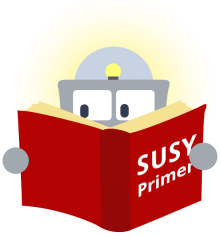

SUSY-AI Documentation

Version 1.1.4

July 15, 2016

About

SUSY-AI [1] is a Python code that allows for a quick classification of models within (submodels of) the phenomenological Minimal Supersymmetric Standard Model (pMSSM) by means of machine learning. The classifier was trained on the original ATLAS data and exclusion analysis [2]. All scripts and files discussed here can be downloaded from the project website (`https://susyai.hepforge.org/`), with an exception of third-party applications and scripts, which can be downloaded from their respective websites.
If you are using SUSY-AI in your scientific work, please cite the SUSY-AI paper:

> Sascha Caron, Jong Soo Kim, Krzysztof Rolbiecki, Roberto Ruiz de Austri, Bob Stienen, "The BSM-AI project: SUSY-AI - Generalizing LHC limits on Supersymmetry with Machine Learning". arXiv: 1605.02797 [hep-ph].

1 Dependencies

To run SUSY-AI inside the Python environment, the following packages have to be installed:

- `cPickle` or `pickle` (usually a native Python module);
- `numpy` (`http://www.numpy.org/`);
- `sklearn` [3] (`http://www.scikit-learn.org/`).

The `cPickle` module is preferred, since it is faster than its `pickle` counterpart. In addition, if `pyslha` [4] is installed or located in the same folder as SUSY-AI, SUSY-AI can make predictions directly from spectrum files. The `pyslha` module can be downloaded from its project website (`https://pypi.python.org/pypi/pyslha`). To work in the Python 3.x environment, the `pyslha` interface version 3.2 or newer is required. SUSY-AI was tested in `Python 2.7.6`, `Python 3.4.3` and `Python 3.5.1`. Due to differences between the `cPickle` module between these distributions, SUSY-AI works faster in `Python 3.x`. Using the most recent Python release is therefore recommended.

2 Usage

2.1 Preliminaries

SUSY-AI consists of two files:

- a classifier `susyai_classifier_python_vX.pkl`, where `X` stands for either `2` or `3`, corresponding to the Python version to be used;
- a Python interface `susyai.py`.

The classifier is saved in the pickle file, one for each major Python release. Although the Python 2.x version of the pickle is compatible with Python 3.x, the Python 3.x version of the pickle is lighter and loads in quicker into SUSY-AI. Unfortunately, the 3.x version is not backwards compatible.

2.2 Setting up SUSY-AI

To use SUSY-AI, one has to place `susyai.py` in a folder that is accessible for importing, e.g. in the folder where the Python script calling it is located:

```
import numpy as np
from susyai import susyai
```

Note that importing of `numpy` *has* to be done first, to avoid problems when loading pickle files. It can also be done by providing an explicit system path to `numpy`:

```
import sys
sys.path.insert(0, '[absolute_path_to_folder]')
import numpy as np
from susyai import susyai
```

The `susyai.py` interface is a class file, so once the classifier interface has been imported, one has to create a respective object and configure it, before using it. Here, it is possible to give an explicit location of the pickle file:

```
sa = susyai(pickle_location = "[pickle_location]")
```

Alternatively, the location can be given at a later point in the script, though before the prediction takes place.

M_1	M_2	M_3	$m_{\tilde{L}_1}$	$m_{\tilde{L}_3}$	$m_{\tilde{E}_1}$	$m_{\tilde{E}_3}$	$m_{\tilde{Q}_1}$	$m_{\tilde{Q}_3}$	$m_{\tilde{U}_1}$
$m_{\tilde{U}_3}$	$m_{\tilde{D}_1}$	$m_{\tilde{D}_3}$	A_t	A_b	A_τ	μ	m_A^2	$\tan\beta$	

Table 1: Variables used for prediction with masses and trilinear couplings in GeV; see Table 2 and 4 of [1].

```
sa = susyai()
sa.set_pickle_location("[pickle_location]")
```

Setting a pickle file does not mean it is immediately load into memory; this has to be done using the `load_pickle()` method. If a pickle is selected but not loaded when prediction starts, the pickle will be loaded automatically. This increases the overall runtime of the prediction method, so loading the pickle manually is recommended.

```
sa = susyai()
sa.set_pickle_location("[pickle_location]")
sa.load_pickle()
```

2.3 Predicting

After setting the `susyai` object, one can proceed to classifying specific points in the pMSSM parameter space in two ways: either by directly giving the input parameters into `susyai`, or by providing location(s) of spectrum file(s) that should be analyzed.

