4 Automatic Outlier Detection Algorithms in Python

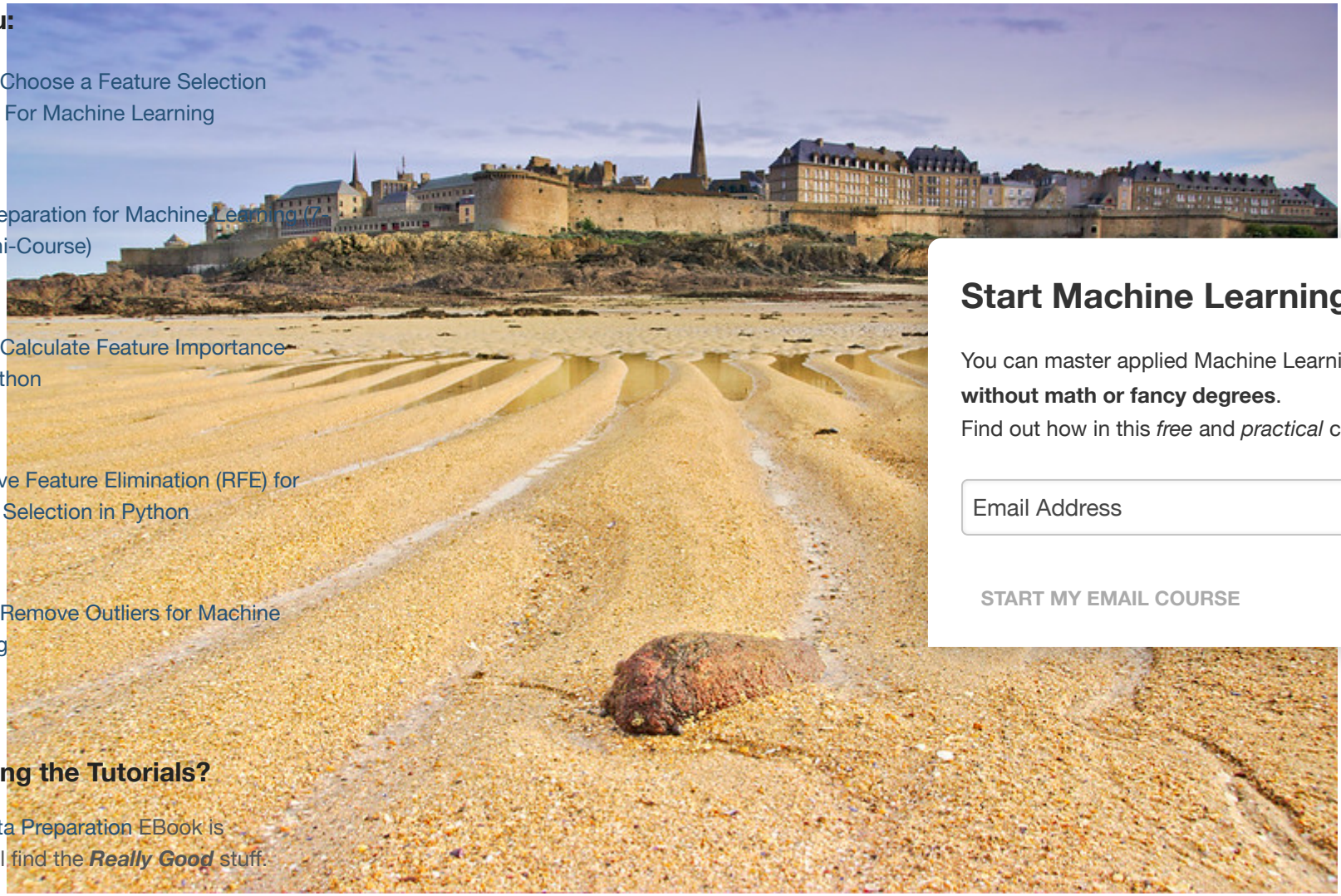
The presence of outliers in a classification or regression dataset can result in a poor fit and lower predictive modeling performance.

