Documentation for Client Usage Predictor Model (Linear Regression)

Code and Documentation produced by Lang (Ron) Chen November 2021-February 2021 for Lucidity Software

Date updated: 11/2/2022

Table of contents

Module file structure

- Home directory:
    - Contains ./Initial, ./History, Predict.py, Documentation.docx, Mapping.xlsx, Hindsight.ipynb
    - Also will contain ./Models, ./Standardisers, Statistics.csv, DateMatchWeek.csv and Predictions.csv which are outputs of the ContinuousTrain.py, Predict.py or InitialTraining.py scripts.
        - (./Models stores the prediction .pickle objects which represents the best predictor objects as of yet; Please do not manually alter these. Same for pickles in ./Standardisers)
        - Statistics.csv: contains accuracy score statistics for the predictor
        - 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.
    - Predict.py: script for predicting client usage this week based on new data inputted
    - Hindsight.ipynb 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
    - Also contains other files which are similar to those in home directory, and also ./Partial_Outputs, ./PreparedData which contains files that are created and stored permanently during the training process
- ./History:
    - Contains folders named by week number.
    - Each weekly folder also contains ./Data, ./Partial_Outputs, ./PreparedData, Predictions.csv, Retrospective.csv and PredictionDetails.csv. The first of the three .csv's is the predictions made after running scripts in a particular week (of course, the predictions would be for the next week), whilst the second adds on the actual observations in hindsight for comparison and empirical evaluation. The last is a more detailed version of Predictions.csv which also show the predicted scores

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 inputs (implicit):
    - Initial raw data files need to be stored in directory './Data'.
    - File names must be 'action.csv', 'assets.csv', 'attachment.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'.
    - 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 and PreparedData of Wrangled Data (many .csv's)
- Statistics File (.csv)
- Standardiser and Linear Model .pickles

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

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 inputs (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', 'attachment.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'.
    - 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 and PreparedData of Wrangled Data (many .csv's)
- Predictions and PredictionDetail 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. 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

    *if this 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)

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 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!)

