
Acton Documentation

Release 0.3.3

Matthew Alger & Cheng Soon Ong

Oct 27, 2017

1	acton	3
1.1	acton package	3
2	Developer Documentation	29
2.1	Contributing	29
2.2	Adding a New Predictor	29
2.3	Adding a New Recommender	30
3	Indices and tables	31
	Python Module Index	33

Contents:

acton package

Subpackages

acton.proto package

Submodules

acton.proto.acton_pb2 module

acton.proto.io module

Functions for reading/writing to protobufs.

acton.proto.io.**GeneratedProtocolMessageType** (*name*, **args*, ***kwargs*)

acton.proto.io.**get_ndarray** (*data*: list, *shape*: tuple, *dtype*: str) → <MagicMock
id='139807259956616'>

Converts a list of values into an array.

Parameters

- **data** – Raw array data.
- **shape** – Shape of the resulting array.
- **dtype** – Data type of the resulting array.

Returns Array with the given data, shape, and dtype.

Return type numpy.ndarray

acton.proto.io.**read_metadata** (*file*: typing.Union[str, typing.BinaryIO]) → bytes

Reads metadata from a protobufs file.

Parameters **file** – Path to binary file, or file itself.

Returns Metadata.

Return type bytes

acton.proto.io.**read_proto**()

Reads a protobuf from a .proto file.

Parameters

- **path** – Path to the .proto file.
- **Proto** – Protocol message class (from the generated protobuf module).

Returns The parsed protobuf.

Return type *GeneratedProtocolMessageType*

`acton.proto.io.read_protos()`
Reads many protobufs from a file.

Parameters

- **file** – Path to binary file, or file itself.
- **Proto** – Protocol message class (from the generated protobuf module).

Yields *GeneratedProtocolMessageType* – A parsed protobuf.

`acton.proto.io.write_proto()`
Serialises a protobuf to a file.

Parameters

- **path** – Path to binary file. Will be overwritten.
- **proto** – Protobuf to write to file.

`acton.proto.io.write_protos(path: str, metadata: bytes = b'')`
Serialises many protobufs to a file.

Parameters

- **path** – Path to binary file. Will be overwritten.
- **metadata** – Optional bytestring to prepend to the file.

Notes

Coroutine. Accepts protobufs, or None to terminate and close file.

acton.proto.wrappers module

Classes that wrap protobufs.

class `acton.proto.wrappers.LabelPool` (*proto: typing.Union[str, mock.mock.LabelPool]*)
Bases: `object`

Wrapper for the LabelPool protobuf.

proto

acton_pb.LabelPool – Protobuf representing the label pool.

db_kwargs

dict – Key-value pairs of keyword arguments for the database constructor.

label_encoder

sklearn.preprocessing.LabelEncoder – Encodes labels as integers. May be None.

DB

Gets a database context manager for the specified database.

Returns Database context manager.

Return type `type`

classmethod `deserialize` (*proto: bytes, json: bool = False*) → `acton.proto.wrappers.LabelPool`
 Deserialises a protobuf into a `LabelPool`.

Parameters

- `proto` – Serialised protobuf.
- `json` – Whether the serialised protobuf is in JSON format.

Returns

Return type `LabelPool`

ids

Gets a list of IDs.

Returns List of known IDs.

Return type `List[int]`

labels

Gets labels array specified in input.

Notes

The returned array is cached by this object so future calls will not need to recompile the array.

Returns `T x N x F NumPy` array of labels.

Return type `numpy.ndarray`

classmethod `make` (*ids: typing.Iterable[int], db: acton.database.Database*) → `acton.proto.wrappers.LabelPool`
 Constructs a `LabelPool`.

Parameters

- `ids` – Iterable of instance IDs.
- `db` – Database

Returns

Return type `LabelPool`

class `acton.proto.wrappers.Predictions` (*proto: typing.Union[str, mock.mock.Predictions]*)
 Bases: `object`

Wrapper for the Predictions protobuf.

proto

`acton_pb.Predictions` – Protobuf representing predictions.

db_kwargs

`dict` – Dictionary of database keyword arguments.

label_encoder

`sklearn.preprocessing.LabelEncoder` – Encodes labels as integers. May be `None`.

DB

Gets a database context manager for the specified database.

Returns Database context manager.

Return type `type`

classmethod `deserialise` (*proto: bytes, json: bool = False*) → `acton.proto.wrappers.Predictions`
Deserialises a protobuf into Predictions.

Parameters

- `proto` – Serialised protobuf.
- `json` – Whether the serialised protobuf is in JSON format.

Returns

Return type `Predictions`

labelled_ids

Gets a list of IDs the predictor knew the label for.

Returns List of IDs the predictor knew the label for.

Return type `List[int]`

classmethod `make` (*predicted_ids: typing.Iterable[int], labelled_ids: typing.Iterable[int], predictions: <MagicMock id='139807260252816'>, db: acton.database.Database, predictor: str = ''*) → `acton.proto.wrappers.Predictions`
Converts NumPy predictions to a Predictions object.

Parameters

- `predicted_ids` – Iterable of instance IDs corresponding to predictions.
- `labelled_ids` – Iterable of instance IDs used to train the predictor.
- `predictions` – T x N x D array of corresponding predictions.
- `predictor` – Name of predictor used to generate predictions.
- `db` – Database.

Returns

Return type `Predictions`

predicted_ids

Gets a list of IDs corresponding to predictions.

Returns List of IDs corresponding to predictions.

Return type `List[int]`

predictions

Gets predictions array specified in input.

Notes

The returned array is cached by this object so future calls will not need to recompile the array.

Returns T x N x D NumPy array of predictions.