2.3.1 Prediction by coordinates

SUSY-AI was trained on 19 input parameters of the pMSSM; see Table 1 and [1]. A user can give these coordinates directly in the code and `susyai` can make predictions for these data point(s) using the `predict()` method. Following the order of the variables in Table 1, the prediction for specific coordinates can be made as follows:

```
from susyai import susyai
import numpy as np

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)

data = np.array([[811.65, -526.38, 2339.95, 3421.54, 2861.32,
          3496.63, 3339.50, 1994.02, 2230.6599, 2691.41,
          1842.7199, 3941.68, 1799.2599, -3041.83, -3501.52,
          -1533.57, 351.21, 3423.61, 12.25]])
clas, pred, cert, coords = sa.predict(data)
```

Predictions can be made for multiple coordinates in the same call, simply by adding new coordinates to the argument:

```
from susyai import susyai
import numpy as np

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)

data = np.array([[811.65, -526.38, 2339.95, 3421.54, 2861.32,
          3496.63, 3339.50, 1994.02, 2230.6599, 2691.41,
```

```
            1842.7199, 3941.68, 1799.2599, -3041.83, -3501.52,
            -1533.57, 351.21, 3423.61, 12.25],
        [-3800.69, 2881.01, 1688.58, 1020.47, 2283.25, 1888.7,
            1047.99, 3108.86, 2056.61, 3898.88, 1712.05, 2943.75,
            1763.34, 2629.84, -2241.57, -3969.9, -813.08, 2199.54,
            10.56]]])
clas, pred, cert,coords = sa.predict(data)
```

The `predict` method returns three `numpy` arrays: classification (`clas`), prediction (`pred`) and confidence (`conf`). Their structure is described in more detail in subsection 2.4.

2.3.2 Prediction by spectrum files

Instead of inputing the variables by hand in the code, one can also let SUSY-AI make predictions directly from `SLHA` spectrum files. To do this, one has to define how to obtain the parameters listed in Table 1 from the file. This is using the `coordinate_selector` method in `susyai`. `susyai` has a default preset for the coordinate selector, documented in appendix B. This preset is creatively called "1" and can be used via

```
from susyai import susyai
import numpy as np

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords = sa.predict_files(files)
```

Note the extra output variable `coords`, which contains the coordinates in (N,19)-dimensional `numpy` array, as they were extracted from the input files defined in the `files` list. When calling the default preset for coordinate selection, the coordinate selector is implicitly set to a list containing the coordinate specifications. Using `get_coordinate_selector()` this preset can be retrieved from the `susyai` object and altered if needed.

2.3.2.1 Selection of coordinates: It may happen that the provided preset does not work for all different structures SLHA-files can have. `susyai`, therefore, accepts customized selections of coordinates to be defined internally. Two formats for doing this are supported: either by inputting a list of [BLOCK, SWITCH]- elements, or by entering a function which does the selection directly from a `pyslha` objects. Reading in the data directly from the spectrum file using a list is recommended:

```
from susyai import susyai
import numpy as np

coordinatelist = [['MSOFT',1],['MSOFT',2],['MSOFT',3],['MSOFT',31],
    ['MSOFT',33],['MSOFT',34],['MSOFT',36],['MSOFT',41],['MSOFT',43],
    ['MSOFT',44],['MSOFT',46],['MSOFT',47],['MSOFT',49],['AU',(3,3)],
    ['AD',(3,3)],['AE',(3,3)],['HMIX',1],['HMIX',4],['HMIX',2]]

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(coordinatelist)

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords = sa.predict_files(files)
```

