Documentation for Client Usage Predictor (Decision Tree)

Code and Documentation produced by Lang (Ron) Chen July-October 2021 for Lucidity Software

Date updated: 11/2/2022

Table of contents

Module file structure

- Home directory:
    - Contains ./Initial, ./History, ContinuousTrain.py, Predict.py, Documentation.docx, Visualise_tree.ipynb, Mapping.xlsx, Hindsight.ipynb
    - Also will contain ./Prediction_Objects, ./Tree_visual, Statistics.csv, DateMatchWeek.csv and Predictions.csv which are outputs of the ContinuousTrain.py, Predict.py or InitialTraining.py scripts.
        - (./Prediction_Objects stores the prediction pickle objects which represents the best predictor objects as of yet; the ./Tree_visual stores the .dot objects which are used by the Visualise_tree.ipynb. Please do not manually alter these)
        - Statistics.csv: contains accuracy score statistics and other information of the predictor currently in use
        - Prediction.csv: contains predictions for client usage this week
        - DateMatchWeek.csv: contains list of date matched to week – helps user quickly and without much difficulty find out what week this week is
        - Mapping.xlsx – maps character codes of clients back to the numerical codes. Please manually update if new clients arise.

    - ContinuousTrain.py: script for training a new predictor object using the new data inputted each week
    - Predict.py: script for predicting client usage this week based on new data inputted
    - Visualise_tree.ipynb allows (with some adjustments of directories in the code) ad hoc visualising of the tree (** designed to be a 'tool' in future update/debugging rather than an essential feature that must run every week)
    - Hindsight.ipny allows ad hoc generation of retrospective.csv which matches the actual change in usage in clients compared to what was predicted.

- ./Initial:
    - Contains InitialTraining.py and ./Data
    - InitialTraining.py: script for training up initial predictor
    - ./Data contains the raw data (should be largest files in the whole file)

    - Also contains other files which are similar to those in home directory, and also ./Partial_Outputs which contains files that are created and stored permanently during the training process

- ./History:
    - Contains folders named by week number.
    - Each week's folder will contain ./Data (containing most recent data – not as

large as that in the Initial folder), ./Partial_Output (from running that week's ContinuousTrain.py script), and other files similar to the home directory.

- These are stored permanently so they can be accessed and reviewed in the future if required.

  **If one wishes to use the prediction objects of a certain week then must alter code of script in PART 10 of Predict.py

- Will also contain Predictions.csv, Retrospective.csv and Statistics.csv. The former of the two is the predictions made after running scripts in a particular week (of course, the predictions would be for the next week), while the latter contains the statistics of the predictor trained in that week.

Input and Output of the scripts

1. InitialTraining.py

Input: 1 argument and several other requirements for location and format of the data

- Argument 1: a cut-off date for data. Must be in form '[d]d/[m]m/yyyy' (If day or month single digit can input just as single digit).
    - please ensure date is valid and is after the FIRSTDATE (by default July 3rd 2017)
        - please see later about adjusting the default FIRSTDATE
    - logically this date should be a Sunday

- Other arguments (implicit):
    - -Initial raw data files need to be stored in directory './Data'.
    - -File names must be 'action.csv', 'assets.csv', 'competency_record.csv', 'form_record.csv', 'form_template.csv', 'incident.csv', 'users.csv', associated with the data of the corresponding filename
    - -Each csv must include columns that include 'domain' for company name, and 'created_at'. 'users.csv' must also have column name 'hr_type', with data values 'Casual', 'Employee', 'Subcontractor' denoting regular employees and 'InductionUser' denoting induction users.
    - -There should be no other tester domains in the data apart from 'demo', 'demo_2' and 'cruse'

    - -the dates for 'action.csv', 'competency_record.csv', 'form_record.csv', 'incident.csv', 'users.csv' should be in form of [d]d/[m]m/yyyy
    - -the dates for 'assets.csv', 'form_template.csv' should be in form of yyyy-mm-dd.

      *if the form of these are different then need to edit PART 2 of the script.

