PIDGINv3 Documentation

Release v0.1beta

Lewis H. Mervin

May 06, 2020

Contents

Release v0.1beta

Date May 06, 2020

Author : Lewis Mervin, lewis.mervin@cantab.net

Supervisor : Dr. A. Bender

CHAPTER 1

Contents

1.1 Overview of PIDGINv3

1.1.1 Introduction

Protein target prediction using Random Forests (RFs) trained on bioactivity data from PubChem (extracted 07/06/18) and ChEMBL (version 24), using the RDKit and Scikit-learn, which employ a modification of the reliability-density neighbourhood Applicability Domain (AD) analysis by Aniceto[1]. This project is the sucessor to PIDGIN version 1[2] and PIDGIN version 2[3]. Target prediction with extended NCBI pathway and DisGeNET disease enrichment calculation is available as implemented in[4].

- Molecular Descriptors : 2048bit RDKit Extended Connectivity FingerPrints (ECFP)[5]
- Algorithm: Random Forests with dynamic number of trees (see docs for details), class weight = 'balanced', sample weight = ratio Inactive:Active
- Models generated at four different cut-off's: 100μM, 10μM, 1μM and 0.1μM
- Models generated both with and without mapping to orthologues
- Pathway information from NCBI BioSystems
- Disease information from DisGeNET
- Target/pathway/disease enrichment calculated using Fisher's exact test and the Chi-squared test

Details for sizes across all activity cut-off's:

[1] Aniceto, N, et al. A novel applicability domain technique for mapping predictive reliability across the chemical space of a QSAR: Reliability-density neighbourhood. *J. Cheminform.* **8**: 69 (2016)

[2] Mervin, L H., et al. Target prediction utilising negative bioactivity data covering large chemical space. *J. Cheminform.* **7**: 51 (2015)

[3] Mervin, L H., et al. Orthologue chemical space and its influence on target prediction. *Bioinformatics.* **34**: 72–79 (2018)

[4] Mervin, L H., et al. Understanding Cytotoxicity and Cytostaticity in a High-Throughput Screening Collection. *ACS Chem. Biol.* **11**: 11 (2016)

[5] Rogers D & Hahn M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**: 742-54 (2010)

	Without orthologues	With orthologues
Distinct Models	10,446	14,678
Distinct Targets [exhaustive total]	7,075 [7,075]	16,623 [60,437]
Total Bioactivities Over all models	39,424,168	398,340,769
Actives	3,204,038	35,009,629
Inactives [Of which are Sphere Exclusion (SE)]	36,220,130 [27,435,133]	363,331,140 [248,782,698]

Full details on all models are provided in the uniprot_information.txt files in the ortho and no_ortho directories (to be downloaded)

1.1.2 Contributing

Development occurs on GitHub. Documentation on Readthedocs. Contributions, feature requests, and bug reports are welcome. Consult the issue tracker.

1.1.3 License

PIDGINv3 is released under the GNU Lesser General Public License version 3.0 ().

Broadly, this means PIDGINv3 can be used in any manner without modification, with proper attribution. Modification of source code must also be released under so that the community may benefit.

1.1.4 Citing PIDGIN

To cite PIDGINv3, please reference either previous versions[23] or use .

1.1.5 References

1.2 Setup and Installation

Development and documentation occurs on GitHub.

PIDGIN is currently only compatible with Python 2.7.x. It also has the following dependencies:

1.2.1 Required dependencies

- NumPy
- SciPy
- RDKit
- Scikit-learn
- Standardiser
- python_utilities

1.2.2 Install with Conda

Follow these steps on Linux/OSX:

1. `Download and install Anaconda2 for Python 2.7 from https://www.continuum.io/downloads`
2. Open terminal in Mac/Linux and run `conda create -c rdkit -c conda-forge --name pidgin3_env python=2.7 rdkit scikit-learn=0.19.0 pydot graphviz standardiser statsmodels`

- N.B. Rdkit may not import on some systems due to a bug. If this happens upgrade to the latest version of conda before creating the above environment using: `conda update conda`
- N.B. Installs the IMI eTOX flatkinson standardiser (replaces ChemAxon's standardizer used in previous PIDGIN versions) and statsmodels for p-value correction in predict_enriched.py