This approach yields the same result as using the following function:

```
from susyai import susyai
import numpy as np

def csel(spectrum):
    coordinates = np.zeros(19)
    coordinates[0] = spectrum.blocks['MSOFT'][1]          # M_1
    coordinates[1] = spectrum.blocks['MSOFT'][2]          # M_2
    coordinates[2] = spectrum.blocks['MSOFT'][3]          # M_3
    coordinates[3] = spectrum.blocks['MSOFT'][31]         # mL1
    coordinates[4] = spectrum.blocks['MSOFT'][33]         # mL3
    coordinates[5] = spectrum.blocks['MSOFT'][34]         # me1
    coordinates[6] = spectrum.blocks['MSOFT'][36]         # me3
    coordinates[7] = spectrum.blocks['MSOFT'][41]         # mQ1
    coordinates[8] = spectrum.blocks['MSOFT'][43]         # mQt
    coordinates[9] = spectrum.blocks['MSOFT'][44]         # mu1
    coordinates[10] = spectrum.blocks['MSOFT'][46]        # mtR
    coordinates[11] = spectrum.blocks['MSOFT'][47]        # md1
    coordinates[12] = spectrum.blocks['MSOFT'][49]        # mbR
    coordinates[13] = spectrum.blocks['AU'][(3,3)]        # At
    coordinates[14] = spectrum.blocks['AD'][(3,3)]        # Ab
    coordinates[15] = spectrum.blocks['AE'][(3,3)]        # Atau
    coordinates[16] = spectrum.blocks['HMIX'][1]          # mu
    coordinates[17] = spectrum.blocks['HMIX'][4]          # mA2
    coordinates[18] = spectrum.blocks['HMIX'][2]          # tanbeta
    return coordinates

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(csel)

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords = sa.predict_files(files)
```

The `csel` function has to follow a couple of rules. Firstly it has to return a `numpy` array of shape (19,) and it must take only a single argument `spectrum`. This function will be called internally with a `pyslha Doc` objects as input argument. Examples on how to use these objects can be found in the example scripts available on the `susyai` website. Noteworthy is that the use of a function has an advantage that the user can define mathematical operations to be applied to input from the spectrum file, e.g. rotations. `susyai` has a built-in method `dict_to_matrix()` to create $N \times N$-ndarrays from `pyslha` dictionary objects. An example of this can be found in the example scripts or in paragraph 2.3.2.3. For further information about the `pyslha Doc` objects, the reader is referred to the `pyslha` documentation at `https://pypi.python.org/pypi/pyslha` and the SUSY-AI example scripts.

2.3.2.2 Selection of IDs: In some cases, it may be useful to assign an ID to each file, so that it is known which coordinate belongs to which file. In `susyai`, there is a standard switch available called `'filename'`, that takes a name of the file (stripped of its extension) as an ID. Alternatively, using another switch `'fullfilename'` the entire filename (including the extension) is used as the ID.

```
from susyai import susyai
import numpy as np

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)
sa.set_id_selector('filename')

files = ['spectrum1.slha', 'spectrum2.slha']
clas, pred, cert, coords, ids = sa.predict_files(files)
```

If this does not suit the user, they can also create a function `isel`, similar to the coordinate selection function `csel` from paragraph 2.3.2.1, that generates an ID based on the spectrum (using the first argument of the function) and the file path (the second argument). This function has to return the ID as a string. The following function is therefore identical to using the `'filename'` switch in the previous script:

```
from susyai import susyai
import numpy as np

def isel(spectrum, filepath):
    f = filepath.split('/')[-1]
    if '.' in f:
        f = '.'.join(f.split('.')[:-1])
    return f

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)
sa.set_id_selector(isel)

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords, ids = sa.predict_files(files)
```

Note the extra output variable `ids` for the `predict_files()` method. This is a `numpy` array containing all IDs of the files used, in the same order as output variables. Therefore `ids[i]` corresponds to `clas[i]`, `pred[i]` etc.

2.3.2.3 Selection of extra information: It is also possible to extract an additional information from the input file that will not be used in prediction. This is done by adding another function, analogous to the `csel` and `isel` functions from the previous paragraphs, called an `xtra_selector` (`xsel`). It takes a spectrum (a `pyslha` `Doc` object) and a path as input arguments and returns a `numpy` `ndarray`. The number of elements in this array is decided by an user, as long as this number is fixed in a single prediction run. For example:

```
from susyai import susyai
import numpy as np

def xsel(spectrum, filepath):
    x = np.zeros(3)
    x[0] = spectrum.blocks['MASS'][1000002]
    x[1] = spectrum.blocks['MASS'][1000021]
    x[2] = spectrum.blocks['MASS'][1000022]
    return x

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)
sa.set_xtra_selector(xsel)
sa.set_id_selector('filename')

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords, xtras, ids = sa.predict_files(files)
```

