# OutlierTree Documentation

**David Cortes**

**Jul 28, 2019**

# Contents:

This is the documentation page for the python package *OutlierTree*. For more details, see the project's GitHub page:
https://www.github.com/david-cortes/outliertree

# CHAPTER 1

# Installation

Package is available on PyPI, can be installed with

```
pip install outliertree
```

**class** outliertree.**OutlierTree**(*max_depth=4*, *min_gain=0.1*, *z_norm=2.67*, *z_outlier=8.0*, *pct_outliers=0.01*, *min_size_numeric=25*, *min_size_categ=75*, *categ_as_bin=True*, *ord_as_bin=True*, *cat_bruteforce_subset=False*, *follow_all=False*, *nthreads=-1*)

Bases: object

Outlier Tree

Explainable outlier detection through decision-tree grouping. Tries to detect outliers by generating decision trees that attempt to "predict" the values of each column based on each other column, testing in each branch of every tried split (if it meets some minimum criteria) whether there are observations that seem too distant from the others in a 1-D distribution for the column that the split tries to "predict" (unlike other methods, this will not generate a score for each observation).

Splits are based on gain, while outlierness is based on confidence intervals. Similar in spirit to the GritBot software developed by RuleQuest research.

Supports columns of types numeric, categorical, and ordinal (for this last one, will consider their order when splitting other columns from them, but not when splitting to "predict" them), and can handle missing values in any of them. Can also pass timestamps that will get converted to numeric but shown as timestamps in the output. Offers option to set columns to be used only to split other columns but not to look at outliers in them.

Infinite values will be taken into consideration when the column is used to split another column (that is, +inf will go into the branch that is greater than something, -inf into the other branch), but when a column is the target of the split, they will be taken as missing - that is, it will not report infinite values as outliers.

**Parameters**

- **max_depth** (*int*) – Maximum depth of the trees to grow. Can also pass zero, in which case it will only look for outliers with no conditions (i.e. takes each column as a 1-d distribution and looks for outliers in there independently of the values in other columns).

- **min_gain** (*float*) – Minimum gain that a split has to produce in order to consider it (both in terms of looking for outliers in each branch, and in considering whether to continue branching from them). Note that default value for GritBot is 1e-6.
- **z_norm** (*float*) – Maximum Z-value (from standard normal distribution) that can be considered as a normal observation. Note that simply having values above this will not automatically flag observations as outliers, nor does it assume that columns follow normal distributions. Also used for categorical and ordinal columns for building approximate confidence intervals of proportions.
- **z_outlier** (*float*) – Minimum Z-value that can be considered as an outlier. There must be a large gap in the Z-value of the next observation in sorted order to consider it as outlier, given by (z_outlier - z_norm). Ignored for categorical and ordinal columns.
- **pct_outliers** (*float*) – Approximate max percentage of outliers to expect in a given branch.
- **min_size_numeric** (*int*) – Minimum size that branches need to have when splitting a numeric column.
- **min_size_categ** (*int*) – Minimum size that branches need to have when splitting a categorical or ordinal column.
- **categ_as_bin** (*bool*) – Whether to make categorical-by-categorical binary splits by binarizing each category in the column and then attempting splits by grouping categories into subsets. Alternative is to create one branch per category of the column being split from. Ignored when there is only one or fewer categorical columns. Can only pass one of 'categ_as_bin' and 'cat_bruteforce_subset'.
- **ord_as_bin** (*bool*) – Same as 'categ_as_bin', but for ordinal columns, and cumulative (i.e. it splits by '<=', not '='). Ignored when there are no ordinal columns or no categorical columns.
- **cat_bruteforce_subset** (*bool*) – Whether to make categorical-by-categorical binary splits by trying all the possible combinations of columns in each subset (that is, it evaluates 2^n potential splits every time). Note that trying this when there are many categories in a column will result in exponential computation time that might never finish. Alternative is to create one branch per category of the column being split from. Ignored when there is only one or fewer categorical columns. Can only pass one of 'categ_as_bin' and 'cat_bruteforce_subset'.
- **follow_all** (*bool*) – Whether to continue branching from each split that meets the size and gain criteria. This will produce exponentially many more branches, and if depth is large, might take forever to finish. Will also produce a lot more spurious outiers. Not recommended.
- **nthreads** (*int*) – Number of parallel threads to use. When fitting the model, it will only use up to one thread per column, while for prediction it will use up to one thread per row. The more threads that are used, the more memory will be required and allocated, so using more threads will not always lead to better speed. Passing zero or negative numbers will default to the maximum number of available CPU cores (but not if the object attribute is overwritten). Can be changed after the object is already initialized.