3. Now run: `source activate pidgin3_env` (This activates the PIDGINv3 virtual environment. N.B This is required for each new terminal session in order to run PIDGIN in the future)
4. Navigate the directory you wish to install PIDGINv3 and in Mac/Linux terminal run `git clone https://github.com/lhm30/PIDGINv3/` (recommended) or download/extract the zip from GitHub webpage (not recommended due to inability to pull updates)
5. (10GB) Download and unzip no_ortho.zip (md5sum: af0fd520de846c3cddcaec74dad4241d) into the PIDGINv3 main directory (leave all subsequent files compressed)
6. (optional 24GB) Models are also available when mapping data between orthologues, as in[1]. N.B The files are 24GB and many models are based solely on orthologue data. To include this functionality, download ortho.zip (md5sum: 8f4e4a76f1837613ec4a3dd501d55753) to the PIDGINv3 main directory and unzip ortho.zip (leave all subsequent files compressed)

- N.B Depending on bandwidth, Step 5/6 may take some time

1.2.3 Filetree structure

Once the models are downloaded and the main zip uncompressed, you should find the following filetree structure within the PIDGINv3 directory (located for this snippet at `$PV3`) if both the optional orthologs (ortho) and models without orthologs (no_ortho) files are used:

```
$PV3 tree -L 2
.
├── biosystems.txt
├── DisGeNET_diseases.txt
├── docs
│   ├── conf.py
│   ├── dev
│   ├── index.rst
│   ├── install.rst
│   ├── make.bat
│   ├── Makefile
│   ├── overview.rst
│   ├── substitutions.rst
│   └── usage
├── examples
│   ├── test2.smi
│   └── test.smi
```

(continues on next page)

[1] Mervin, L H., et al. Orthologue chemical space and its influence on target prediction. *Bioinformatics.* **34**: 72–79 (2018)

(continued from previous page)

```
├── LICENSE
├── nontoxic_background.csv
├── no_ortho
│   ├── ad_analysis
│   ├── bioactivity_dataset
│   ├── pkls
│   ├── training_log.txt
│   ├── training_results
│   └── uniprot_information.txt
├── no_ortho.zip
├── ortho
│   ├── ad_analysis
│   ├── bioactivity_dataset
│   ├── check_ad2.py
│   ├── check_ad.py
│   ├── pkls
│   ├── training_log.txt
│   ├── training_results
│   └── uniprot_information.txt
├── ortho.zip
├── predict_enriched.py
├── predict.py
└── README.rst
```

1.3 Usage and Examples

To facilitate the use of PIDGINv3, examples of usage are provided below, ordered by expected frequency of use and increasing complexity.

1.3.1 Command Line Arguments

PIDGINv3 uses a Command Line Interface (CLI) for all available functionality.

This tutorial assumes the PIDGINv3 repository is located at `$PV3`.

List of available arguments

To see all available options, run

```
$ python $PV3/predict.py -h
Usage: predict.py [options]

Options:
  -h, --help            show this help message and exit
  -f FILE               Input smiles or sdf file (required)
  -d DELIM, --smiles_delim=DELIM
                                                Input file (smiles) delimiter char␣
↪(default: white
                                                space ' ')
  --smiles_column=SMICOL
                                                Input file (smiles) delimiter column␣
↪(default: 0)
```

(continues on next page)

(continued from previous page)

```
  --smiles_id_column=IDCOL
                                                Input file (smiles) ID column␣
↪(default: 1)
  -o FILE               Optional output prediction file name
  -t, --transpose       Transpose output (rows are compounds, columns are
                                                targets)
  -n NCORES, --ncores=NCORES
                                                No. cores (default: 1)
  -b BIOACTIVITY, --bioactivity=BIOACTIVITY
                                                Bioactivity threshold (can use␣
↪multiple split by ','.
                                                E.g. '100,10'
  -p PROBA, --proba=PROBA
                                                RF probability threshold (default:␣
↪None)
  --ad=AD               Applicability Domain (AD) filter using percentile of
                                                weights [float]. Default: 90 (integer␣
↪for percentile)
  --known_flag          Set known activities (annotate duplicates betweem
                                                input to train with correct label)
  --orthologues         Set to use orthologue bioactivity data in model
                                                generation
  --organism=ORGANISM   Organism filter (multiple can be specified using
                                                commas ',')
  --target_class=TARGETCLASS
                                                Target classification filter
  --min_size=MINSIZE    Minimum number of actives used in model generation
                                                (default: 10)
  --performance_filter=P_FILT
                                                Comma-seperated performance filtering␣
↪using following
                                                nomenclature: validation_set[tsscv,
↪l50so,l50po],metric
                                                [bedroc,roc,prauc,brier],performance_
↪threshold[float].
                                                E.g 'tsscv,bedroc,0.5'
  --se_filter           Optional setting to restrict to models which do not
                                                require Sphere Exclusion (SE)
  --training_log        Optional setting to add training_details to the
                                                prediction file (large increase in␣
↪output file size)
  --ntrees=NTREES       Specify the minimum number of trees for warm-start
                                                random forest models (N.B Potential␣
↪large
                                                latency/memory cost)
  --preprocess_off      Turn off preprocessing using the flatkinson (eTox)
                                                standardizer (github.com/flatkinson/
↪standardiser),
                                                size filter (100 >= Mw >= 1000 and␣
↪organic mol check
                                                (C count >= 1)
  --std_dev             Turn on matrix calculation for the standard deviation
                                                of prediction across the trees in the␣
↪forest
  --percentile          Turn on matrix calculation for the percentile of AD
```