Again, note an additional output variable `xtras` and where its position among the output variables. `susyai` has a built-in method `dict_to_matrix()` to create $N \times N$-ndarrays from `pyslha` dictionary objects.

```
from susyai import susyai
import numpy as np
```

```
def xsel(spectrum, filepath):
    usqmixinv = np.linalg.inv(
        susyai.dict_to_matrix(spectrum.blocks['USQMIX'])
    )
    uq = np.array([spectrum.blocks['MASS'][1000002],
         spectrum.blocks['MASS'][1000004],
         spectrum.blocks['MASS'][1000006],
         spectrum.blocks['MASS'][2000002],
         spectrum.blocks['MASS'][2000004],
         spectrum.blocks['MASS'][2000006]])

    coordinates = np.zeros(2)
    coordinates[0] = abs(np.dot(usqmixinv, uq)[0])
    coordinates[1] = abs(np.dot(usqmixinv, uq)[3])
    return coordinates

sa = susyai("susyai_classifier_python_v3.pkl")
sa.set_coordinate_selector(1)
sa.set_xtra_selector(xsel)

files = ['spectrum_01.slha', 'spectrum_02.slha']
clas, pred, cert, coords = sa.predict_files(files)
```

2.4 Prediction output

Both `predict()` and `predict_files()` yield the following output variables (in this specific order). All output variables are `numpy ndarray`s:

- **classification**: indicates whether the parameter point is allowed (1.0) or not (0.0);
- **prediction**: indicates whether the parameter point is allowed or not, based on a continuous scale from 0.0 (not allowed) to 1.0 (allowed);
- **confidence**: the probability that the classification is correct; see [1] for more details.
- **coordinates**: the coordinates used for the evaluation; these can differ from the input coordinates when the `alter` switch is set to `True`.

Additionally, when using the `predict_files()` method, two extra output variables may be returned, depending on the configuration of `susyai`:

- **extra information**: the additional information retrieved using the function defined by `set_xtra_selector()`. The variable is only returned if the user has set the selector function.
- **ids**: file IDs retrieved using the method the user defined via `set_id_selector()`. The variable is only returned if the user has set the selection method (either a custom function or `'filename'`).

2.4.1 Verbosity

SUSY-AI will show its progress using log messages, each preceded by the UNIX time stamp. This default behavior can be changed by the `set_verbosity()` method, which takes a single argument, indicating the behavior you want SUSY-AI to follow:

Value	Description
`0`	Both messages and warnings will be suppressed
`1`	Messages are suppressed, warnings will be shown
`2` *(default)*	All messages and warnings are shown

Note that this setting will not affect the warning displayed by the program when `pyslha` is not found when loading `susyai`.

2.5 Assumptions about the LSP nature

SUSY-AI was trained exclusively on spectra in which the lightest neutralino was the lightest supersymmetric particle (LSP). In order to make the usage of `susyai` possible for spectra with different LSPs, it will proceed with the prediction, but will give a warning and save a list of the IDs of all 'problematic' files, as long as an ID selector has been set. This list can be retrieved after prediction from `susyai`, so that the user can decide whether to trust the prediction or not.

```
from os import popen
import numpy as np
from susyai import susyai

folders = ['nsusy','msugra']
files = []
for i in range(len(folders)):
    fs = popen('ls %s | grep "%s"' % (folders[i], '.slha')).readlines()
    for j in range(len(fs)):
        files.append(folders[i]+'/'+fs[j].rstrip())

sa = susyai()
sa.set_id_selector('filename')
for i in range(len(files)):
    sa.get_data_from_file(files[i])
print sa.get_id_lists('LSP')
```

2.6 The `min_confidence` parameter

Both `predict()` and `predict_files()` can take an extra input parameter called `min_confidence`. If this parameter is set, all results with a confidence below this value are removed from the returned variables.

2.7 The `map_outsiders` configuration variable

The `susyai` object has an additional configuration variable `map_outsiders`, which is by default set to `False`. When this parameter is set to `True` by

```
sa.set_map_outsiders(True)
```

all input variables are checked whether they are within the sampling region that was used for training of the classifier accompanying `susyai`. If a data point lies outside the range of $\texttt{minimum} + 0.1125 * \texttt{full\_range}$ and $\texttt{maximum} - 0.1125 * \texttt{full\_range}$ for a specific coordinate, the point is mapped to this boundary, in order to avoid boundary effects of the learning algorithm.