Outputs:

- Partial Outputs of Wrangled Data (many .csv's)
- Coefficients for Usage Scores (.csv)
- Quantile for Usage Scores and Percentage Changes (many .csv's)
- Tree Visualisation Objects (.dot)
- DecisionTreePredictor Objects (.pickle)
- Statistics File (.csv)

These are exported to various directories including the home directory (relatively '..'), the History directory (into the relevant week) and the current directory

2. Predictions.py

Input: 2 arguments and several other requirements for location and format of the data

- Argument 1: a start date for the recent data (script will only include data after this date when looking at the 'recent' data). Must be in form '[d]d/[m]m/yyyy' (If day or month single digit can input just as single digit).
    - please ensure date is valid and is after the FIRSTDATE (by default July 3rd 2017)
        - please see later about adjusting the default FIRSTDATE
    - logically this date should be a Monday

- Argument 2: a cut-off date for data. Must be in form '[d]d/[m]m/yyyy' (If day or month single digit can input just as single digit).
    - please ensure date is valid and is after the FIRSTDATE (by default July 3rd 2017) and after the date of Argument 1
        - please see later about adjusting the default FIRSTDATE
    - logically this date should be a Sunday

- Other arguments (implicit):
    - -Initial raw data files need to be stored in directory './History/Week {this week's week number}/Data'.
    - -File names must be 'action.csv', 'assets.csv', 'competency_record.csv', 'form_record.csv', 'form_template.csv', 'incident.csv', 'users.csv', associated with the data of the corresponding filename
    - -Each csv must include columns that include 'domain' for company name, and 'created_at'. 'users.csv' must also have column name 'hr_type', with data values 'Casual', 'Employee', 'Subcontractor' denoting regular employees and 'InductionUser' denoting induction users.
    - -There should be no other tester domains in the data apart from 'demo', 'demo_2' and 'cruse'

    - -the dates for all files should be in form of yyyy-mm-dd.

      *if the form of these are different then need to edit PART 2 of the script.

Outputs:

- Partial Outputs of Wrangled Data (many .csv's)
- Predictions for Client Usage This Week (.csv)

These are exported to various directories including the home directory (relatively '.'), the History directory (into the relevant week)

3. ContinuousTrain.py

Input: 2 arguments and several other requirements for location and format of the data

- Argument 1: a start date for the recent data (script will only include data after this date when looking at the 'recent' data). Must be in form '[d]d/[m]m/yyyy' (If day or month single digit can input just as single digit).
    - please ensure date is valid and is after the FIRSTDATE (by default July 3rd 2017)
        - please see later about adjusting the default FIRSTDATE
    - logically this date should be a Monday

- Argument 2: a cut-off date for data. Must be in form '[d]d/[m]m/yyyy' (If day or month single digit can input just as single digit).
    - please ensure date is valid and is after the FIRSTDATE (by default July 3rd 2017) and after the date of Argument 1
        - please see later about adjusting the default FIRSTDATE
    - logically this date should be a Sunday

- Other arguments (implicit):
    - -Initial raw data files need to be stored in directory './History/Week {this week week number}/Data'.
    - -File names must be 'action.csv', 'assets.csv', 'competency_record.csv', 'form_record.csv', 'form_template.csv', 'incident.csv', 'users.csv', associated with the data of the corresponding filename
    - -Each csv must include columns that include 'domain' for company name, and 'created_at'. 'users.csv' must also have column name 'hr_type', with data values 'Casual', 'Employee', 'Subcontractor' denoting regular employees and 'InductionUser' denoting induction users.
    - -There should be no other tester domains in the data apart from 'demo', 'demo_2' and 'cruse'

    - -the dates for all files should be in form of yyyy-mm-dd.

      *if the form of these are different then need to edit PART 2 of the script.

Outputs:

- Partial Outputs of Wrangled Data (many .csv's)
- Coefficients for Usage Scores (.csv)
- Quantile for Usage Scores and Percentage Changes (many .csv's)
- Tree Visualisation Objects (.dot)
- DecisionTreePredictor Objects (.pickle)
- Statistics File (.csv)

These are exported to various directories including the History directory (into the relevant week) and the home directory (relatively '.') if the new predictor is good enough to replace the current predictor.

4. Hindsight.py

Input: 1 argument

- Argument 1: The week number of the week that was predicted for (i.e. if want to perform hindsight analysis for the predictions made for week 220 then the argument should be '220')

Output:

- A retrospective.csv in ./History/Week {argument inputted – 1} (i.e. if did a retrospective on week 220 then the retrospective.csv would be placed in ./History/Week 219 because that is where the matching Predictions.csv is stored)

A note on the week numbers

This may be a point of future confusion so the developer will say a few more words