Return type `numpy.ndarray`

class `acton.proto.wrappers.Recommendations` (*proto:* `typing.Union[str, mock.mock.Recommendations]`)

Bases: `object`

Wrapper for the Recommendations protobuf.

proto

`acton_pb.Recommendations` – Protobuf representing recommendations.

db_kwargs

`dict` – Key-value pairs of keyword arguments for the database constructor.

label_encoder

`sklearn.preprocessing.LabelEncoder` – Encodes labels as integers. May be `None`.

DB

Gets a database context manager for the specified database.

Returns Database context manager.

Return type `type`

classmethod `deserialise` (*proto:* `bytes`, *json:* `bool = False`) → `acton.proto.wrappers.Recommendations`

Deserialises a protobuf into `Recommendations`.

Parameters

- **proto** – Serialised protobuf.
- **json** – Whether the serialised protobuf is in JSON format.

Returns

Return type `Recommendations`

labelled_ids

Gets a list of labelled IDs.

Returns List of labelled IDs.

Return type `List[int]`

classmethod `make` (*recommended_ids:* `typing.Iterable[int]`, *labelled_ids:* `typing.Iterable[int]`, *recommender:* `str`, *db:* `acton.database.Database`) → `acton.proto.wrappers.Recommendations`

Constructs a `Recommendations`.

Parameters

- **recommended_ids** – Iterable of recommended instance IDs.
- **labelled_ids** – Iterable of labelled instance IDs used to make recommendations.
- **recommender** – Name of the recommender used to make recommendations.
- **db** – Database.

Returns

Return type `Recommendations`

recommendations

Gets a list of recommended IDs.

Returns List of recommended IDs.

Return type `List[int]`

`acton.proto.wrappers.deserialize_encoder` (*encoder: mock.mock.LabelEncoder*) →
<MagicMock name='mock.LabelEncoder'
id='139807259529112'>

Deserialises a LabelEncoder protobuf.

Parameters `encoder` – LabelEncoder protobuf.

Returns LabelEncoder (or None if no encodings were specified).

Return type sklearn.preprocessing.LabelEncoder

`acton.proto.wrappers.validate_db` (*db: mock.mock.Database*)
Validates a Database proto.

Parameters `db` – Database to validate.

Raises ValueError

Module contents

Submodules

acton.acton module

Main processing script for Acton.

`acton.acton.draw` (*n: int, lst: typing.List[T], replace: bool = True*) → typing.List[T]
Draws n random elements from a list.

Parameters

- `n` – Number of elements to draw.
- `lst` – List of elements to draw from.
- `replace` – Draw with replacement.

Returns n random elements.

Return type List[T]

`acton.acton.get_DB` (*data_path: str, pandas_key: str = None*) -> (<class 'acton.database.Database'>,
<class 'dict'>)
Gets a Database that will handle the given data table.

Parameters

- `data_path` – Path to file.
- `pandas_key` – Key for pandas HDF5. Specify iff using pandas.

Returns

- `Database` – Database that will handle the given data table.
- `dict` – Keyword arguments for the Database constructor.

`acton.acton.label` (*recommendations: acton.proto.wrappers.Recommendations*) → ac-
ton.proto.wrappers.LabelPool
Simulates a labelling task.

Parameters

- `data_path` – Path to data file.

- **feature_cols** – List of column names of features. If empty, all columns will be used.
- **label_col** – Column name of the labels.
- **pandas_key** – Key for pandas HDF5. Specify iff using pandas.

Returns

Return type *acton.proto.wrappers.LabelPool*

```
acton.acton.main(data_path: str, feature_cols: typing.List[str], label_col: str, output_path: str,
                 n_epochs: int = 10, initial_count: int = 10, recommender: str = 'Random-
                 Recommender', predictor: str = 'LogisticRegression', pandas_key: str = '',
                 n_recommendations: int = 1)
```

Simulate an active learning experiment.

Parameters

- **data_path** – Path to data file.
- **feature_cols** – List of column names of the features. If empty, all non-label and non-ID columns will be used.
- **label_col** – Column name of the labels.
- **output_path** – Path to output file. Will be overwritten.
- **n_epochs** – Number of epochs to run.
- **initial_count** – Number of random instances to label initially.
- **recommender** – Name of recommender to make recommendations.
- **predictor** – Name of predictor to make predictions.
- **pandas_key** – Key for pandas HDF5. Specify iff using pandas.
- **n_recommendations** – Number of recommendations to make at once.

```
acton.acton.predict(labels: acton.proto.wrappers.LabelPool, predictor: str) → ac-
ton.proto.wrappers.Predictions
```

Train a predictor and predict labels.

Parameters

- **labels** – IDs of labelled instances.
- **predictor** – Name of predictor to make predictions.

```
acton.acton.recommend(predictions: acton.proto.wrappers.Predictions, recommender: str
                      = 'RandomRecommender', n_recommendations: int = 1) → ac-
ton.proto.wrappers.Recommendations
```

Recommends instances to label based on predictions.

Parameters

- **recommender** – Name of recommender to make recommendations.
- **n_recommendations** – Number of recommendations to make at once. Default 1.

Returns

Return type *acton.proto.wrappers.Recommendations*

`acton.acton.simulate_active_learning` (*ids: typing.Iterable[int], db: acton.database.Database, db_kwargs: dict, output_path: str, n_initial_labels: int = 10, n_epochs: int = 10, test_size: int = 0.2, recommender: str = 'RandomRecommender', predictor: str = 'LogisticRegression', n_recommendations: int = 1*)

Simulates an active learning task.

Parameters

- **ids** – IDs of instances in the unlabelled pool.
- **db** – Database with features and labels.
- **db_kwargs** – Keyword arguments for the database constructor.
- **output_path** – Path to output intermediate predictions to. Will be overwritten.
- **n_initial_labels** – Number of initial labels to draw.
- **n_epochs** – Number of epochs.
- **test_size** – Percentage size of testing set.
- **recommender** – Name of recommender to make recommendations.
- **predictor** – Name of predictor to make predictions.
- **n_recommendations** – Number of recommendations to make at once.