Detailed explanations for the more complicated arguments

SMILES options (-d / –smiles_column / –smiles_id_column)

PIDGINv3 interprets SMILES files (`*.smi` or `*.smiles`) using the conventional OpenSMILES specification §4.5), comprising a first column of smiles separated by a white line ( ) character and additional entries as identifiers.

An example of such a file is included in the examples directory for the SMILES file named `test.smi`, containing two molecules whose SMILES strings are defined as:

Listing 1: test.smi

```
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)cc5c2cc1OC CompoundID1
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)c(OC)c5c2cc1OC CompoundID2
```

The following arguments can alter this behaviour, if desired, to accomodate for different file strcutures:

- `-d` (or `--smiles_delim`)
- `--smiles_column`
- `--smiles_id_column`

For example `test2.smi` contains a comma separated file whose first column is the ID and SMILES in the second column.

Listing 2: test2.smi

```
Input1,CCCCOc1ccc2c3ccnc(\C=C\c4ccc(OC)cc4)c3n(CCCc5ccccc5)c2c1
Input2,COc1ccc(\C=C\c2nccc3c4ccccc4n(CCCc5ccccc5)c23)cc1
Input3,CCCCn1c2ccccc2c3ccnc(\C=C\c4ccc(OC)cc4)c13
Input4,COc1cc(\C=C\c2nccc3c4ccccc4[nH]c23)cc(OC)c1OC
Input5,COc1ccc(\C=C\c2c3c(cc[n+]2Cc4ccccc4)c5ccccc5n3CCCc6ccccc6)cc1
Input6,CCCCn1c2ccccc2c3ccnc(C)c13
Input7,Cc1nccc2c3ccccc3n(CCCc4ccccc4)c12
Input8,COc1ccc(\C=C\c2nccc3c4ccccc4[nH]c23)cc1
Input9,C(=C\c1nccc2c3ccccc3[nH]c12)/c4ccccc4
Input10,Cc1nccc2c3ccccc3[nH]c12
```

Thus the following command should be used when running any commands:

```
$ python $PV3/predict.py -f test2.smi -d ',' --smiles_column 1 --smiles_id_column 0
```

Note: PIDGINv3 generates a warning message for any user input files which are neither `*.smi` / `*.smi` or `*.sdf`, and will interpret any other file as a SMILES.

Transpose options (-t)

Transposes the prediction matrix from rows as targets and columns as compounds, to rows are columns and columns as compounds.

Note: This will remove the metadata for each target (just the Uniprot name will be used) to ensure only one column header is used.

RF probability theshold (-p)

The continuous probabilities from each model [p(activity)] for input compounds can be converted into binary predictions at a user-specified threshold. The choice of required p(activity) indicates a degree of confidence in predictions when binarizing probabilities.

Note: These probabilities are different from PIDGIN **'version 2'_** in that they have not been Platt-scaled, since this increased the number of false positives.

Applicability domain threshold (–ad / –percentile)

PIDGINv3 applies the reliability-density neighbourhood Applicability Domain (AD) analysis by Aniceto et al., from: doi.org/10.1186/s13321-016-0182-y.