> Each Monday, the script should put the data of last week into the folder called './History/Week {lastweek number}/Data'.
>
> The Predictions.csv that is stored in ./History/Week {week number} is actually the predictions for Week {week number + 1} (i.e. this week is week 217, then the prediction.csv for this week's client usage change will be permanently stored in the folder of week 216 – and another copy will also be stored in the home directory until the script runs Predict.py in week 20)

Recommended usage

The recommended manner for this system to be used is as follows

- Monday Morning: place data from last week into the correct directory and run Predict.py (argument 1 = date of last Monday, argument 2 = date of last Sunday) to receive a prediction of client usage for the upcoming week
- Monday Morning: run the ContinuousTrain.py script (argument 1 = date of last Monday, argument 2 = date of last Sunday) to train up a new model for comparison with current one (replacing current if its statistics are better)

    *if these can be done automatically then the recommendation is to run them early Monday morning any time after the warehousing for the previous week is complete and before the team arrives for work

- Monday Morning (or any time this week for the matter of fact): run the Hindsight.py script (argument 1 = week number of this week)

Note the order of running the ContinuousTrain.py script and Predict.py script can be swapped – and thus if the inclusion of last week's data results in a better predictor than the existing one, the better predictor can be used for predicting this week's change. There is a chance that the new predictor is not as good as the current one and thus won't be used (ever), but even if it does upstage the old predictor, the output of the predictions are unlikely to make a significant difference so the order of running ContinuousTrain.py and Predict.py really shouldn't matter.

Further note if there were a few weeks where predictions/training was missed, that is ok. The script is designed so that if say the script last predicted and trained on week 218 (so that means last activity in the folder of WEEK 217!!), and 'forgot' to do so on week 219, and week 220, then on week 221 simply put the data of week 218, 219 and 220 into ./History/Week {220}/Data, and run the Predict.py ContinuousTrain.py scripts with argument 1 = the Monday of week 218 and argument 2 = the Sunday of week 220. This will mean there is no Week 218 and Week 219 folder in History but otherwise the module will work as if it's never missed Week 218 and Week 219.

(fun fact: this setup is inspired by the Markov property in math where the event next week is only dependent upon the event of the current week! In fact if you read the Data Science Decisions section, you'll see this theory is one of the principle inspirations behind the entire project – used for both the algorithm and the design of the module)

**Disclaimer:

- At least in the early phases of trailing this system the predictions should be used only to assist the BI group in terms of filtering the companies rather than an extremely trustworthy tool to base important decisions on
- The predictor would likely flag around 10% of all data, regardless of the week. It may be more volatile around Christmas times or Easter times when every company's drops.

- The predictor could also flag more than 10% under unusual circumstances (i.e. during COVID lockdowns)

Sample command line code for running scripts in Shell

(assuming currently in home directory and this is Monday of week 223 – 4th October 2021, and all predictions/training has been done for the previous weeks)

```
python Predict.py 27/9/2021 3/10/2021

python ContinuousTrain.py 27/9/2021 3/10/2021

python Hindsight.py 222
```

To run the InitialTraining.py (which for the existing data basically must be run on week 215)

```
cd ./Initial

python InitialTraining.py 15/8/2021
```

Flask

Alternatively, the programme could be used via the Flask application front end:

Navigate to the Flask directory in command line and type

```
export FLASK_APP=Programme.py
```

for mac or

```
$env:FLASK_APP="Programme"
```

for windows

```
flask run
```

and then copy and paste the server link into a browser to launch the Flask application.

Then follow the instructions in the application.

Statistics.csv interpretation

Stat Name	Should Have Been	Predicted	Divided By
'True Positive 1'	Increase	Decrease	Total number of Increase in test data
'True Positive 2'	Decrease	Decrease	Total number of Decrease in test data
'Bad False Positive 1'	Increase	Decrease	Total number of Increase in test data
'Bad False Positive 2'	Decrease	Increase	Total number of Decrease in test data
'False Positive 1'	Normal	Increase	Total number of Normal in test data
'False Positive 2'	Normal	Decrease	Total number of Normal in test data
'False Negative 1'	Increase	Normal	Total number of Increase in test data
'False Negative 2'	Decrease	Normal	Total number of Decrease in test data
'True Negative'	Normal	Normal	Total number of Normal in test data

- Statistics also contain a row called selection which indicates which 'Usage Score' (out of 1 = r^2, 2 = NMI for activity_score, 3 = NMI for actions) the predictor ended up choosing to use; and also a row 'Week' which indicates which week this predictor was produced

Predictions.csv interpretation

- Predictions column:

  Increase: means the client is predicted to significantly increase their usage this week