`acton.acton.try_pandas` (*data_path: str*) → bool
Guesses if a file is a pandas file.

Parameters **data_path** – Path to file.

Returns True if the file is pandas.

Return type bool

`acton.acton.validate_predictor` (*predictor: str*)
Raises an exception if the predictor is not valid.

Parameters **predictor** – Name of predictor.

Raises ValueError

`acton.acton.validate_recommender` (*recommender: str*)
Raises an exception if the recommender is not valid.

Parameters **recommender** – Name of recommender.

Raises ValueError

acton.cli module

Command-line interface for Acton.

`acton.cli.lines_from_stdin` () → typing.Iterable[str]
Yields lines from stdin.

`acton.cli.read_binary` () → bytes
Reads binary data from stdin.

Notes

The first eight bytes are expected to be the length of the input data as an unsigned long long.

Returns Binary data.

Return type bytes

`acton.cli.read_bytes_from_buffer` (*n: int, buffer: typing.BinaryIO*) → bytes
Reads n bytes from stdin, blocking until all bytes are received.

Parameters

- **n** – How many bytes to read.
- **buffer** – Which buffer to read from.

Returns Exactly n bytes.

Return type bytes

`acton.cli.write_binary` (*string: bytes*)
Writes binary data to stdout.

Notes

The output will be preceded by the length as an unsigned long long.

acton.database module

Wrapper class for databases.

class `acton.database.ASCIIReader` (*path: str, feature_cols: typing.List[str], label_col: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139807260785408'> = None*)

Bases: `acton.database.Database`

Reads ASCII databases.

feature_cols

List[str] – List of feature columns.

label_col

str – Name of label column.

max_id_length

int – Maximum length of IDs.

n_features

int – Number of features.

n_instances

int – Number of instances.

n_labels

int – Number of labels per instance.

path

str – Path to ASCII file.

encode_labels

bool – Whether to encode labels as integers.

label_encoder

sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_db

Database – Underlying ManagedHDF5Database.

_db_filepath

str – Path of underlying HDF5 database.

_tempdir

str – Temporary directory where the underlying HDF5 database is stored.

get_known_instance_ids () → typing.List[int]

Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids () → typing.List[int]

Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features (*ids: typing.Sequence[int]*) → <MagicMock id='139807260303992'>

Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139807260317288'>

Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto () → mock.mock.Database

Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int], features: <MagicMock id='139807260330192'>*)

write_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139807260347584'>*)

class acton.database.Database

Bases: abc.ABC

Base class for database wrappers.

get_known_instance_ids () → typing.List[int]

Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids () → typing.List[int]

Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features (*ids: typing.Sequence[int]*) → <MagicMock id='139807261056472'>

Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels (*labeller_ids: typing.Sequence[int]*, *instance_ids: typing.Sequence[int]*) → <MagicMock id='139807260993072'>

Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto () → mock.mock.Database

Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int]*, *features: <MagicMock id='139807261015176'>*)

Writes feature vectors to the database.

Parameters

- **ids** – Iterable of IDs.
- **features** – N x D array of feature vectors. The ith row corresponds to the ith ID in *ids*.

write_labels (*labeller_ids: typing.Sequence[int]*, *instance_ids: typing.Sequence[int]*, *labels: <MagicMock id='139807260947624'>*)

Writes label vectors to the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.
- **labels** – T x N x D array of label vectors. The ith row corresponds to the ith labeller ID in *labeller_ids* and the jth column corresponds to the jth instance ID in *instance_ids*.

class `acton.database.FITSReader` (*path: str*, *feature_cols: typing.List[str]*, *label_col: str*, *hdu_index: int = 1*, *encode_labels: bool = True*, *label_encoder: <MagicMock name='mock.LabelEncoder' id='139807260490160'> = None*)

Bases: `acton.database.Database`

Reads FITS databases.

hdu_index

int – Index of HDU in the FITS file.

feature_cols

List[str] – List of feature columns.

label_col

str – Name of label column.

n_features

int – Number of features.

n_instances

int – Number of instances.

n_labels

int – Number of labels per instance.

path

str – Path to FITS file.

encode_labels

bool – Whether to encode labels as integers.

label_encoder

sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_hdulist

astropy.io.fits.HDUList – FITS HDUList.

get_known_instance_ids () → *typing.List[int]*

Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type *List[str]*

get_known_labeller_ids () → *typing.List[int]*

Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type *List[str]*

read_features (*ids: typing.Sequence[int]*) → <MagicMock id='139807260498968'>

Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type *numpy.ndarray*

read_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <Magic-

Mock id='139807260532808'>

Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x 1 array of label vectors.

Return type *numpy.p*

to_proto () → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int], features: <MagicMock id='139807260589416'>*)

write_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139807260561480'>*)

class acton.database.HDF5Database (*path: str*)

Bases: *acton.database.Database*

Database wrapping an HDF5 file as a context manager.

path

str – Path to HDF5 file.

_h5_file

h5py.File – HDF5 file object.

class acton.database.HDF5Reader (*path: str, feature_cols: typing.List[str], label_col: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139807260694008'> = None*)

Bases: *acton.database.HDF5Database*

Reads HDF5 databases.

feature_cols

List[str] – List of feature datasets.

label_col

str – Name of label dataset.

n_features

int – Number of features.

n_instances

int – Number of instances.

n_labels

int – Number of labels per instance.

path

str – Path to HDF5 file.

encode_labels

bool – Whether to encode labels as integers.

label_encoder

sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_h5_file

h5py.File – HDF5 file object.