In this procedure, three parameters are calculated on a per-compound basis across the training data for each target model. 1.) The nearest-neighbour similarity (`sim`) [the largest Tanimoto Coefficient (Tc) similarity] to all data points for the target model 2.) The RF probability of activity for the true label of the training compound (i.e. the probability of being active for an active compound or the inactivity prediction for an inactive compound) for the realised models (`bias`). 3.) The standard deviation (`std_dev`) of this probability calculation, computed by the deviation of predictions across all trees in the forest (this metric is considered a level prediction certainty). These values are used to compute the weights (`w`) for each training compound instance using the following equation:

Note: w = sim / (bias * std_dev)

Reliability increases with the increase of `w`, whereby higher reliability is associated with higher similarity and low `bias * std_dev`. In practice, this procedure penalizes high similarity which is associated with poor bias and precision observed in the trained model.

At run time, the user specifies the cut-off for applicability domain (AD) percentile (n) required for input compounds, using the following command:

- `--ad`

where `int` or (n) is a number between 0-100. In this case, the corresponding threshold encapsulating n% of the pre-computed weights is calculated (i.e. n-th percentile of `w` values). Weights are next calculated on a per-input compound basis by calculating the nearest neighbour similarity to the training set and identifying the corresponding (pre-computed) training compound bias and std_deviation for the near neighbour. The corresponding percentile value for the input compound is calculated in the same manner as above. A percentile value for the input compound above the user-specified percentile threshold means the compound is considered within the applicability domain given the user-specified conditions, and the corresponding probability of activity [p(activity)] (or the binary prediction, if specified) is written in the prediction matrix. Conversely, a weight below the percentile means an input compound is outside the AD, and in this case an `NaN` (not available) is added to the output matrix.

Note: Higher confidence in the applicability domain (larger n) will increase run-time or latency, since the code will quit looping through training upon identifying a compound within the AD.

This feature can be effectively turned off by specifying the following command (not recommended):

```
$ python $PV3/predict.py -f test2.smi -d ',' --smiles_column 1 --smiles_id_column 0 --
↪ad 0
```

If a user would like to obtain a matrix comprising the percentile weights for each of the input compounds, then the command line argument `--percentile` can be used.

Annotating known activity in ChEMBL or PubChem (–known_flag)

Known actives from ChEMBL and the inactives used from PubChem (possibly only a subset due to undersampling) can be annotated using the command:

- `--known_flag`

> **Note:** This requires the full matrix of similarities between input and training compounds to be computed, and hence increases computational cost/latency.

Filtering the models by pre-calculated performance (–performance_filter)

Leave 50% of the random scaffold out (L50SO) and 50% of the ChEMBL publication (from which the bioactivity data has been extracted) ID's out (L50PO) was performed over 4-splits for the training data as a validation set. The data was also split using time-series split validation (TSSCV). The ROC, BEDROC, Precision-recall curve (PR-AUC) and Brier score were computed over the folds and stored in the file training_log.txt in either the ortho or no_ortho directories. This data can be incorporated into the output prediction file for use as a desired performance value using the command:

- `--performance_filter`

where the user should supply comma-seperated performance filtering using following nomenclature: validation_set[tsscv,l50so,l50po], metric[bedroc,roc,prauc,brier], performance_threshold float].

For example the following command would provice predictions for the models with a BEDROC of 0.5 during TSSCV:

```
$ python $PV3/predict.py -f test.smi --ad 0 --performance_filter tsscv,bedroc,0.5
```

Incorporating training log with predictions (–training_log)

The results from the above analysis can be appended to the target information columns to provide detailed information to the user for filering. This will increase the file size due to the significant amount of data. The column headings have the following meanings:

- `MODEL_ID`: ID of the model (nomenclature defined as 1. the Uniprot IDs annotated in the active/inactive training set, 2. if Sphere exclusion (SE) has been used and 3. an underscore followed by the threshold for activity)
- `FINGERPRINT`: Type of molecular fingerprint used
- `N_TREES`: Number of trees in the forest (differs depending on the tree optimisation)
- `TRAIN_TIME_SEC`: Time taken to train the RF model
- `OUTOFBAG_ERROR_2DP`: Out-of-bag (OOB) score for the RF (Sklearn calculated)
- `N_SCAFFOLDS`: Number of generic Murcko scaffolds within the chemistry of training data
- `N_PUBLICATIONS`: Number of ChEMBL publications across training data

Time-series split (TSSCV) validation, followed by the metric used and the average/median/standard dev. over the 4 folds:

- TSCV_BEDROC_AVG_MED_STD
- TSCV_ROC_AVG_MED_STD
- TSCV_PRAUC_AVG_MED_STD
- TSCV_BRIER_AVG_MED_STD
- TSCV_TRAIN_SPLIT_SIZES

