1.0 Overview
The functionality of the program can be conceptualized as a pipeline of operations, with each operation occupying its own discrete script. This design is reflected to a high degree within the *scripts* and *python_notebooks* subdirectories. Each operation within the pipeline exists as a standalone script capable of executing independently of it's sibling scripts (within reason), and, bar the preprocessing script, is coupled with it's own configuration file allowing for parameter control over a given scripts relevant behaviour.

The following will detail the operations performed by each of the subscripts, the main wrapper script, the accompanying data transformations that occur throughout the pipeline as well as any necessary configuration involved with running the scripts. Environment configuration and dependency management can be resolved by reading readme file contained within the code base.

2.0 Preprocessing
The operations contained within the preprocessing script can be summarised as follows:

- Import raw honey pot datasets for both static and dynamic features
- Preprocess static features and dynamic features dataframes
- Square static and dynamic feature dataframes, ensuring that a mutual set of user_id's are shared between the static and dynamic feature datasets
- Export the finalized static features, and export the intermediate dynamic features dataframes

Currently, only preprocessing for the honeypot dataset exists. There is also currently no configuration required for the preprocessing, as the preprocessing behaviour and output is statically defined. The preprocessing script can be incorporated within the main wrapper script, or executed in isolation by locating to the preprocessing sub-directory and calling *python3 hp_preprocessing.py*.

3.0 Feature Engineering
The operations contained within the feature engineering script can be summarised as follows:

- Import the intermediate, dynamic features dataframe (consolidated tweet-document corpus for all users)
- Configure and generate a count vector based upon the user configuration
- Create a term frequency matrix by vectorizing the consolidated tweet-document corpus utilizing the generated count vector, applying lemmatization, stemming etc. during the process
- Generate dynamic features using a document-topic distribution derived from the term frequency matrix (GOSS, LOSS, document-topic distribution entropy), based upon the user configuration
- Export the dynamic features dataframe

The feature engineering scripts are probably the most complex scripts contained within the project. Additional sub scripts utilized during the feature engineering process are contained within this directory as *nlp_vector_config.py* and *dynamic_features.py* respectively. For simplicity's sake however, the relevant operations are invoked in a decisive way within *hp_dynamic_feature_generation.py*.

Parameter configuration for the count vector and the lda modeller is contained within *configs/hp_fe_config.json*. Within this configuration file the n-gram range as well as the document min/max incidence threshold can be specified. The number of topics to be generated by the lda modeller (and thus utilized by the document-topic distribution) as well as the number of iterations undertaken by the lda modeller can also be specified within this configuration file.

The feature engineering script can be incorporated within the main wrapper script, or executed in isolation by locating to the preprocessing sub-directory and calling *python3 hp_dynamic_feature_generation.py* in a similar way to the preprocessing script.

4.0 Clustering

The operations contained within the clustering script can be summarised as follows:

- Create a master feature dataframe by merging the static and dynamic feature sets
- Perform kmeans clustering using a selected number of features contained within the master feature dataframe, appending the cluster allocation as an additional feature within the master dataframe
- Segment the master dataframe based upon the newly created cluster allocations, generate as many cluster dataframes as specified within the user configuration
- Evaluate the composition of the newly segmented dataframes
- Export segmented dataframes and the resultant analysis

Parameter configuration for the kmeans cluster model as well as the features to be utilized within this cluster model is contained within *configs/hp_cluster_config.json*. The number of clusters as well as the number of iterations used to fit the kmeans clusters can be specified. A boolean scheme indexing across a selection of the features contained within the master feature dataframe can be configured to select which features to utilize during the clustering process.

As always, the clustering script can be incorporated within the main wrapper script, or executed in isolation by locating to the clustering sub-directory and calling *python3 hp_clustering.py*.

5.0 Classification

The operations contained within the classification script can be summarised as follows:

- Import all cluster-segregated dataframes
- For each dataframe, generate a set of models specified within the user configuration using unique features for each cluster that are also specified within the user configuration
- For each group of models, evaluate the accuracy of the models via precision, recall, f1, support and cross validated model accuracy
- Concatenate and export all results

Parameter configuration for the models to utilize during the classification process and the feature specification for each cluster is contained within *configs/hp_classification_config.json*. The number of folds as well as the relative composition of training/testing data utilized during the kfold CV process has been set to values of 10 and 0.6/0.4. Similarly to the clustering config, a boolean scheme that indexes across all available models can be configured to determine which models are utilized within the final classification for each cluster (global effect). Finally, the features to utilize within each of the clusters can also be specified, configuration for this process does require that a number of cluster configurations commensurate with the number of available/previously defined clusters is provided for use.

As always, the classification script can be incorporated within the main wrapper script, or executed in isolation by locating to the classification sub-directory and calling *python3 hp_classification.py*.

6.0 Wrapper Script
The operations contained within the wrapper script can be summarised as follows:

- Call various sub-modules in combination with one another based upon the user configuration
- Execute these various configurations for both the honey pot dataset and the 14 million tweet dataset, based upon user configuration

The schema for the wrapper script follows the same boolean scheme featured within other configuration files, indexing across all of the previously detailed sub-scripts. The wrapper script is purposed with providing a convenient way to call combinations of the subscripts, without having to locate into their containing directories to invoke their functionality.

7.0 Data Processing Scheme
During the pipeline's lifecycle, the honeypot dataset is divided, transformed and consolidated multiple times. Some of the more important intermediate changes (particularly the output of a particular subscript) are written as csv files, allowing the the effect of subscripts to be examined or potentially piped/collected for use within other programs. Specifically, snapshots of the data are checkpointed:

- As they are originally contained, that is, as raw data
- After initial preprocessing; the state of the static features (inherited from a previous project) is such that the static features largely bypass the preprocessing phase, for the dynamic features, all tweet for a particular user are collated and grouped
- After dynamic feature generation; the state of the static features doesn't change at this point, however the dynamic feature set undergoes a substantial transformation (detailed within the feature engineering subscript)
- After the preliminary clustering; individual data frames segmented based upon the clustering results are saved

This state pipeline is once again highly reflected via the containing sub-directories. Appropriate directories for raw, preprocessed and final_feature datasets exist containing these various states.