_is_multidimensional

bool – Whether the features are in a multidimensional dataset.

get_known_instance_ids () → typing.List[int]

Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids () → typing.List[int]

Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features (*ids: typing.Sequence[int]*) → <MagicMock id='139807260702872'>

Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels (*labeller_ids: typing.Sequence[int]*, *instance_ids: typing.Sequence[int]*) → <MagicMock id='139807260716168'>

Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto () → mock.mock.Database

Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int]*, *features: <MagicMock id='139807260729072'>*)

write_labels (*labeller_ids: typing.Sequence[int]*, *instance_ids: typing.Sequence[int]*, *labels: <MagicMock id='139807260754720'>*)

class `acton.database.ManagedHDF5Database` (*path: str*, *label_dtype: str = None*, *feature_dtype: str = None*)

Bases: `acton.database.HDF5Database`

Database using an HDF5 file.

Notes

This database uses an internal schema. For reading files from disk, use another Database.

path

str – Path to HDF5 file.

label_dtype

str – Data type of labels.

feature_dtype

str – Data type of features.

_h5_file

h5py.File – Opened HDF5 file.

_sync_attrs

List[str] – List of instance attributes to sync with the HDF5 file's attributes.

get_known_instance_ids () → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids () → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features (*ids: typing.Sequence[int]*) → <MagicMock id='139807260583192'>
Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139807260638456'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto () → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int], features: <MagicMock id='139807260907560'>*)
Writes feature vectors to the database.

Parameters

- **ids** – Iterable of IDs.
- **features** – N x D array of feature vectors. The ith row corresponds to the ith ID in *ids*.

Returns N x D array of feature vectors.

Return type numpy.ndarray

write_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139807260576008'>*)
Writes label vectors to the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.
- **labels** – T x N x D array of label vectors. The ith row corresponds to the ith labeller ID in *labeller_ids* and the jth column corresponds to the jth instance ID in *instance_ids*.

class `acton.database.PandasReader` (*path: str, feature_cols: typing.List[str], label_col: str, key: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139807260394720'> = None*)

Bases: `acton.database.Database`

Reads HDF5 databases.

feature_cols

List[str] – List of feature datasets.

label_col

str – Name of label dataset.

n_features

int – Number of features.

n_instances

int – Number of instances.

n_labels

int – Number of labels per instance.

path

str – Path to HDF5 file.

encode_labels

bool – Whether to encode labels as integers.

label_encoder

sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_df

pandas.DataFrame – Pandas dataframe.

get_known_instance_ids () → *typing.List[int]*

Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type *List[str]*

get_known_labeller_ids () → *typing.List[int]*

Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type *List[str]*

read_features (*ids: typing.Sequence[int]*) → *<MagicMock id='139807260403528'>*

Reads feature vectors from the database.

Parameters *ids* – Iterable of IDs.

Returns N x D array of feature vectors.

Return type *numpy.ndarray*

read_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → *<MagicMock id='139807260412728'>*

Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.

- **instance_ids** – Iterable of instance IDs.

Returns T x N x 1 array of label vectors.

Return type numpy.ndarray

to_proto () → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features (*ids: typing.Sequence[int], features: <MagicMock id='139807260429728'>*)

write_labels (*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139807260459472'>*)

acton.database.product (*seq: typing.Iterable[int]*)
Finds the product of a list of ints.

Parameters **seq** – List of ints.

Returns Product.

Return type int

acton.database.serialise_encoder (*encoder: <MagicMock name='mock.LabelEncoder' id='139807261044408'>*) → mock.mock.LabelEncoder
Serialises a LabelEncoder as a protobuf.

Parameters **encoder** – LabelEncoder.

Returns Protobuf representing the LabelEncoder.

Return type LabelEncoderPB

acton.kde_predictor module

A predictor that uses KDE to classify instances.

class **acton.kde_predictor.KDEClassifier** (*bandwidth=1.0*)
Bases: BaseEstimator, ClassifierMixin

A classifier using kernel density estimation to classify instances.

fit (*X, y*)
Fits kernel density models to the data.

Parameters

- **X** (*array_like, shape (n_samples, n_features)*) – List of n_features-dimensional data points. Each row corresponds to a single data point.
- **y** (*array-like, shape (n_samples,)*) – Target vector relative to X.

predict (*X*)
Predicts class labels.

Parameters **X** (*array_like, shape (n_samples, n_features)*) – List of n_features-dimensional data points. Each row corresponds to a single data point.

predict_proba (*X*)
Predicts class probabilities.

Class probabilities are normalised log densities of the kernel density estimates.

Parameters **X** (*array_like, shape (n_samples, n_features)*) – List of *n_features*-dimensional data points. Each row corresponds to a single data point.

acton.labellers module

Labeller classes.

class `acton.labellers.ASCIITableLabeller` (*path: str, id_col: str, label_col: str*)
Bases: `acton.labellers.Labeller`

Labeller that obtains labels from an ASCII table.

path
str – Path to table.

id_col
str – Name of the column where IDs are stored.

label_col
str – Name of the column where binary labels are stored.

_table
astropy.table.Table – Table object.

query (*id_: int*) → <MagicMock id='139807260094536'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns 1 x 1 label array.

Return type `numpy.ndarray`

class `acton.labellers.DatabaseLabeller` (*db: acton.database.Database*)
Bases: `acton.labellers.Labeller`

Labeller that obtains labels from a Database.

_db
acton.database.Database – Database with labels.

query (*id_: int*) → <MagicMock id='139807260103344'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns 1 x 1 label array.

Return type `numpy.ndarray`

class `acton.labellers.Labeller`
Bases: `abc.ABC`

Base class for labellers.

query (*id_: int*) → <MagicMock id='139807260078432'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns T x F label array.

Return type `numpy.ndarray`

acton.plot module

Script to plot a dump of predictions.

`acton.plot.plot` (*predictions: typing.Iterable[typing.BinaryIO]*)
Plots predictions from a file.

Parameters `predictions` – Files containing predictions.

acton.predictors module

Predictor classes.

`acton.predictors.AveragePredictions` (*predictor: acton.predictors.Predictor*) → `acton.predictors.Predictor`
Wrapper for a predictor that averages predicted probabilities.

Notes

This effectively reduces the number of predictors to 1.