Leave 50% of publications out, followed by the metric used and the average/median/standard dev. over the 4 folds:

- L50PO_BEDROC_AVG_MED_STD
- L50PO_ROC_AVG_MED_STD
- L50PO_PRAUC_AVG_MED_STD
- L50PO_BRIER_AVG_MED_STD
- L50PO_TRAIN_SPLIT_SIZES

Leave 50% of scaffolds out, followed by the metric used and the average/median/standard dev. over the 4 folds:

- L50SO_BEDROC_AVG_MED_STD
- L50SO_ROC_AVG_MED_STD
- L50SO_PRAUC_AVG_MED_STD
- L50SO_BRIER_AVG_MED_STD
- L50SO_TRAIN_SPLIT_SIZES

The training data was then used to benchmark the realised models using all data, to obtain the following metrics:

- TRAIN_V_TRAIN_BEDROC
- TRAIN_V_TRAIN_ROC
- TRAIN_V_TRAIN_PRAUC
- TRAIN_V_TRAIN_BRIER

The average and standard deviations across all probabilities of activity [p(activity)] for each of the actives and inactivity [p(inactivity)] for all inactives were recorded for compared to the realised models:

- INACTIVES_AVG_STD_PRED
- ACTIVES_AVG_STD_PRED

Increasing the number of trees (–ntrees)

Thanks to the warm_start function of Scikit-learn RF's, the number of trees in the forests can be globally increased (at the cost of latency/increased CPU) using the command:

- --ntrees

Turning off pre-processing (–preprocess_off)

PIDGINv3 implements a pre-processing feature which is turned on by default, to ensure all input molecules are standardised using the flatkinson (eTox) standardiser (github.com/flatkinson/standardiser), and that any molecules outside the applicability domain of the models, defined by the chemical property filters imposed on ChEMBL and PubChem training data [size filter 100 >= Mw >= 1000 / organic mol check (Carbon count >= 1)] are removed. This

functionality can be turned off to force PIDGINv3 to give unreliable predictions (in cases when the input space maybe outside the domain of applicability or when molecules have been pre-standardised) using the following command:

- `--preprocess_off`

Output the standard dev. of predictions across the trees (–std_dev)

The standard deviation of the predictions across the trees can be output to the prediction matrix (in place of the probability for activity), using the following command:

- `--std_dev`

1.3.2 Getting started

This tutorial assumes the PIDGINv3 repository is located at `$PV3`.

Generating predictions for human targets

In this example, we will work with the input file named `test.smi` in the examples directory, which containins two molecules whose SMILES strings are defined as:

Listing 3: test.smi

```
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)cc5c2cc1OC CompoundID1
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)c(OC)c5c2cc1OC CompoundID2
```

The following code will generate the RF probabilities at 1μM for all human targets for the input file:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 1
```

This script outputs the RF output from each of the Random Forest classifiers across the targets for the all compounds into a probability matrix, where the columns are compounds and the rows are targets.

If using `--organism`, it must be as specified in the uniprot_information.txt and if using spaces enclosed by quotes (“”) - as in the above example. The organism filter uses fuzzy matching, so `--organism homo` would also achieve a similar filtered list.

Generating binary predictions

The following code will generate binary predictions at 0.1 and 1μM for all human targets, at a threshold of 0.5 (the compound was more often predicted active compared to inactive):

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 0.1,1 -p 0.5
```

The threshold can be increased to increase the confidence in the prediction.

Note: These probabilities are different from PIDGIN version 2 in that they have not been Platt-scaled, since this increased the number of false positives.

Decreasing applicability domain (AD) filter

To reduce the stringency in the AD filter, the `--ad` parameter (defulat:90) can be reduced, as in the following snippet:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 1 -p 0.5 --ad 60
```

In this case, the threshold for the applicability domain weights calculated across the targets has been reduced from 90% to 60%, and thus compounds that are further from the AD are now accepted.

Outputting the AD results

To following snippet calculates the weights for each of the input compounds and outputs their corresponding percentile value, so that a user can view the matrix of percentiles for each compound and accept/reject predictions at a percentile threshold without the need to re-run predictions:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 1 --percentile
```

Silencing the AD filter

To following snippet would therefore turn off the AD filter, since all predictions are accepted:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 1 --ad 0
```

Combining model filters

If the user is interested in a given target class (for example "Lipase") then the following can be used:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" --target_class Lipase
```