3 Interfacing SUSY-AI

Because the interface of `sklearn` as well as `cPickle` are native to Python, all computational work has to be performed using Python. If a user wants to call SUSY-AI in a different environment like C or C++ for example, an interface has to be used (see `https://docs.python.org/2/extending/embedding.html`). The user should consult the documentation of the specific environment for more details.

References

[1] S. Caron, J. S. Kim, K. Rolbiecki, R. Ruiz de Astri, and B. Stienen, "SUSY Phenomenology at the LHC Using Machine Learning," *JHEP*, p. 21, 2016.

[2] G. Aad *et al.*, "Summary of the ATLAS experiment's sensitivity to supersymmetry after LHC Run 1 — interpreted in the phenomenological MSSM," *JHEP*, vol. 10, p. 134, 2015.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] A. Buckley, "PySLHA: a Pythonic interface to SUSY Les Houches Accord data," *Eur. Phys. J.*, vol. C75, no. 10, p. 467, 2015.

A Method reference

This is a method reference for SUSY-AI. Please refer to the manual itself, or the accompanying paper [1] for full details of used concepts.

A.1 Constructor

`__init__(pickle_location = None, messages = 2, coordinate_selector = 1, xtra_selector = None, id_selector = None, map_outsiders = False)`

The constructor takes six optional arguments that are subsequently all set by the corresponding setters. The constructor is called on construction of the object:

```
sa = susyai( ...  )
```

where the dots can be replaced by any variable from the `__init__` method just mentioned. These parameters are not mandatory. They can also be set later in the script by calling corresponding setter-methods.
For all parameters set by this constructor the same limitations and rules apply as if they were set by their corresponding setter-methods. Look at the documentation for these setters for information on these limitations and rules.

A.2 Getters

`get_classifier()`

After using `load_pickle()` (or `set_classifier()`), returns the loaded classifier.

`get_coordinate_selector()`

Returns the coordinate selector function if manually set, `None` otherwise. When preset "1" is chosen as a coordinate selector through `set_coordinate_selector()`, the output will be a list containing this preset selection. See appendix B for more details.

`get_id_lists(name = None)`

During running `susyai`, some file IDs may be deemed interesting for further inspection by the user. These IDs are saved in an ID list with label `name`. Calling `get_id_lists()` and providing that label as parameter, returns that list. If no name was provided as parameter, all lists are returned in the form of a dictionary. `None` will be returned if a non-existent label was provided.
Currently the only ID list that is generated by `susyai` is `LSP`, which contains all file ids corresponding to spectrum files with an LSP other than the lightest neutralino.

`get_id_selector()`

Returns the ID selector function if manually set, otherwise `None`.

`get_max(i)`

Returns a maximum value for feature `i` as used in the training of the classifier supplied with SUSY-AI. `i` has to be in range 0 to 19, otherwise the IndexError is raised.

`get_min(i)`

Returns a minimum value for feature `i` as used in the training of the classifier supplied with SUSY-AI. `i` has to be in range 0 to 19, otherwise the IndexError is raised.

`get_name(i)`

Returns a name of feature `i`. `i` has to be in range 0 to 19, otherwise the IndexError is raised.

`get_pickle_location()`

Returns a location of the pickle (string) if set, otherwise returns `None`

`get_map_outsiders()`

Returns a boolean, indicating if the coordinates are going to be mapped to the inside of the original sampling region used for training of the classifier.

`get_verbosity()`

Returns a flag set for showing messages. See `set_verbosity()` for more information on the flags.

`get_xtra_selector()`

Returns an extra selector function if manually set. Otherwise returns `None`.

A.3 Setters

`set_classifier(classifier)`

Sets the classifier used in predicting to `classifier`. This has to be a sklearn classifier that implements the `predict_proba()` method.
Although SUSY-AI allows for classifiers other than the original classifier to be used, these custom classifiers won't be able to make use of the mapping-functionality without manually altering the sampling region lists in the class.

`set_coordinate_selector(csel = 1)`

Sets the method by which the coordinates are extracted from the spectrumfile.

- **Function**: argument `csel` can be set to a function, that accepts only a single argument `spectrum`, containing a `pyslha Doc` object. This function must return the coordinate in the form of a `numpy` array of shape (19,), containing the parameters following table 1;
- **Integer** *(default = 1)*: using the preset function B for coordinate extraction files by setting `csel` to `1`. Currently, only this preset function is implemented;
- **List**: When the coordinates are directly readable from the file and no mathematical alterations to the read data have to be made, a list of shape (19,2) can be supplied as `csel`, where every first column indicating the block and every second one the switch of the variable to be selected.