Parameters `predictor` – Predictor to wrap.

Returns Predictor with averaged predictions.

Return type *Predictor*

class `acton.predictors.Committee` (*Predictor: type, db: acton.database.Database, n_classifiers: int = 10, subset_size: float = 0.6, **kwargs: dict*)
Bases: `acton.predictors.Predictor`

A predictor using a committee of other predictors.

n_classifiers

int – Number of logistic regression classifiers in the committee.

subset_size

float – Percentage of known labels to take subsets of to train the classifier. Lower numbers increase variety.

_db

`acton.database.Database` – Database storing features and labels.

_committee

`List[sklearn.linear_model.LogisticRegression]` – Underlying committee of logistic regression classifiers.

_reference_predictor

Predictor – Reference predictor trained on all known labels.

fit (*ids: typing.Iterable[int]*)

Fits the predictor to labelled data.

Parameters `ids` – List of IDs of instances to train from.

predict (*ids: typing.Sequence[int]*) → (`<MagicMock id='139807260155016'>`, `<MagicMock id='139807260167416'>`)
Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters `ids` – List of IDs of instances to predict labels for.

Returns

- `numpy.ndarray` – An $N \times T \times C$ array of corresponding predictions.
- `numpy.ndarray` – A N array of confidences (or None if not applicable).

reference_predict (`ids: typing.Sequence[int]`) -> (`<MagicMock id='139807260076744'>`, `<MagicMock id='139807260080672'>`)

Predicts labels using the best possible method.

Parameters `ids` – List of IDs of instances to predict labels for.

Returns

- `numpy.ndarray` – An $N \times 1 \times C$ array of corresponding predictions.
- `numpy.ndarray` – A N array of confidences (or None if not applicable).

class `acton.predictors.GPClassifier` (`db: acton.database.Database`, `max_iters: int = 50000`,
`n_jobs: int = 1`)

Bases: `acton.predictors.Predictor`

Classifier using Gaussian processes.

`max_iters`

`int` – Maximum optimisation iterations.

`label_encoder`

`sklearn.preprocessing.LabelEncoder` – Encodes labels as integers.

`model_`

`gpy.models.GPClassification` – GP model.

`_db`

`acton.database.Database` – Database storing features and labels.

fit (`ids: typing.Iterable[int]`)

Fits the predictor to labelled data.

Parameters `ids` – List of IDs of instances to train from.

predict (`ids: typing.Sequence[int]`) -> (`<MagicMock id='139807260541168'>`, `<MagicMock id='139807260071080'>`)

Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters `ids` – List of IDs of instances to predict labels for.

Returns

- `numpy.ndarray` – An $N \times 1 \times C$ array of corresponding predictions.
- `numpy.ndarray` – A N array of confidences (or None if not applicable).

reference_predict (*ids: typing.Sequence[int]*) -> (<MagicMock id='139807259866896'>, <MagicMock id='139807259908136'>)
 Predicts labels using the best possible method.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An $N \times 1 \times C$ array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

class `acton.predictors.Predictor`

Bases: `abc.ABC`

Base class for predictors.

prediction_type

str – What kind of predictions this class generates, e.g. `classification`.

fit (*ids: typing.Iterable[int]*)

Fits the predictor to labelled data.

Parameters **ids** – List of IDs of instances to train from.

predict (*ids: typing.Sequence[int]*) -> (<MagicMock id='139807259769040'>, <MagicMock id='139807259778632'>)
 Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An $N \times T \times C$ array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

prediction_type = `'classification'`

reference_predict (*ids: typing.Sequence[int]*) -> (<MagicMock id='139807260274472'>, <MagicMock id='139807260202544'>)
 Predicts labels using the best possible method.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An $N \times 1 \times C$ array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

`acton.predictors.from_class` (*Predictor: type, regression: bool = False*) → *type*

Converts a scikit-learn predictor class into a Predictor class.

Parameters

- **Predictor** – scikit-learn predictor class.
- **regression** – Whether this predictor does regression (as opposed to classification).

Returns Predictor class wrapping the scikit-learn class.

Return type `type`

`acton.predictors.from_instance` (*predictor: BaseEstimator, db: acton.database.Database, regression: bool = False*) → `acton.predictors.Predictor`

Converts a scikit-learn predictor instance into a Predictor instance.

Parameters

- **predictor** – scikit-learn predictor.
- **db** – Database storing features and labels.
- **regression** – Whether this predictor does regression (as opposed to classification).

Returns Predictor instance wrapping the scikit-learn predictor.

Return type *Predictor*

acton.recommenders module

Recommender classes.

class `acton.recommenders.EntropyRecommender` (*db: acton.database.Database*)

Bases: `acton.recommenders.Recommender`

Recommends instances by confidence-based uncertainty sampling.

recommend (*ids: typing.Sequence[int], predictions: <MagicMock id='139807259709000'>, n: int = 1, diversity: float = 0.5*) → `typing.Sequence[int]`

Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – $N \times 1 \times C$ array of predictions. The *i*th row must correspond with the *i*th ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in $[0, 1]$.

Returns IDs of the instances to label.

Return type `Sequence[int]`

class `acton.recommenders.MarginRecommender` (*db: acton.database.Database*)

Bases: `acton.recommenders.Recommender`

Recommends instances by margin-based uncertainty sampling.

recommend (*ids: typing.Sequence[int], predictions: <MagicMock id='139807259735040'>, n: int = 1, diversity: float = 0.5*) → `typing.Sequence[int]`

Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.

- **predictions** – $N \times 1 \times C$ array of predictions. The i th row must correspond with the i th ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in $[0, 1]$.

Returns IDs of the instances to label.

Return type Sequence[int]

class `acton.recommenders.QBCRecommender` (*db: acton.database.Database*)

Bases: `acton.recommenders.Recommender`

Recommends instances by committee disagreement.

recommend (*ids: typing.Sequence[int]*, *predictions: <MagicMock id='139807259669336'>*, *n: int = 1*, *diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – $N \times T \times C$ array of predictions. The i th row must correspond with the i th ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in $[0, 1]$.