Filters can be combined, for instance:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" --target_class GPCR --
↪min_size 25 --performance_filter tsscv,prauc,0.7
```

would filter human models for GPCRs with a minimum number of 25 actives in the training set and with a minimum precision-recall AUC (PR-AUC) performance of 0.7 during time-series split cross validation (TSSCV).

Additional criteria can be added, for instance:

```
$ python $PV3/predict.py -f test.smi --organism "Rattus" -b 0.1,1 -p 0.5 --min_size␣
↪50 --se_filter --performance_filter l50po,bedroc,0.8
```

would filter rat models that did not require Sphere Exclusion (SE) (i.e. sufficient number of inactives available) and a minimum number of 50 actives in the training set, with a minimum BEDROC performance of 0.8 during leave 50% of ChEMBL publications in the training data out over 4-fold cross validation (L50PO) to produce a binary matrix of predictions at a probability cut-off of 0.5 and for models trained with bioactivity data at a threshold of 0.1 & 1.01μM.

1.3.3 Extended functionality

This tutorial assumes the PIDGINv3 repository is located at `$PV3`.

The input file named `test.smi` is used for these examples

Listing 4: test.smi

```
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)cc5c2cc1OC CompoundID1
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)c(OC)c5c2cc1OC CompoundID2
```

Generating transposed predictions

The following code will output the RF probabilities at 10μM for all human targets to a transposed file:

```
$ python $PV3/predict.py -f test.smi --organism "Homo sapiens" -b 10 --transpose
```

This script outputs the RF output from each of the Random Forest classifiers across the targets for the all compounds into a probability matrix, where the rows are compounds and the columns are targets.

Increasing trees and getting the standard dev. for input compounds

The following snippet will increase the minimum number of RF trees to 250 for all 0.1μM ligase targets and then calculate the standard deviation of the predictions across the 250 trees in the forests across the filtered targets:

```
$ python $PV3/predict.py -f test.smi --ntrees 250 --target_class Ligase --std_dev
```

Note: The max number of trees when generating the models was set to 250. An algorithm to search for the optimal trees was performed as follows: 1. start at 90 trees and calculate the out-of-bag error (OOB) for the forest. 2. Increase the trees by 10 and calculate difference in OOB score. 3. Repeat until 1 minute of train time is reached or there was no performance gain on two trees incement occasions (test for convergence) or a maximum of 250 trees is reached.

Annotating predictions with known activity

The probabilities output are clipped between `0.001` and `0.999`, so that a perfect score of 0.0 and 1.0 is not obtained from the RFs. This behaviour affords the explicit annotation of duplicate bioactivity data between input compounds and the training set by specifying known inactives with a score of `0.0` and actives with `1.0`. To activate this functionality use the following snippet:

```
$ python $PV3/predict.py -f test.smi --organism Drosophila -b 100 --known_flag
```

which would provide predictions for all Drosophila targets with a 100μM cut-off, and would calculate overlap between input compounds and the training set and annotate these instead of providing predictions.

Note: This setting increases latency since every input compound has to be compared for perfect Tanimoto coefficient (Tc) similarity of `1.0` against every training compound.

1.3.4 Enrichment Predictions

This tutorial assumes the PIDGINv3 repository is located at `$PV3` and is concerned with the script `predict_enriched.py`

This script calculates target prediction enrichment (using Fishers' t-test) between two input SMILES/SDF files as in[1]. Target predictions are extended with NCBI Biosystems pathways and DisGeNET diseases. Pathway or disease-gene association enrichment (using chi-square test) enrichment is calculated for the two input SMILES/SDF files.

The approach is used to annotate which targets/pathways/diseases are statistically associated between two compound sets given their input SMILES/SDF files. This analysis is important since a (predicted) target is not necessarily responsible for eliciting an observed mechanism-of-action. Some target prediction models also behave promiscuously, due to biases in training data (chemical space) and the nature of the target.

The analysis must use a cut-off for the probability of activity from the random forest for each target. Predictions are generated for the models using the reliability-density neighbourhood Applicability Domain (AD) analysis by Aniceto from: doi.org/10.1186/s13321-016-0182-y

`biosystems.txt` contains pathway data from the NCBI biosystems used to annotate target predictions. Pathway results can be filtered by source (e.g. KEGG/Reactome/GO) afterward.

`DisGeNET_diseases.txt` contains disease data used to annotate target predictions. DisGeNET gene-disease score takes into account the number and type of sources (level of curation, organisms), and the number of publications supporting the association. The score ranges from 0 to 1 in accordance to increasing confidence in annotations, resepctively. A DisGeNET_threshold can be supplied at runtime when annotating predictions with diseases (0.06 threshold applied by default, which includes associations from curated sources/animal models supporting the association or reported in 20-200 papers). More info on the score here: http://disgenet.org/web/DisGeNET/menu/dbinfo#score

List of available arguments

To see all available options, run