This method will raise a ValueError when a selector is tried to be set that does not correspond to one of these types.

`set_id_selector(isel)`

Sets the method by which the id of the file can be extracted to `isel`.

- **Function**: argument `isel` can be set to a function, that accepts the arguments `spectrum` (containing a `pyslha Doc` object) and `filepath` (containing the path to the file that is being predicted). This function must return the id in the form of a string;
- **'filename'**: will take the filepath (stripped from its extension) as id;
- **'fullfilename'**: will take the filepath (*including* its extension) as id;
- **`None`** *(default)*: No id will be taken from the file.

This method will raise a ValueError when a selector is tried to be set that does not correspond to one of these types.

`set_verbosity(verbosity = 2)`

Sets whether or not messages are printed to the screen.
This method will raise a ValueError if the supplied value for `verbosity` does not correspond to one of these options. The warning given on importing the script when the `pyslha` package is not found is not affected by this setting: it will always be shown if `pyslha` is not found.

value	description
`0`	Both messages and warnings will be surpressed
`1`	Messages are surpressed, warnings will still be shown
`2` *(default)*	All messages and warnings are shown (default)

`set_pickle_location(pickle_location)`

Sets the filepath of the pickle that has to be loaded into SUSY-AI to `pickle_location` (string).
This method will raise a TypeError if the supplied value for `pickle_location` was not a string and an IOError if the pickle could not be found at this location.

`set_map_outsiders(map_outsiders = False)`

Sets whether or not coordinates outside of the sampling range used in training the accompanying classifier have to be mapped to the inside of this range. `map_outsiders` has to be a boolean.

`set_xtra_selector(xsel)`

Sets the method by which extra information (xtras called in this class) are extracted to `xsel`.

- **Function**: argument `xsel` can be set to a function, that accepts the arguments `spectrum` (containing a `pyslha Doc` object) and `filepath` (containing the path to the file that is being predicted). This function must return the extra data in the form of a `numpy ndarray` of shape (1,N), with N the number of extra variables. N may not change during a single run;
- **`None`**: No extra information will be taken from the file.

This method will raise a ValueError when a selector is tried to be set that does not correspond to one of these types.

A.4 Messages

`message(text)`

Will print `text` to the screen, preceded by the UNIX timestamp in seconds if `get_messages()` yields a value greater than or equal to 2.

`warn(text)`

Will print a warning containing `text` to the screen, preceded by the UNIX timestamp in seconds if `get_messages()` yields greater than or equal to 1.

`warn_once(name, text)`

Identical to `warn(text)`, but will only print a warning with name `name` once every prediction run.

A.5 Pickle

`load_pickle()`

Loads the pickle as set via the constructor or `set_pickle_location()`. This has only to be done once in a single run.
It is recommended to call this method before calling either `predict()` or `predict_files()`.
This method will raise and Exception if no pickle was set to be loaded.

A.6 Selecting data

`check_lsp(spectrum)`

Checks if the lightest neutralino, indicated by PDG code 1000022, is the lightest supersymmetric particle (LSP) in `spectrum` (a `pyslha` Doc object). If the `MASS` block or the mass of the lightest neutralino was not found, a warning will be printed. Analogously a warning will be printed if the lightest neutralino is not the LSP. Every warning printed by `check_lsp()` is only given once every prediction run.
Additionally to the printing of an error message, SUSY-AI will return an integer value, providing a programmatic interface to the result of this method. Possible returned values are:

value	description
`1`	The lightest neutralino was identified as the LSP.
`0`	No mass block was given, so LSP could not be determined.
`-1`	The lightest neutralino is not the LSP. All file-ids are saved and can be read out via `get_id_lists('LSP')`.
`-2`	The lightest neutralino is not the LSP. File-ids could not be saved for later investigation due to the absence of an id-selector.

`dict_to_matrix(dictionary)`

Takes a dictionary of length N*N with tuples (1,1) to (N,N) and returns the corresponding `numpy ndarray`.

