to create new files for comparisons then: python3 ratings_scale.py to create new data in the scaling_dataset directory
vi) python3 bloxverse_ratings.py to show superiority of Bloxverse vs other rating agencies
vii) python3 ico_ratings_performance.py to show trends in the ICO market
ATTN: If execution of long scrap_icos_main.py frozen: find Process ID number and execute '>kill -CONT #PID' in another terminal to resuscitate run.

<u>Recommended Execution Sequence (Every time a new ICO needs to be investigated and rated)</u>

i) python3 rank_ico.py: Save RADAR figures and Standard Output to File
ii) python3 rank_ico_q_sa.py: Save RADAR figures and Standard Output to File. Comment-out accordingly to avoid problems with Quarterly comparisons.

BloxVerse advantages:

- Speed to deliver report and SWOT analysis to ICO
- Accuracy - outperforms traditional methods
- Richest ICO dataset
- Lower cost to ICO ($3,000 compared to $20,000 with ICORating, $5,000 with ICOBench - look more sources).
- Better analytics methods: Clustering important in addressing comparison within a group of similar ICOs. Quarterly/Half-Year comparison important in addressing market conditions closer to time of ICO.

Zoom into your ICOs galaxy and current time! Explore your ICO within the SpaceTime of the CryptoCurrency World