```
$ python $PV3/predict_enriched.py -h
Usage: predict_enriched.py [options]

Options:
  -h, --help            show this help message and exit
  --f1=FILE             Firest input smiles or sdf file (required)
  --f2=FILE             Second input smiles or sdf file (required)
  -d DELIM, --smiles_delim=DELIM
                                                Input file (smiles) delimiter char␣
↪(default: white
                                                space ' ')
  --smiles_column=SMICOL
                                                Input file (smiles) delimiter column␣
↪(default: 0)
  --smiles_id_column=IDCOL
                                                Input file (smiles) ID column␣
↪(default: 1)
  -o FILE               Optional output prediction file name
  -n NCORES, --ncores=NCORES
                                                No. cores (default: 1)
  -b BIOACTIVITY, --bioactivity=BIOACTIVITY
                                                Bioactivity Um threshold (required).␣
↪Use either
                                                100/10/1/0.1 (default:10)
  -p PROBA, --proba=PROBA
                                                RF probability threshold (default:␣
↪None)
  --ad=AD               Applicability Domain (AD) filter using percentile of
```

(continues on next page)

[1] Mervin, L H., et al. Understanding Cytotoxicity and Cytostaticity in a High-Throughput Screening Collection. *ACS Chem. Biol.* **11**: 11 (2016)

(continued from previous page)

```
                                                weights [float]. Default: 90 (integer␣
↪for percentile)
  --known_flag          Set known activities (annotate duplicates betweem
                                                input to train with correct label)
  --orthologues         Set to use orthologue bioactivity data in model
                                                generation
  --organism=ORGANISM   Organism filter (multiple can be specified using
                                                commas ',')
  --target_class=TARGETCLASS
                                                Target classification filter
  --min_size=MINSIZE    Minimum number of actives used in model generation
                                                (default: 10)
  --performance_filter=P_FILT
                                                Comma-seperated performance filtering␣
↪using following
                                                nomenclature: validation_set[tsscv,
↪l50so,l50po],metric
                                                [bedroc,roc,prauc,brier],performance_
↪threshold[float].
                                                E.g 'tsscv,bedroc,0.5'
  --se_filter           Optional setting to restrict to models which do not
                                                require Sphere Exclusion (SE)
  --training_log        Optional setting to add training_details to the
                                                prediction file (large increase in␣
↪output file size)
  --ntrees=NTREES       Specify the minimum number of trees for warm-start
                                                random forest models (N.B Potential␣
↪large
                                                latency/memory cost)
  --preprocess_off      Turn off preprocessing using the flatkinson (eTox)
                                                standardizer (github.com/flatkinson/
↪standardiser),
                                                size filter (100 >= Mw >= 1000 and␣
↪organic mol check
                                                (C count >= 1)
  --dgn=DGN_THRESHOLD   DisGeNET score threshold (default: 0.06)
```

Generating enrichment predictions

In this example, we will work with a two SMILES input files, comprising cytotoxic compounds in the file named `cytotox_library.smi` and (putative) non-toxic compounds in the file named `nontoxic_background.smi`. Both are located in the examples directory.

The corresponding top 5 SMILES strings are:

Listing 5: cytotox_library.smi

```
C1Cc2cc3OCOc3c4c5ccccc5C[C@@H](N1)c24
[Cl-].COc1ccc2cc3c4cc5OCOc5cc4CC[n+]3cc2c1OC
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)cc5c2cc1OC
COc1cc2c3CN4CCC[C@H]4[C@@H](O)c3c5ccc(O)c(OC)c5c2cc1OC
C[C@H](CCC(=O)[C@@H](C)[C@H]1[C@H](C[C@H]2[C@@H]3CC[C@H]4C[C@H](CC[C@]4(C)[C@H]3CC(=O)[C@]12C)OC(=O)C
```

and

Listing 6: nontoxic_background.smi