`get_data_from_file(filepath)`

Uses the coordinate selector, xtra selector and id selector to get data from the spectrum file defined in `filepath`. It will return three variables:

- **coordinates**: A `numpy ndarray` of shape (1,19), containing the coordinates of the file specified by `filepath`.
- **xtras**: A textttnumpy `ndarray` of shape(1,x), with x the number of xtras as returned by the set xtra selector in `set_xtra_selector()`. *If no xtra selector was set, an empty array will be returned.*
- **ids**: A list of length 1, containing the id of the file as generated by the id selector set through `set_id_selector()`. *If no id selector was set, an empty list will be returned.*

`get_spectrum(filepath)`

Returns the a `Doc` object (native to `pyslha`) containing the spectrum of the file at `filepath`.
This method will raise an IOERror if the file in `filepath` could not be found.

`select_coorindates(self, spectrum)`

Returns the coordinates out of a spectrumfile to be used in prediction in the form of a `numpy ndarray` of shape (19,). Which values are returned, depends on the value set for `coordinate_selector` in the constructor or in a call of the `set_coordinate_selector()` method.
This method will raise a ValueError if the set coordinate_selector does not correspond to one of the allowed types (see `set_coordinate_selector()`.

`select_id(spectrum, filepath)`

Returns the id corresponding to spectrum `spectrum` and filepath `filepath` in the form of a string. How this value is determined, depends on the value set for `id_selector` in the constructor or in a call of the `set_id_selector()` method. If no id_selector has been set, it will return an empty `numpy ndarray`.
This method will raise a ValueError if the set id_selector does not correspond to one of the allowed types (seestions for you in `set_id_selector()`.

`select_xtras(spectrum, filepath)`

Returns extra information out spectrum `spectrum` (corresponding to the file in filepath `filepath`) in the form of a `numpy ndarray` of shape (N,). Which values are returned, depends on the value set for `xtra_selector` in the constructor or in a call of the `set_xtra_selector()` method. If no xtra_selector has been set, it will return an empty `numpy ndarray`.
This method will raise a ValueError if the set xtra_selector does not correspond to one of the allowed types (see `set_xtra_selector()`.

A.7 Prediction

`apply_min_confidence(arrayorlist, confidence, min_confidence)`

Returns all elements `i` of `arrayorlist` where `confidence[i]` is greater than or equal to `min_confidence`. The output is of the same type as `arrayorlist`, which has to be either an `numpy ndarray` or a `list`.

`calculate_results(prediction)`

Returns values for the predicted classification and the confidence of this classification for a prediction between 0.0 and 1.0 in `prediction`.

`predict(coordinates, min_confidence = 0.5)`

Returns a the following information on prediction of the provided N coordinates:

- **classification**: A `numpy ndarray` of length N containing the values 1.0 (meaning 'allowed') and 0.0 (meaning 'excluded');
- **prediction**: A `numpy ndarray` of length N containing values between 0.0 and 1.0, indicating the probability the coordinate is allowed.
- **confidence**: A `numpy ndarray` of length N containing values between 0.5 and 1.0, indicating the confidence the classification given in `classification` for that given datapoint is correct.
- **coordinates**: A `numpy ndarray` of shape (N,19), containing the coordinates as used in prediction. If the coordinates were mapped, the mapped coordinates will be returned.

The coordinates have to be supplied in a `numpy ndarray` of shape (N,19), with N the number of coordinates. The returned values are all `numpy` arrays of length N.
Setting the `min_confidence` argument in this method will remove all results with a confidence less than this value. `min_confidence` must be set to a value between 0.5 and 1.0, otherwise a ValueError will be raised.

`predict_files(filepaths, min_confidence = 0.5)`

Returns the following information on prediction of the provided N files in *list* (!) `filepaths`:

- **classification**: A `numpy ndarray` of length N containing the values 1.0 (meaning 'allowed') and 0.0 (meaning 'excluded');
- **prediction**: A `numpy ndarray` of length N containing values between 0.0 and 1.0, indicating the probability the coordinate is allowed.
- **confidence**: A `numpy ndarray` of length N containing values between 0.5 and 1.0, indicating the confidence the classification given in `classification` for that given datapoint is correct.
- **coordinates**: A `numpy ndarray` of shape (N,19), containing the coordinates as used in prediction. If the coordinates were mapped, the mapped coordinates will be returned.
- **xtras**: A `numpy ndarray` of shape(N,x), with x the number of xtras as returned by the set xtra selector in `set_xtra_selector()`. *If no xtra selector was set, this array will not be returned.*
- **ids**: A `numpy ndarray` of shape (N,), containing the ids of the coordinates as generated by the id selector set through `set_id_selector()`. *If no id selector was set, this array will not be returned.*

Setting the `min_confidence` argument in this method will remove all results with a confidence less than this value. `min_confidence` must be set to a value between 0.5 and 1.0, otherwise a ValueError will be raised.