Returns IDs of the instances to label.

Return type Sequence[int]

class `acton.recommenders.RandomRecommender` (*db: acton.database.Database*)

Bases: `acton.recommenders.Recommender`

Recommends instances at random.

recommend (*ids: typing.Sequence[int]*, *predictions: <MagicMock id='139807259651488'>*, *n: int = 1*, *diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – $N \times T \times C$ array of predictions.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in $[0, 1]$.

Returns IDs of the instances to label.

Return type Sequence[int]

class `acton.recommenders.Recommender`

Bases: `abc.ABC`

Base class for recommenders.

recommend (*ids*: typing.Sequence[int], *predictions*: <MagicMock id='139807259625512'>, *n*: int = 1, *diversity*: float = 0.5) → typing.Sequence[int]
Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x T x C array of predictions.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**UncertaintyRecommender** (*db*: acton.database.Database)

Bases: *acton.recommenders.Recommender*

Recommends instances by confidence-based uncertainty sampling.

recommend (*ids*: typing.Sequence[int], *predictions*: <MagicMock id='139807259687120'>, *n*: int = 1, *diversity*: float = 0.5) → typing.Sequence[int]
Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x 1 x C array of predictions. The *i*th row must correspond with the *i*th ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

acton.recommenders.**choose_boltzmann** (*features*: <MagicMock id='139807259615416'>, *scores*: <MagicMock id='139807259604808'>, *n*: int, *temperature*: float = 1.0) → typing.Sequence[int]

Chooses *n* scores using a Boltzmann distribution.

Notes

Scores are chosen from highest to lowest. If there are less scores to choose from than requested, all scores will be returned in order of preference.

Parameters

- **scores** – 1D array of scores.
- **n** – Number of scores to choose.

- **temperature** – Temperature parameter for sampling. Higher temperatures give more diversity.

Returns List of indices of scores chosen.

Return type Sequence[int]

`acton.recommenders.choose_mmr` (*features*: <MagicMock id='139807259792048'>, *scores*: <MagicMock id='139807259768928'>, *n*: int, *l*: float = 0.5) → typing.Sequence[int]

Chooses n scores using maximal marginal relevance.

Notes

Scores are chosen from highest to lowest. If there are less scores to choose from than requested, all scores will be returned in order of preference.

Parameters

- **scores** – 1D array of scores.
- **n** – Number of scores to choose.
- **l** – Lambda parameter for MMR. $l = 1$ gives a relevance-ranked list and $l = 0$ gives a maximal diversity ranking.

Returns List of indices of scores chosen.

Return type Sequence[int]

Module contents

Developer Documentation

Contributing

We accept pull requests on GitHub. Contributions must be PEP8 compliant and pass formatting and function tests in the test script `/test`.

Adding a New Predictor

A predictor is a class that implements `acton.predictors.Predictor`. Adding a new predictor amounts to implementing a subclass of `Predictor` and registering it in `acton.predictors.PREDICTORS`.

Predictors must implement:

- `__init__(db: acton.database.Database, *args, **kwargs)`, which stores a reference to the database (and does any other initialisation).
- `fit(ids: Iterable[int])`, which takes an iterable of IDs and fits a model to the associated features and labels,
- `predict(ids: Sequence[int]) -> numpy.ndarray`, which takes a sequence of IDs and predicts the associated labels.
- `reference_predict(ids: Sequence[int]) -> numpy.ndarray`, which behaves the same as `predict` but uses the best possible model.

Predictors should store data-based values such as the model in attributes ending in an underscore, e.g. `self.model_`.

Why Does Acton Use Predictor?

Acton makes use of `Predictor` classes, which are often just wrappers for scikit-learn classes. This raises the question: Why not just use scikit-learn classes?

This design decision was made because Acton must support predictors that do not fit the scikit-learn API, and so using scikit-learn predictors directly would mean that there is no unified API for predictors. An example of where Acton diverges from scikit-learn is that scikit-learn does not support multiple labellers.

Adding a New Recommender

A recommender is a class that implements `acton.recommenders.Recommender`. Adding a new recommender amounts to implementing a subclass of `Recommender` and registering it in `acton.recommenders.RECOMMENDERS`.

Recommenders must implement:

- `__init__(db: acton.database.Database, *args, **kwargs)`, which stores a reference to the database (and does any other initialisation).
- `recommend(ids: Iterable[int], predictions: numpy.ndarray, n: int=1, diversity: float=0.5) -> Sequence[int]`, which recommends `n` IDs from the given IDs based on the associated predictions.

Indices and tables

- `genindex`
- `modindex`
- `search`

a

acton, 27
acton.acton, 8
acton.cli, 10
acton.database, 11
acton.kde_predictor, 19
acton.labellers, 20
acton.plot, 21
acton.predictors, 21
acton.proto, 8
acton.proto.acton_pb2, 3
acton.proto.io, 3
acton.proto.wrappers, 4
acton.recommenders, 24

Symbols

_committee (acton.predictors.Committee attribute), 21
 _db (acton.database.ASCIIReader attribute), 12
 _db (acton.labellers.DatabaseLabeller attribute), 20
 _db (acton.predictors.Committee attribute), 21
 _db (acton.predictors.GPClassifier attribute), 22
 _db_filepath (acton.database.ASCIIReader attribute), 12
 _df (acton.database.PandasReader attribute), 18
 _h5_file (acton.database.HDF5Database attribute), 15
 _h5_file (acton.database.HDF5Reader attribute), 15
 _h5_file (acton.database.ManagedHDF5Database attribute), 16
 _hdulist (acton.database.FITSReader attribute), 14
 _is_multidimensional (acton.database.HDF5Reader attribute), 15
 _reference_predictor (acton.predictors.Committee attribute), 21
 _sync_attrs (acton.database.ManagedHDF5Database attribute), 16
 _table (acton.labellers.ASCIITableLabeller attribute), 20
 _tempdir (acton.database.ASCIIReader attribute), 12