```
CC12CC3CC(C)(C1)CC(C3)(C2)NCc1ccccc1OCc1ccccc1F
CCCn1ccnc1C(=O)c1cc2ccccc2s1
Cc1nn(C)c(C)c1S(=O)(=O)NC1CCCCC1N
CC(C)CN(CC(C)C)C(=O)c1ccc(nc1C)C(F)(F)F
CC1CCc2c(C1)sc1nc(CN3CCOCC3)nc(Oc3ccccc3F)c21
```

The following code will generate cow target prediction enrichment at 1μM (with lenient AD filters of 30 percentiles and probability of activity cut-off of 0.45) along with enriched pathways and diseases (0.06 score threshold) for the cytotoxic compounds, when compared to the non-toxic compounds.

```
$ python $PV3/predict_enriched.py --f1 cytotox_library.smi --f2 nontoxic_background.
↪smi --organism "Bos taurus" -b 1 -p 0.45 --ad 30 -n 4
```

Three files are output for the target, pathway and disease enrichment calculations, with the naming convention:

`[f1]_vs[f2]_out_[disease/pathway]_predictions_enriched[timestamp].txt`

The rows in each file correspond to the ranked enriched list of targets/pathways/diseases that are more statistically associated with the first SMILES/SDF file (`--f1`) of (e.g. cytotoxic) compounds. A higher Odd's Ratio (column `Odds_Ratio`) or Risk Ratio (`Risk_Ratio`) indicates a larger degree of enrichment for a given target/pathway/disease compared to the second input `--f2` (nontoxic) compound set.

The output has columns for the number of compound predictions (column `[f1/f2]_[In]Actives_[probability_activity]`) and the associated percentage `Percent_[f1/f2]_[In]Actives_[probability_activity]`) of compounds with that prediction.

The Fishers or Chi-squared p-values are provided (`[Fishers_Test/Chisquared]_P_Value`) including the Benjamini & Hochberg corrected values in the column named `[Fishers_Test/Chisquared]_P_Value_Corrected`. The output should be filtered for a given preference.

The percentage NaN predictions (compounds outside the Applicability Domain (AD) filter that were not given an active/inactive target prediction) are also provided in the column entitled `[f1/f2]_Proportion_Nan_Predictions_[ad]`.

Note: Please note that the Odd's and Risk ratios are implemented in a different way to the previous version of PIDGIN. For this version, larger numbers indicate larger enrichments.

In this example, there are six targets with a corrected p-value less than 0.05 with a Odds or Risk ratio greater than 1.0. All targets have known links to cytotoxicity, for example three are related to Tublin with known mechanisms to cytotoxicity (via cytoskeletal machinery).

More complicated example

Target/pathway/disease enrichment analysis can be combined with all model filters outlined in the previous section "Getting started".

For example, the following code:

```
$ python $PV3/predict_enriched.py --f1 cytotox_library.smi --f2 nontoxic_background.
↪smi --organism Drosophila -b 100 --known_flag --ad 0 -n 4 -p 0.8 --min_size 50 --se_
↪filter --performance_filter l50po,bedroc,0.8
```

would filter for Drosophila models that did not require Sphere Exlusion (SE) (i.e. sufficient number of inactives available) and a minimum number of 50 actives in the training set, with a minimum BEDROC performance of 0.8

for leave out 50% of ChEMBL publications from training data over 4-fold cross validation (L50PO), to produce enrichment predictions at a 0.8 probability cut-off at a threshold of 100μM, with the Applicability Domain (AD) filter silenced and where known activities (in ChEMBL or PubChem) are set.

References

1.4 Developer Notes

Contributions to PIDGINv3 are welcome.

The following documentation is designed to aid developers contribute code and new functionaility.

1.4.1 Warnings and Errors

`MolFromSmilesError` is raised due to "None" from Chem.MolFromSmiles when importing user mols

`PreprocessViolation` is raised due to preprocess violation when applied to input molecules

`SdfNoneMolError` raised due to "None" mol during enumeration through Chem.SDMolSupplier

> **Note:** Rdkit does not generate warning when enumerating through MolSupplier so this check is performed. Future work aims to enable parallel mol generation from SDFiles (see to do)

1.4.2 Contributing Code

Please submit any issues to the issue tracker to enable other developers to contribute to the project and reduce work load.

1.4.3 Documentation Usage

Coming soon...

- Parallel SDF import
- Enrichment anaylsis for two files
- Detailed similarity to training set analysis
- 3D E3FP fingerprints