Identifying and **removing outliers** is challenging with simple statistical methods for most machine learning datasets given the large number of input variables. Instead, automatic outlier detection methods can be used in the modeling pipeline and compared, just like other data preparation transforms may be applied to the dataset.

In this tutorial, you will discover how to use automatic outlier detection and removal to improve machine learning predictive modeling performance.

After completing this tutorial, you will know:

- Automatic outlier detection models provide an alternative to statistical techniques with a larger number of input variables with complex and unknown inter-relationships.
- How to apply automatic outlier detection and removal to the training dataset only to avoid data leakage.
- How to evaluate and compare predictive modeling pipelines with outliers removed from the training dataset.



Model-Based Outlier Detection and Removal in Python
Photo by Zoltán Vörös, some rights reserved.

Tutorial Overview

This tutorial is divided into three parts; they are:

1. Outlier Detection and Removal
2. Dataset and Performance Baseline
    1. House Price Regression Dataset
    2. Baseline Model Performance
3. Automatic Outlier Detection
    1. Isolation Forest
    2. Minimum Covariance Determinant
    3. Local Outlier Factor
    4. One-Class SVM

Outlier Detection and Removal

Outliers are observations in a dataset that don't fit in some way.

It is the most common or familiar type of outlier is the observations that are far from the rest of the observations.

This is easy to understand when we have one or two variables and we can visualize the data as a histogram or scatter plot, and can be challenging when we have many input variables defining a high-dimensional input feature space.

In this case, simple statistical methods for identifying outliers can break down, such as methods that use standard deviations or the interquartile range.

It can be important to identify and remove outliers from data when training machine learning algorithms for predictive modeling.

Outliers can skew statistical measures and data distributions, providing a misleading representation of the underlying data and relationships. Removing outliers from training data prior to modeling can result in a better fit of the data and, in turn, more skillful predictions.

Thankfully, there are automatic model-based methods for identifying outliers in input data. Importantly, each method approaches the definition of an outlier is slightly different ways, providing alternate approaches to preparing a training dataset that is just

like any other data preparation step in a modeling pipeline.

Before we dive into automatic outlier detection methods, let's first select a standard machine learning dataset that we can use as the basis for our investigation.

Dataset and Performance Baseline

In this section, we will first select a standard machine learning dataset and establish a baseline in p

This will provide the context for exploring the outlier identification and removal method of data prep

House Price Regression Dataset

We will use the house price regression dataset.

This dataset has 13 input variables that describe the properties of the house and suburb and requires the prediction of the median value of houses in the suburb in thousands of dollars.

You can learn more about the dataset here:

- House Price Dataset (housing.csv)
- House Price Dataset Description (housing.names)

No need to download the dataset as we will download it automatically as part of our worked examples.

Open the dataset and review the raw data. The first few rows of data are listed below.

We can see that it is a regression predictive modeling problem with numerical input variables, each of which has different scales.


```
0.00632,18.00,2.310,0,0.5380,6.5750,65.20,4.0900,1,296.0,15.30,396.90,4.98,24.00
0.02731,0.00,7.070,0,0.4690,6.4210,78.90,4.9671,2,242.0,17.80,396.90,9.14,21.60
0.02729,0.00,7.070,0,0.4690,7.1850,61.10,4.9671,2,242.0,17.80,392.83,4.03,34.70
0.03237,0.00,2.180,0,0.4580,6.9980,45.80,6.0622,3,222.0,18.70,394.63,2.94,33.40
0.06905,0.00,2.180,0,0.4580,7.1470,54.20,6.0622,3,222.0,18.70,396.90,5.33,36.20
...
```


The dataset has many numerical input variables that have unknown and complex relationships. We don't know that outliers exist in this dataset, although we may guess that some outliers may be present.