A

acton (module), 27
 acton.acton (module), 8
 acton.cli (module), 10
 acton.database (module), 11
 acton.kde_predictor (module), 19
 acton.labellers (module), 20
 acton.plot (module), 21
 acton.predictors (module), 21
 acton.proto (module), 8
 acton.proto.acton_pb2 (module), 3
 acton.proto.io (module), 3
 acton.proto.wrappers (module), 4
 acton.recommenders (module), 24
 ASCIIReader (class in acton.database), 11
 ASCIITableLabeller (class in acton.labellers), 20
 AveragePredictions() (in module acton.predictors), 21

C

choose_boltzmann() (in module acton.recommenders), 26
 choose_mmr() (in module acton.recommenders), 27
 Committee (class in acton.predictors), 21

D

Database (class in acton.database), 12
 DatabaseLabeller (class in acton.labellers), 20
 DB (acton.proto.wrappers.LabelPool attribute), 4
 DB (acton.proto.wrappers.Predictions attribute), 5
 DB (acton.proto.wrappers.Recommendations attribute), 7
 db_kwargs (acton.proto.wrappers.LabelPool attribute), 4
 db_kwargs (acton.proto.wrappers.Predictions attribute), 5
 db_kwargs (acton.proto.wrappers.Recommendations attribute), 7
 deserialise() (acton.proto.wrappers.LabelPool class method), 5
 deserialise() (acton.proto.wrappers.Predictions class method), 6
 deserialise() (acton.proto.wrappers.Recommendations class method), 7
 deserialise_encoder() (in module acton.proto.wrappers), 7
 draw() (in module acton.acton), 8

E

encode_labels (acton.database.ASCIIReader attribute), 11
 encode_labels (acton.database.FITSReader attribute), 14
 encode_labels (acton.database.HDF5Reader attribute), 15
 encode_labels (acton.database.PandasReader attribute), 18
 EntropyRecommender (class in acton.recommenders), 24

F

feature_cols (acton.database.ASCIIReader attribute), 11
 feature_cols (acton.database.FITSReader attribute), 14
 feature_cols (acton.database.HDF5Reader attribute), 15
 feature_cols (acton.database.PandasReader attribute), 18
 feature_dtype (acton.database.ManagedHDF5Database attribute), 16

fit() (acton.kde_predictor.KDEClassifier method), 19
fit() (acton.predictors.Committee method), 21
fit() (acton.predictors.GPClassifier method), 22
fit() (acton.predictors.Predictor method), 23
FITSReader (class in acton.database), 13
from_class() (in module acton.predictors), 23
from_instance() (in module acton.predictors), 24

G

GeneratedProtocolMessageType() (in module acton.proto.io), 3
get_DB() (in module acton.acton), 8
get_known_instance_ids() (acton.database.ASCIIReader method), 12
get_known_instance_ids() (acton.database.Database method), 12
get_known_instance_ids() (acton.database.FITSReader method), 14
get_known_instance_ids() (acton.database.HDF5Reader method), 15
get_known_instance_ids() (acton.database.ManagedHDF5Database method), 16
get_known_instance_ids() (acton.database.PandasReader method), 18
get_known_labeller_ids() (acton.database.ASCIIReader method), 12
get_known_labeller_ids() (acton.database.Database method), 13
get_known_labeller_ids() (acton.database.FITSReader method), 14
get_known_labeller_ids() (acton.database.HDF5Reader method), 15
get_known_labeller_ids() (acton.database.ManagedHDF5Database method), 17
get_known_labeller_ids() (acton.database.PandasReader method), 18
get_ndarray() (in module acton.proto.io), 3
GPClassifier (class in acton.predictors), 22

H

HDF5Database (class in acton.database), 15
HDF5Reader (class in acton.database), 15
hdu_index (acton.database.FITSReader attribute), 13

I

id_col (acton.labellers.ASCIITableLabeller attribute), 20
ids (acton.proto.wrappers.LabelPool attribute), 5

K

KDEClassifier (class in acton.kde_predictor), 19

L

label() (in module acton.acton), 8
label_col (acton.database.ASCIIReader attribute), 11
label_col (acton.database.FITSReader attribute), 14
label_col (acton.database.HDF5Reader attribute), 15
label_col (acton.database.PandasReader attribute), 18
label_col (acton.labellers.ASCIITableLabeller attribute), 20
label_dtype (acton.database.ManagedHDF5Database attribute), 16
label_encoder (acton.database.ASCIIReader attribute), 12
label_encoder (acton.database.FITSReader attribute), 14
label_encoder (acton.database.HDF5Reader attribute), 15
label_encoder (acton.database.PandasReader attribute), 18
label_encoder (acton.predictors.GPClassifier attribute), 22
label_encoder (acton.proto.wrappers.LabelPool attribute), 4
label_encoder (acton.proto.wrappers.Predictions attribute), 5
label_encoder (acton.proto.wrappers.Recommendations attribute), 7
labelled_ids (acton.proto.wrappers.Predictions attribute), 6
labelled_ids (acton.proto.wrappers.Recommendations attribute), 7
Labeller (class in acton.labellers), 20
LabelPool (class in acton.proto.wrappers), 4
labels (acton.proto.wrappers.LabelPool attribute), 5
lines_from_stdin() (in module acton.cli), 10

M

main() (in module acton.acton), 9
make() (acton.proto.wrappers.LabelPool class method), 5
make() (acton.proto.wrappers.Predictions class method), 6
make() (acton.proto.wrappers.Recommendations class method), 7
ManagedHDF5Database (class in acton.database), 16
MarginRecommender (class in acton.recommenders), 24
max_id_length (acton.database.ASCIIReader attribute), 11
max_iters (acton.predictors.GPClassifier attribute), 22
model_ (acton.predictors.GPClassifier attribute), 22