**Variables**

- **`is_fitted`** (*`bool`*) – Indicates if the model as already been fit.
- **`flaggable_values`** (*`dict[ncols]`*) – A dictionary indicating for each column which kind of values are possible to flag as outlier in at least one of the explored branches. For numerical and timestamp columns, will indicate the lower and upper bounds of the normal range (that is, it can only flag values *outside* of that interval), while for categorical, ordinal,

and boolean, it will indicate the categories (which it can flag as outliers). If the lower bound is higher than the upper bound, it means that any value can potentially be flagged as outlier. If no values in a column can be flagged as outlier, entry for that column will be an empty dict.

- **cols_num** (*array(ncols_numeric, )*) – Names of the numeric columns in the data passed to '.fit'.
- **cols_cat** (*array(ncols_categ, )*) – Names of the categorical/string columns in the data passed to '.fit'.
- **cols_bool** (*array(ncols_bool, )*) – Names of the boolean columns in the data passed to '.fit'.
- **cols_ord** (*array(ncols_ordinal, )*) – Names of the ordinal columns in the data passed to '.fit'.
- **cols_ts** (*array(ncols_tiemstamp, )*) – Names of the timestamp columns in the data passed to '.fit'.

## References

[1] GritBot software : https://www.rulequest.com/gritbot-info.html

**fit** (*df*, *cols_ignore=None*, *outliers_print=10*, *return_outliers=True*)

Fit Outlier Tree model to data.

---

**Note:** Row names will be taken as the index of the data frame. Column types will be taken as follows:

Numeric -> Everything that is a subtype of numpy's 'number' (e.g. integers and floats).

Categorical -> Python object (**even if the underlying types are numbers!!!**), boolean, and pandas Categorical.

Ordinal -> pandas Categorical with ordered attribute - the order will be the same as they have when passed.

Timestamp -> numpy's datetime64 dtype - they will only be taken with a precision of seconds due to numerical precision issues (code uses C 'long double'), and will be used as numerical but presented as timestamp in the outputs. If the required precision is less (e.g. just dates or years) it's recommended to pass them as numbers instead (e.g. 2008.1095 for some day in February 2008).

---

**Note:** You can look at the object's attributes to ensure that the columns are being interpreted as the type (numeric, categorical, ordinal, boolean, timestamp) that they should have.

---

**Note:** Boolean or binary columns must be passed as categorical (will not accept them as numerical, nor as ordinal). If they have missing values, pandas will not be able to have them with dtype "bool" however (they need to be passed either as "object" or "Categorical" dtypes).

---

**Note:** Do NOT do one-hot or dummy encoding on categorical variables.

---

**Parameters**

- **df** (*DataFrame(n_rows, n_cols)*) – Pandas' DataFrame with normal data that might contain some outliers.
- **cols_ignore** (*boolean array(n_cols, ) or string array(n_ignore, )*) – Array containing columns which will not be split, but will be evaluated for usage in splitting other columns. Can pass either a boolean array with the same number of columns as 'df', or a list/array of column names (must match with those of 'df'). Pass 'None' to use all columns.
- **outliers_print** (*int or None*) – Maximum number of flagged outliers in the training data to print after fitting the model. Pass zero or None to avoid printing any. Outliers can be printed from resulting data frame afterwards through '.print_outliers'.
- **return_outliers** (*bool*) – Whether to return a DataFrame with information about outliers flagged in the training data. If 'True', will return this information as a DataFrame. If 'False', will return this same object. See the documentation for '.predict' for more information about the format.