The example below loads the dataset and splits it into the input and output columns, splits it into tra s of the data arrays.


```
# load and summarize the dataset
from pandas import read_csv
from sklearn.model_selection import train_test_split
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# summarize the shape of the dataset
print(X.shape, y.shape)
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# summarize the shape of the train and test sets
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```


Running the example, we can see that the dataset was loaded correctly and that there are 506 rows of data with 13 input variables and a single target variable.

The datase est sets with 339 rows used for model training and 167 for model evaluation.

```
(506, 13) (506,)
```

```
2 (339, 13) (167, 13) (339,) (167,)
```


Next, let's evaluate a model on this dataset and establish a baseline in performance.

Baseline Model Performance

This is a regression predictive modeling problem, meaning that we will be predicting a numeric value. All input variables are also numeric.

In this case, we will fit a linear regression algorithm and evaluate model performance by training the model on the test dataset and making a prediction on the test data and evaluate the predictions using the mean absolute error (MAE).

The complete example of evaluating a linear regression model on the dataset is listed below.


```
# evaluate model on the raw dataset
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1
# fit the model
model = LinearRegression()
model.fit(X_train, y_train)
# evaluate the model
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```


Running the example fits and evaluates the model, then reports the MAE.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that the model achieved a MAE of about 3.417. This provides a baseline in performance to which we can compare different outlier identification and removal procedures.

```
1 MAE: 3.417
```

Next, we can try removing outliers from the training dataset.

Automatic Outlier Detection

The scikit-learn library provides a number of built-in automatic methods for identifying outliers in data.

In this section, we will review four methods and compare their performance on the house price dataset.

Each method will be defined, then fit on the training dataset. The fit model will then predict which ex... hich are not (so-called inliers). The outliers will then be removed from the training dataset, then the mode... ted c... entire test dataset.

It would be invalid to fit the outlier detection method on the entire training dataset as this would resu... access to data (or information about the data) in the test set not used to train the model. This may r... nce.

...ld attempt to detect outliers on "*new data*" such as the test set prior to making a prediction,

One approach might be to return a "*None*" indicating that the model is unable to make a prediction ... ng ...ion to explore that may be appropriate for your project.

Isolation Forest

Isolation Forest, or iForest for short, is a tree-based anomaly detection algorithm.

It is based on modeling the normal data in such a way as to isolate anomalies that are both few in number and different in the feature space.

> *... our proposed method takes advantage of two anomalies' quantitative properties: i) they are the minority consisting of fewer instances and ii) the... that are very different from those of normal instances.*

— Isolation Forest, 2008.

The scikit-learn library provides an implementation of Isolation Forest in the IsolationForest class.

Perhaps the most important hyperparameter in the model is the "*contamination*" argument, which is used to help estimate the number of outliers in the dataset. This is a value between 0.0 and 0.5 and by default is set to 0.1.


```
# identify outliers in the training dataset
iso = IsolationForest(contamination=0.1)
yhat = iso.fit_predict(X_train)
```

Once identified, we can remove the outliers from the training dataset.

```

# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train[mask, :], y_train[mask]
```

Putting this together, the complete example of evaluating the linear model on the housing dataset with outliers removed using isolation forest is listed below.

```
# evaluate model performance with outliers removed using isolation forest
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import IsolationForest
from sklearn.metrics import mean_absolute_error
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# summarize the shape of the training dataset
print(X_train.shape, y_train.shape)
# identify outliers in the training dataset
iso = IsolationForest(contamination=0.1)
yhat = iso.fit_predict(X_train)
# select all rows that are not outliers
```

```
# mask and select outliers
mask = yhat != -1
X_train, y_train = X_train[mask, :], y_train[mask]
# summarize the shape of the updated training dataset
print(X_train.shape, y_train.shape)
# fit the model
model = LinearRegression()
model.fit(X_train, y_train)
# evaluate the model
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```

Running the example fits and evaluates the model, then reports the MAE.

Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that that model identified and removed 34 outliers and achieved a MAE of about 3.417.