***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

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
cd ..
```

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 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

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

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

Retrospective.csv interpretation

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

Data Science Process

1. Took raw data and collected each client's weekly usage of 7 Lucidity Software modules (Actions, Assets, Competency records, Form records, Form templates, Incident, Users) from 2017.7.3 to 2021.8.15. Each client's weekly usage (with their usage of the 7 modules as attributes) are now instances of data known as Client-Weeks.

2. For each individual client, all the Client-Weeks counting backwards to the most recent week where they did not have any activity would be deleted; once the most recent week is reached where they have used at least one module once, this client-week and all that came before it would be retained (even if there were previous weeks with 0 activity over all 7 modules). This is the 'dropping of end zero weeks'.

   This is because if there is no recent activity it is then assumed that the clients have terminated their usage of Lucidity's services, and thus shouldn't be included in either training or testing (including former clients would affect predictions because the final 'increase' 'decrease' prediction is made by ranking the predicted scores of all clients that week and selecting the top 5% and bottom 5% respectively – thus having irrelevant/dummy clients in the mix would not distort the predictions.)

3. The data of the client usage are normalised – with respect to data within the same module of the same client (i.e. the Actions data of client A would be normalised with respects to all the actions data of client A (after dropping the 'end zero weeks' in step 2) only. This is so to allow comparisons between large clients and small clients.

4. Targets were created for each of the 7 modules for each client-week – all related to the change or percentage change between activity scores in two weeks.

5. A linear regression model would be fitted for each of the 7 module's time-series normalised data vs their targets, and if the r^2 value for this predictor surpassed a pre-determined value, then they are accepted to be used as part of the final model.

6. The final predicted value is a weighted sum of the predicted values out of the 'accepted models'

7. The top 5% of all the predicted scores in a particular week would be predicted as 'increase', whilst the bottom 5% would be predicted 'decrease'

Those labelled in yellow are subject to variations in Data Science decisions. See below

Variations in experiment

The process of data wrangling, target-engineering, and combining the scores into one single predictor allows for many choices to be made; it is impossible to determine which choice is best without actually training them and testing. Utilising modern computational power, scripts were made to train and test models with different combinations of the various Data Science decisions.

Below are the different choices made, and their denotation in the scripts.

1. Rub off the first 26 weeks of all clients?

Because the first 26 weeks of any clients may include volatility (i.e. start-up usage may be small and thus have an effect on normalisation), a reasonable choice is to discard all the Client-Week rows which came from the first 26 self-weeks of a client's usage of Lucidity.

Denotation:
'A': All
'-26': without the first 26 weeks

2. How many 'Pastweeks' to use for time-series data?

Predictions should be done with more than 1 week of data, but how many past-weeks to use is a key question. In the scripts, the range of 1 week to 12 weeks of past data for each module were trailed.

Denotation:
'1': 1 past week
'2': 2 past weeks
…

3. How to make each module's target for each Client-Week?

Three options were used:

A. The difference between the normalised scores of each module each week
Denotation: "S_D"

B. The same as 1 except instead of using change as the target, it uses the quantile of this Client-Week's change with respect to all Client-Weeks of the same week. (i.e. what quantile does this client's change sit within all of Lucidity's client's change this week)
Denotation: "Q_D"

C. The same as 2, except instead of using the difference of normalised scores of each module each week as its underlying score (like how 2 uses 1 as its underlying score), it uses percentage change between the two normalised scores

*However, for %change, used special formula:

$$\%change = (next\ week\ normalised\ use - this\ week\ normalised\ use)/abs(this\ week\ normalised\ use)$$

because normalised data could be negative, so if abs() is not used for the denominator, could result in an increase being assigned a negative %change value.

i.e. next week's score is pos, this week's score is neg, but +/- = -, so an increase of usage leads to a negative/decreasing percentage change

Denotation: "Q_P"

4. What to use as past weeks

A. Use past week's normalised observations

B. Use past week's targets as the attributes for all past weeks except the current week
Explanation: we won't have this week's %change until next week.

Part A was used for all three target engineering methods in 3, whilst B was only used for part 3's B and C.

Denotation: "Q(S)_D"; "Q(S)_P"

5. Cutoff value

AKA Feature Selection Cutoff Value (fs_val), it will determine whether or not to 'accept' each linear regression model based on their r^2 value.

Cutoff value choices: 0.2, 0.3, 0.4

6. How to weigh the scores in the final weighted sum.

1. No weighting (all scaled by 1)

2. Scaled by the r^2 value of the linear regression model for that particular module

Denoted: 'r^2' and '1'

*note that 'r' was mistakenly used some of the experiments – it should be ignored

Train-test split and determining best model

A Five-Fold Split (each 80%-20%) were made for each type of data manipulation. The splits were done in terms of weeks rather than Client-Weeks because the final prediction is made in terms of whole weeks of Client-Weeks.

For each of these combinations of choices and their splits, the TP1 - true positive 1 value (#correctly predicted positives/#predicted positive) and TP2 - true positive 2 value (#correctly predicted negatives/#predicted negative) was recorded in a results csv that the script outputted. The average of TP1 and TP2 gives the overall TP - true positive value for this particular way to train the model

The best model would be selected by taking the average of the TP values of all 5 of its splits, and selecting the one with the top TP. Note if any split couldn't return a result because none of its module Linear Regression Models yielded r^2 values above the fsval, then it would not be considered at all.

Calculating the combinations, 720 different models were trained and tested, and taking into account the five splits meant 3600 models trained and tested – all done automatically via script.

The final best model was 'S_D' '-26' '11 prev weeks' 'fsval = 0.4' , which incorporated the accepted models of Assets, Form Template, Incident, Users.
A different set of data was train-test split for training this final model, and it had a TP value of 0.5*(0.2342857 + 0.72) = 47.7%

It's Bad False Positive score (average of false positive where predicted increase but was actually decrease, and predicted decrease but was actually increase, was 2.9157%. Whilst this is not ideal, technically Lucidity would look into all clients that are predicted to increase and decrease – so whilst it may be a surprise for the Bis when they monitor a decrease which actually increases, nonetheless it served the effect of getting them to monitor it ☺

Index	Statistics
True Positive 1	0.2342857142857143
True Positive 2	0.72
Bad False Positive 1	0.0057142857142857…
Bad False Positive 2	0.05714285714285714

The Model does tend to predict significantly more accurately for decreases – likely due to the inherent nature of the data – see Lucidity Data Report

Advantage compared to other methods (i.e. engineer own score)

This takes into account the rise and fall in usage of (supposedly) each module as opposed to one overall engineered score – which may not capture all the dynamics of the usage of the clients. It is perhaps the best direction for model training given Lucidity does not have a statistic (such as logins) that can be used specifically to measure client usage.

Statistically, it defeated all 'engineered scores' models.

Downsides

The final model only utilises data from 4 modules, so if in one Client-Week the Client uses all other modules but not these four, it theoretically leads to a poorer performance. However, as normalisation moves '0's to a non zero value, thus avoiding a lot of the issues associated with this number, it practically shouldn't have much effect.

> *note that although it only uses four attributes, users must continue download the weekly data for all 7 modules into the correct directory! This is because the other three modules are still being used to determine whether there was activity by this client at all (if a client-week only used other modules in a particular week but not the four 'predictive modules' and the predictor didn't have the data of the 3 non-predictive modules, it would regard that Client-Week as an 'end zero week' and make no predictions for it at all

Also, hindsight performance is delayed by one week compared to previous decision tree model – because this time we use weekly percentage change of different modules as our 'observed target', and thus require one more week to have the right data.

Future recommendations

1. Percentage change: Don't use it – this method caused all sorts of issues and most of the times none of the module Linear Regression models surpassed 0.2.

2. Order of Train-Test Split and Normalisation: Perform split before normalise when conducting this experiment, one mistake was normalising the data before test-train splitting. This means that the test data could fit better than future empirical data, and thus the test results are better than what they really are (overfit)

3. Three-way split: since used r^2 as part of the prediction, should have done a three way train-verify-test split (70%-15%-15%) and use middle 15% to calculate the r^2 to prevent overfit. However, this could have issues for comparison of the TP values of 'r2' and '1' models

4. Empirical Evaluation: even though has 47.7% accuracy, not very sure whether even the observed values are correct (i.e. whether the 'observed increases' are really increases in real business setting).

5. Continuous Training: Test results demonstrated that basically 'S_D', '-26' '11/12 past weeks', 'fsval = 0.4' gave the best TP results. Thus theoretically could implement a continuous training scheme where data of each week is combined with existing data to re-train just these two models and use the best to replace the existing model – if the best new model can surpass the existing.

At this stage, forecasting (predicting beyond one week) is not recommended. See Lucidity Data Report