N

n_classifiers (acton.predictors.Committee attribute), 21
n_features (acton.database.ASCIIReader attribute), 11
n_features (acton.database.FITSReader attribute), 14
n_features (acton.database.HDF5Reader attribute), 15
n_features (acton.database.PandasReader attribute), 18
n_instances (acton.database.ASCIIReader attribute), 11

n_instances (acton.database.FITSReader attribute), 14
 n_instances (acton.database.HDF5Reader attribute), 15
 n_instances (acton.database.PandasReader attribute), 18
 n_labels (acton.database.ASCIIReader attribute), 11
 n_labels (acton.database.FITSReader attribute), 14
 n_labels (acton.database.HDF5Reader attribute), 15
 n_labels (acton.database.PandasReader attribute), 18

P

PandasReader (class in acton.database), 17
 path (acton.database.ASCIIReader attribute), 11
 path (acton.database.FITSReader attribute), 14
 path (acton.database.HDF5Database attribute), 15
 path (acton.database.HDF5Reader attribute), 15
 path (acton.database.ManagedHDF5Database attribute), 16
 path (acton.database.PandasReader attribute), 18
 path (acton.labellers.ASCIITableLabeller attribute), 20
 plot() (in module acton.plot), 21
 predict() (acton.kde_predictor.KDEClassifier method), 19
 predict() (acton.predictors.Committee method), 21
 predict() (acton.predictors.GPCClassifier method), 22
 predict() (acton.predictors.Predictor method), 23
 predict() (in module acton.acton), 9
 predict_proba() (acton.kde_predictor.KDEClassifier method), 19
 predicted_ids (acton.proto.wrappers.Predictions attribute), 6
 prediction_type (acton.predictors.Predictor attribute), 23
 predictions (acton.proto.wrappers.Predictions attribute), 6
 Predictions (class in acton.proto.wrappers), 5
 Predictor (class in acton.predictors), 23
 product() (in module acton.database), 19
 proto (acton.proto.wrappers.LabelPool attribute), 4
 proto (acton.proto.wrappers.Predictions attribute), 5
 proto (acton.proto.wrappers.Recommendations attribute), 7

Q

QBCRecommender (class in acton.recommenders), 25
 query() (acton.labellers.ASCIITableLabeller method), 20
 query() (acton.labellers.DatabaseLabeller method), 20
 query() (acton.labellers.Labeller method), 20

R

RandomRecommender (class in acton.recommenders), 25
 read_binary() (in module acton.cli), 10
 read_bytes_from_buffer() (in module acton.cli), 11
 read_features() (acton.database.ASCIIReader method), 12
 read_features() (acton.database.Database method), 13
 read_features() (acton.database.FITSReader method), 14
 read_features() (acton.database.HDF5Reader method), 16

read_features() (acton.database.ManagedHDF5Database method), 17
 read_features() (acton.database.PandasReader method), 18
 read_labels() (acton.database.ASCIIReader method), 12
 read_labels() (acton.database.Database method), 13
 read_labels() (acton.database.FITSReader method), 14
 read_labels() (acton.database.HDF5Reader method), 16
 read_labels() (acton.database.ManagedHDF5Database method), 17
 read_labels() (acton.database.PandasReader method), 18
 read_metadata() (in module acton.proto.io), 3
 read_proto() (in module acton.proto.io), 3
 read_protos() (in module acton.proto.io), 4
 recommend() (acton.recommenders.EntropyRecommender method), 24
 recommend() (acton.recommenders.MarginRecommender method), 24
 recommend() (acton.recommenders.QBCRecommender method), 25
 recommend() (acton.recommenders.RandomRecommender method), 25
 recommend() (acton.recommenders.Recommender method), 25
 recommend() (acton.recommenders.UncertaintyRecommender method), 26
 recommend() (in module acton.acton), 9
 recommendations (acton.proto.wrappers.Recommendations attribute), 7
 Recommendations (class in acton.proto.wrappers), 6
 Recommender (class in acton.recommenders), 25
 reference_predict() (acton.predictors.Committee method), 22
 reference_predict() (acton.predictors.GPCClassifier method), 22
 reference_predict() (acton.predictors.Predictor method), 23

S

serialise_encoder() (in module acton.database), 19
 simulate_active_learning() (in module acton.acton), 9
 subset_size (acton.predictors.Committee attribute), 21

T

to_proto() (acton.database.ASCIIReader method), 12
 to_proto() (acton.database.Database method), 13
 to_proto() (acton.database.FITSReader method), 14
 to_proto() (acton.database.HDF5Reader method), 16
 to_proto() (acton.database.ManagedHDF5Database method), 17
 to_proto() (acton.database.PandasReader method), 19
 try_pandas() (in module acton.acton), 10

U

UncertaintyRecommender (class in acton.recommenders), 26

V

validate_db() (in module acton.proto.wrappers), 8

validate_predictor() (in module acton.acton), 10

validate_recommender() (in module acton.acton), 10

W

write_binary() (in module acton.cli), 11

write_features() (acton.database.ASCIIReader method), 12

write_features() (acton.database.Database method), 13

write_features() (acton.database.FITSReader method), 15

write_features() (acton.database.HDF5Reader method), 16

write_features() (acton.database.ManagedHDF5Database method), 17

write_features() (acton.database.PandasReader method), 19

write_labels() (acton.database.ASCIIReader method), 12

write_labels() (acton.database.Database method), 13

write_labels() (acton.database.FITSReader method), 15

write_labels() (acton.database.HDF5Reader method), 16

write_labels() (acton.database.ManagedHDF5Database method), 17

write_labels() (acton.database.PandasReader method), 19

write_proto() (in module acton.proto.io), 4

write_protos() (in module acton.proto.io), 4