  Decrease: means the client is predicted to significantly decrease their usage this week

  Normal: means the client is not predicted to significantly increase or decrease their usage this week

  N/A: No prediction made this week, because they had no activity this week. This could either have been because 1) they are no longer a client 2) there is a problem in syncing data somewhere upstream 3) Neither of 1) or 2) has happened but they just haven't used Lucidity in the last week. Please use domain knowledge to aide interpretation

- Usage Score column:

Records whether the client's activity (base score) last week was ranked as low usage, medium usage or high usage. This 'low', 'medium' and 'high' is classified by whether the base score of a client last week sat in the lower 25%, middle 50% or top 25% of the historic weekly usage (that were not 'discarded' for being a tail 0 or initial fluctuation) of all clients[1]

If this row shows N/A then no prediction was made for this client this week (see above)

- Client ID and Client name are used for the ease of joining tables with other BI files later on.

- The last column 'Predictor's Week Number' only contains one value. It records which predictor was used to create this prediction (i.e. the week that the predictor that was used was trained)

Retrospective.csv interpretation

All columns same as Predictions.csv (of the same week) except for the ActualObs column, which is the actual observed usage for that week

Please note if the Predictions column contains something other than 'N/A' but ActualObs contains 'N/A', that is not an error, but just means in the previous week to that week there was activity so a prediction for that week was made, but when that week came by the client had 0 usage

[1] See Data Science Decisions for explanation of discarding records

Data Science Decisions

Below is a brief explanation of the Data Science theory behind the making of this decision model:

*** the PARTs labelled here refers to those from the InitialTraining.py. ContinuousTrain.py may have slightly different code in some of the PARTs but the theory is still exactly the same*

This predictor is based on the Decision Tree (entropy) model.

The raw data was wrangled into the format of counts of use of a particular module per week per company. ***(PART 1, 2, 3)***

Then, as none of the raw data/wrangled raw data provides a readymade usage score, a usage score was created in the following way:

> *Usage Score = f(competency, form, user_inductees) = a*competency + b*form + c * user_inductees*

There were three methods used to get the coefficients a, b, c, with three scores outputted. They were:

1. Taking linear regression of each of {competency, form and user_inductees} against the sum of these three (from now on referenced as a 'preliminary action score'), and accepting the r^2 values of these regressions as the coefficients a, b, c.
2. Taking the Normalised Mutual Information (NMI) score of each of {competency, form and user_inductees} against preliminary action score and using the NMI scores as the coefficients a, b, c
3. Taking the NMI score of each of {competency, form and user_inductees} against the number of 'actions' (from raw data) and using the NMI score as the coefficients a, b, c

- the attributes {competency, form and user_inductees} were chosen for this task because from domain knowledge these are the modules all clients use, and these are the modules that in the designer's opinion would be a good representation of a company's activity.

- (e.g. user_employee wouldn't be a good attribute to use because companies wouldn't typically hire consistently, so they may have lots of activity but most weeks have little addition of employee; risk wouldn't be a good attribute because there is too much intuitive randomness associated with it – a client may have increased their usage of Lucidity that week but did not see any incidents.)

- *the reason for using a self-derived measure of activity level is ultimately due to the lack of data on 'logins per day' (up to the beginning of the project Lucidity has only recently restructured their capturing system of logins from 'last login' only to all*

logins. Altering this score towards using number of logins as a measure of activity could be a priority FUTURE CHANGE when enough data has been collected

Concerns have been raised about using the r^2 coefficient of linear regression between two factors which are clearly dependent. Although this is a problem if the only goal of the linear regression was to determine the relationship between the x and y, here it is merely used to return a coefficient for alternative use. If the concept of using Linear Regression on dependent factors is troubling, then please alter the code to disregard usage_score_1 (although a warning in advance: it may be quite difficult to erase score_1 completely from the code – it will require changes in more than one place)

(PART 4, 5)

Then, any rows of data representing weeks with no activity at all at the 'tail end' of each company's data by week were removed from the analysis (because they are treated as having 'left' the company). (i.e. a company had no activity after week 15 and the current week is week 20, then that company's data in week 15-20 would not be included in the analysis. However, if there was activity on week 20, then all of the company's data would be included. These wiped off rows are referred to as 'tail zeros'. ***(PART 6)***