`map_coordinates(coordinates)`

Maps the coordinates in `coordinates` to inside the sampled region used in training the accompanying pickled classifier. The coordinates have to be a `numpy ndarray` of shape (N,19), with N the number of coordinates. Returns the mapped coordinates in this same shape.

B Coordinate selection preset-1

SUSY-AI has a pre-defined function for selecting coordinates from SLHA-files. For reference, the values that are taken to construct the coordinates are mentioned here:

Variable	Block	Switch
M_1	MSOFT	1
M_2	MSOFT	2
M_3	MSOFT	3
mL1	MSOFT	31
mL3	MSOFT	33
me1	MSOFT	34
me3	MSOFT	36
mQ1	MSOFT	41
mQt	MSOFT	43
mu1	MSOFT	44
mtR	MSOFT	46
md1	MSOFT	47
mbR	MSOFT	49
At	AU	3,3
Ab	AD	3,3
Atau	AE	3,3
mu	HMIX	1
mA^2	HMIX	4
tanbeta	HMIX	2

If, by any means, this list is not correct for your files, you can create your own function for selecting the values from the spectrum files (see `set_coordinate_selector()` in the method reference). If the values you want to select are directly readable from the file (so without mathematical computations), you can also input a list as input for `set_coordinate_selector()`. The list corresponding to the selection above is

```
[['MSOFT',1],['MSOFT',2],['MSOFT',3],['MSOFT',31],['MSOFT',33],
    ['MSOFT',34],['MSOFT',36],['MSOFT',41],['MSOFT',43],['MSOFT',44],
    ['MSOFT',46],['MSOFT',47],['MSOFT',49],['AU',(3,3)],['AD',(3,3)],
    ['AE',(3,3)],['HMIX',1],['HMIX',4],['HMIX',2]]
```

and it outputted when setting this preset trough `set_coordinate_selector(1)` and subsequently calling `get_coordinate_selector()`. By altering this list and setting this altered one as the new coordinate selector, one can use this preset as template for his/her own selector.

C Summary of example scripts

Included in the download of SUSY-AI is a folder 'examples', containing example scripts on how to work with SUSY-AI. In this appendix, a short explaination is given on what can be found in each file.

No.	Filename	Description
01	Prediction of single coordinate	Uses an 1d-array to generate a prediction.
02	Prediction of multiple coordinates	Uses an array to predict multiple coordinates at the same time. This is significantly quicker than calling `predict()` multiple times!
03	Prediction from file	Reads a spectrum from a .slha file and generates a prediction for it, using preset-1 coordinate selection.
04	Prediction from multiple files	Reads in multiple .slha files and generates predictions for it, using preset-1 coordinate selection.
05	Coordinate selection via function	Prediction of a single file with selection of coordinates via user defined function. This selection is - in this specific case - equal to the one of preset-1.
06	Coorindate selection via list	Prediction for a single file with selection of coordinates via definitions in a list. This selection is - in this specific case - equal to the one of the preset-1.
07	Extra information selector	Selects extra information from a spectrum file on prediction.
08	Extra information selection and applying rotation matrix	Using an xtra selector function to get extra information from a spectrum file and applying a rotation matrix to the extracted information. Uses the `dict_to_matrix()` method to do the rotation.
09	ID selection via filename	Selects the id of a file as the filename of the file, stripped from its extension.
10	ID selection via function	Selects the id of a file via an user-defined function.
11	Mapping coordinates	Example on how to map coordinates before prediction and on how to get the mapped coordinates into your script.
12	Using minimal confidence	Predicting using a minimal confidence to remove uncertain predictions from the results.
13	Applying minimal confidence afterwards	Applies multiple different minimal confidence levels to result data, without performing the prediction multiple times.
14	Surpressing messages	Example on how to surpress printes messages from `susyai`. No prediction is done in this script.
15	Get data from file	Shows how to extract data from a file, without doing any predictions.
16	Full example	An example combining nearly all methods one can use in `susyai`.