**Returns** **result_df or self** – Either a DataFrame with the information about potential outliers detected in the training data, or a copy of this object if passing 'return_outliers' = 'False'. In the former, format is the same as when calling '.predict'. See the documentation for '.predict' for more information about the output format.

**Return type** DataFrame(n_rows, 6) or obj

**generate_gritbot_files**(*df*, *cols_ignore=None*, *save_folder='.'*, *file_name='data'*)

Generate data files for GritBot software

Generates CSV (.data) and naming (.names) files as required for use by the GritBot software (not included in this Python package). Note that this only generates the data files and doesn't do anything else.

**Parameters**

- **df** (*DataFrame(nrows, ncols)*) – Pandas DataFrame in the same format as requried by '.fit'.
- **cols_ignore** (*boolean array(n_cols, ) or string array(n_ignore, )*) – Array containing columns which will not be split, but will be evaluated for usage in splitting other columns. Pass 'None' to use all columns.
- **save_folder** (*str*) – Path to folder where to save the generated files.
- **file_name** (*str*) – Prefix for the file names (before the dot).

### References

[1] GritBot software : https://www.rulequest.com/gritbot-info.html

**predict**(*df*, *outliers_print=None*)

Detect outliers on new data

---

**Note:** The group statistics for outliers are calculated only on rows that are not flagged as outliers - that is, they do not include the row being reported in their calculations, and exclude any outliers that were flagged at a previous parent branch.

---

---

**Note:** This will generate conditions having these criteria: "<=", ">", "in", "=", "!=" (not equal). When it says "in", it means that the value is within the subset of categories provided. For ordinal columns, they will

always be represented as "in", but the subset will follow the order. The ".print" method will additionally simplify conditions to numeric "between", and merge repeated splits that are on the same column.

**Parameters**

- **df** (*DataFrame(n_rows, n_cols)*) – Pandas DataFrame in the same format and with the same columns as the one that was passed to '.fit'.
- **outliers_print** (*int or None*) – Maximum number of outliers to print, if any are found. Pass zero or None to avoid printing any. Outliers can be printed from resulting data frame afterwards through '.print_outliers'.

**Returns**

**result_df** – DataFrame indicating for each row whether it is a suspected outlier or not. When they are suspected outliers, the resulting DataFrame will contain columns with

a) information about the column in the input data whose value is suspicious,

b) aggregate statistics for the values in this column among the normal observations,

c) conditions that qualify the row to be put into that group,

d) depth in the decision tree branch in which they can be considered an outlier,

e) Whether the conditions that make it belong to the decision tree branch contain any NA split,

f) The resulting outlier score or probability (based on Chebyshyov's bound for numerical columns, and a simple upper conficence bound for categorical and ordinal columns - but note that it will always prefer to assign a row to an outlier branch that does not follow any NA path, or failing that, to the one with the smallest depth) - lower scores indicate lower probabilities and thus more outlierness.

Information such as the conditions in the tree or the group statistics are returned as dictionaries, since they are not tabular format. The index of the output will be the same as that of the input. Outliers can be printed from resulting data frame afterwards through '.print_outliers'.

**Return type** DataFrame(n_rows, 6)

**`print_outliers`**(*df_outliers*, *max_outliers=15*)

Print outliers in readable format

See the documentation for 'predict' for more details. This function will additionally perform some simplifications on the branch conditions, such as taking the smallest value when it is split two times by "<=". Can also pass a smaller DataFrame with only selected outliers to be printed.

**Parameters**

- **df_outliers** (*DataFrame(n_rows, 6)*) – DataFrame with outliers information as output by 'fit' or 'predict'.
- **max_outliers** (*int*) – Maximum number of outliers to print.

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Index

## F



## G



## O



## P