Data from each company's first 26 weeks where a company's usage score was below its overall (from week 1 up to current week) 0.25 quantile and percentage change above its 0.75 quantile were removed from analysis. This was done because when a company starts using Lucidity there are inherent fluctuations with their usage that shouldn't be captured in the data. Also when usage scores are small, their percentage change in the next week can be so high that it distorts the analysis. Using the 0.25 quantile of its own overall data rather than all company's overall data is because it better accounts for the different sizes of different companies. The magic number of 'first 26 weeks' was chosen because initially the design specification was to wipe off the entire first 26 weeks, but the designer believed a more mathematically/DS rigorous method of selection could be employed to give a better result. Of course, given there are three usage scores and thus three sets of percentage changes, each usage score has a slightly different set of rows that are to be wiped off according to the aforementioned rule. These wiped off rows are referred to as the 'initial fluctuations'. However, if a company hasn't been using Lucidity for 26 weeks then their data is neither used for training or included in the prediction – but this doesn't prevent the data from their first 26 weeks from being used in future predictor trainings once they have been with Lucidity for more than 26 weeks. ***(PART 7)***

Apart from the attributes gained from wrangling the data, the designer also created what he terms 'secondary attributes' for each row of the data: namely the horizontal quantile, vertical quantile and trend.

The horizontal quantile is the quantile which the matching percentage change*[2] of the current

[2] As week 1's score is matched with week 2's percentage change in our data wrangling stage, the

week sits within the set of all its past and present percentage changes, excluding weeks that are wiped off because they are deemed an initial fluctuation (note when calculating the horizontal quantile for week 10, even if we have 20 weeks of data we only include the first 10 weeks in our quantile calculations, provided that none of the first 10 weeks were deemed an initial fluctuation and thus excluded)

The vertical quantile is the quantile which the matching percentage change of this client in this current week sits within the set of all clients' matching percentage change this week, excluding any rows that are wiped off because they are deemed a tail zero or an initial fluctuation.

The trend is the number of weeks in a row which there have been increasing or decreasing. The absolute value of the trend column is the number of weeks which the trend has been sustained, while the sign represents whether this is a decrease or increase ***(PART 8)***

Following the creation of these secondary attributes, any rows with 0 activity for all modules (not a tail 0) were also discarded because in predicting, these would be predicted as 'N/A', so it would make no sense to include them in the training process. However retaining them up to this point was important because these weeks were still part of the pattern of usage in client's history, so they fully deserve to be incorporated into the secondary attributes created above

Then, recognising that changes when the usage score is large is more sensitive to percentage change as opposed to lower scores, it was decided that for each usage score, three decision tree predictors would be created, one for rows which the usage score is in the bottom 25% of that particular usage score, one for rows which the usage score is in the middle 50% of that particular usage score, and one for rows which the usage score is in the top 25% of that particular usage score. (So: each of the usage score would have three predictors that are to be later evaluated in unity compared to the set of the three predictors of the other [two] usage scores) ***(PART 9)***

For each of these groups (i.e. usage score 1, top 25% usage score), rows would be labelled as 'Increase', 'Normal' or 'Decrease' if their percentage change is within the top 5%, middle 95% or bottom 5% of their group. The values of 5% were chosen because the design specification specified a maximum of 10% of companies to be flagged as increasing or decreasing; and while this cannot guarantee that the number flagged will always be under 10%, empirical results suggest it is close enough.

Having now labelled the data, the data is put through a k-fold test-train loop (k = 5 loops and max tree depth set = 12) (with each fold involving fitting decision trees for each of the bottom 25%, middle 50% and top 25%) to calculate an average true positive accuracy score for each of the three predictors. The true positive accuracy scores of the three predictors are then combined into a weighted true positive accuracy score (weighted by proportions of rows in

'matching percentage change' of week 1's score is week 2's percentage change, and hence by induction each week's score's 'matching percentage change' is the percentage change observed in the next week

each group). The weighted accuracy score of the models for the three different usage scores are compared and the two lower ones are eliminated. The choice of k=5 is because it leaves 20% testing, and the depth of 12 is because there are now overall 12 attributes that are available, and with depth = 12 every attribute is given the chance to be used. Note the use of true positive accuracy scores as opposed to just accuracy score, because empirically the true negatives (normal -> normal) is very close to 1 anyway. True positive is also used because the marked (marked as increase or decrease) rows in the test sample only makes up roughly 10%, so what may seem like a very good overall accuracy score may actually be terrible at predicting (e.g. if the predictor just predicted everything as 'normal' it would still have a roughly 90% accuracy, so its difficult to tell whether a 92%-93% accuracy score is really that good) ***(PART 10)***

After the Usage Score that produced the top weighted score is determined, that set then undergoes a loop where the maximum depth of each is varied and each's weighted score is found. Then the system selects the 'maximum depth' which produced the highest weighted true positive accuracy score to be the final tester.

> *A future update could be to do 5-fold testing on each of the depth, but that would be computationally expensive. The reason it was not done here was mainly because it ensures that the model with the highest score out of all the test cases would ultimately be selected. Of course, one may consider this to have too much risk for overfitting, and any changes is up to the discretion of future editors to this programme.*

(PART 11)

<u>Inherent downsides</u>

-because of the lack of initial labelling of data, the designer derived a number of labels from the data for this project, and while they are all mostly mathematical rather than personal judgements, inherently the mathematical choices are personal judgements and thus whether a predicted 'increase' really represents increased usage of the company should be something that is verified once it has been put into use

-scores or labels that were derived:

1. Usage Score (and by transitivity the percentage change)
2. Initial fluctuations
3. The secondary attributes (the designer is confident the theory is right, however the fact it is derived from the Usage Score is what could be a point of concern)

-Overall, the only 'leap of faith' that the designer undertook was creating the usage score – the mathematical analysis following that have carefully avoided making a second 'leap of faith' that would add inherent uncertainty, but ultimately all the subsequent analysis were based off that initial 'leap of faith', and users of this predictor should keep this in mind. However, the beauty of DS is that sometimes computers pick up trends and work really well on things that are unexplainable, so it

may well end up that this model works extremely well.

-another inherent problem is a slight deviation from the design specification, which stated “at maximum, 10% of all companies should be flagged”, implying that if some companies are unworthy of being flagged they should not be flagged, while even if there are enough clients making growth only 10% should be flagged. However, as there were no initial labels, the designer could only try very hard to maintain the latter of the implied requirements, and as roughly 10% of all clients are returned weekly, there may well be companies that shouldn’t have been flagged but are on there due to the model basically returning 10% of companies each week.

-overall, the risks associated with this problem could be mitigated if the BI team uses this as a primary filter tool to decide which client to inspect weekly. They should use it with the domain knowledge (e.g. if it is Christmas/Easter then intuitively not all of the 10% would likely be actual increases or decreases, rather most are just flag to ‘making up the numbers to 10%).

*a problem noted during the developer’s own testing phase, which fell on the 2021 July-September Victorian Covid Lockdowns, was that for weeks the predictor only returned ‘Decrease’ (more than 10%), and no increase. Thus under exceptional circumstances the predictor may flag more than 10% of all clients – and the reason is likely the same as the aforementioned.

<u>Extra notes</u>

-From a CS perspective, throughout the manipulation of the data no rows were ‘deleted’ from the overall data frame. Instead, columns that served as labels for marking data to be removed were employed. This strategy worked particularly well as a large part of the project relied on working with three concurrent sets of scores, and constant deleting invites errors. It is recommended that future editors maintain this style.

-As the most time-costly phase of the InitialTraining.py script is the initial wrangling of raw data (PART 2), repeated wrangling of the full raw dataset in the Predict.py and ContinuousTraining.py were deliberately avoided. Instead, the code was designed so it can wrangle the incremental new data independently and concatenate onto the wrangled version of existing data, saving computational time. However, the run-time of all these scripts depend on the size of the raw data. Please ensure that the incremental data files just contain the data of the relevant weeks and not much more. (i.e. if one placed the full history’s action.csv into one of the .History/Week {}/Data folder then the Predict.py and ContinuousTraining.py script will take as long to run as InitialTraining.py

Future update suggestions

- could try to group in terms of month (mostly change in module 1 to 3 – all of the remaining basically the same) – but you risk losing lots of data and also monthly data could have a lot more noise due to some weeks having 0 input
- If can get interpretation for 'update date'/'delete date', could include those in the initial data wrangling
- could use logins as the measure for activity level once collect enough data on it. This could avoid having to manipulate the data just to get a usage score
- For the dates in PART 2 of the module it may be beneficial to eventually change it to pd.to_date instead of the current setup of different lists. However this may require alteration of the code in PART 1. As the designer ran out of time to change the code to pd.to_date and test it to make sure it runs, this was left unresolved and should be a priority for FUTURE CHANGE

- Places where potential future updates could be made are flagged with '#### FUTURE UPDATE' in the script. Please use ctrl-F to find them and read the suggestions.