```
(339, 13) (339,)
(305, 13) (305,)
MAE: 3.189
```

Minimum Covariance Determinant

If input variables have a Gaussian distribution, then simple statistical methods can be used to detect outliers.

For example, if the dataset has two input variables and both are Gaussian, then the feature space forms a multi-dimensional Gaussian and knowledge of this distribution can be used to identify values far from the distribution.

This approach can be generalized by defining a hypersphere (ellipsoid) that covers the normal data, and data that falls outside this shape is considered an outlier. An efficient implementation of this technique for multivariate data is known as the Minimum Covariance Determinant, or MCD for short.

> *The Minimum Covariance Determinant (MCD) method is a highly robust estimator of multivariate location and scatter, for which a fast algorithm is available. It also serves as a convenient and efficient tool for outlier detection.*

— Minimum Covariance Determinant and Extensions, 2017.

The scikit-learn library provides access to this method via the EllipticEnvelope class.

It provides the "*contamination*" argument that defines the expected ratio of outliers to be observed in practice. In this case, we will set it to a value of 0.01, found with a little trial and error.

```
# identify outliers in the training dataset
ee = EllipticEnvelope(contamination=0.01)
yhat = ee.fit_predict(X_train)
```

Once identified, the outliers can be removed from the training dataset as we did in the prior example.

Tying this together, the complete example of identifying and removing outliers from the housing data... ...ant determinant) method is listed below.

```
# evaluate model performance with outliers removed using elliptical envelope
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.covariance import EllipticEnvelope
from sklearn.metrics import mean_absolute_error
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# summarize the shape of the training dataset
print(X_train.shape, y_train.shape)
# identify outliers in the training dataset
ee = EllipticEnvelope(contamination=0.01)
yhat = ee.fit_predict(X_train)
# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train[mask, :], y_train[mask]
# summarize the shape of the updated training dataset
print(X_train.shape, y_train.shape)
# fit the model
```

```
# model = LinearRegression()
model.fit(X_train, y_train)
# evaluate the model
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```

Running the example fits and evaluates the model, then reports the MAE.

Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that the elliptical envelope method identified and removed only 4 outliers, resulting in a drop in MAE from 3.417 with the robust linear regression model to 3.388.

```
(339, 13) (339,)
(335, 13) (335,)
MAE: 3.388
```

Local Outlier Factor

A simple approach to identifying outliers is to locate those examples that are far from the other exam

This can work well for feature spaces with low dimensionality (few features), although it can become ed, referred to as the curse of dimensionality.

Local outlier factor, or LOF for short, is a technique that attempts to harness the idea of nearest neighbors for outlier detection. Each example is assigned a scoring of how isolated or how likely it is to be outliers based on the size of its local neighborhood. Those examples with the largest score are more likely to be outliers.

> *We introduce a local outlier (LOF) for each object in the dataset, indicating its degree of outlier-ness.*
>
> — LOF: Identifying Density-based Local Outliers, 2000.

The scikit-learn library provides an implementation of this approach in the LocalOutlierFactor class.

The model provides the "*contamination*" argument, that is the expected percentage of outliers in the dataset, be indicated and defaults to 0.1.

```
...
# identify outliers in the training dataset
lof = LocalOutlierFactor()
yhat = lof.fit_predict(X_train)
```

Tying this together, the complete example of identifying and removing outliers from the housing dataset using the local outlier factor method is listed below.

```
# evaluate model performance with outliers removed using local outlier factor
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import mean_absolute_error
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# summarize the shape of the training dataset
print(X_train.shape, y_train.shape)
# identify outliers in the training dataset
lof = LocalOutlierFactor()
yhat = lof.fit_predict(X_train)
# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train[mask, :], y_train[mask]
# summarize the shape of the updated training dataset
print(X_train.shape, y_train.shape)
# fit the model
model = LinearRegression()
model.fit(X_train, y_train)
# evaluate the model
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```

Running the example fits and evaluates the model, then reports the MAE.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that the local outlier factor method identified and removed 34 outliers, the same number as isolation forest, resulting in a drop in MAE from 3.417 with the baseline to 3.356. Better, but not as good as isolation forest, suggesting a different set of outliers were identified and removed.

```
(339, 13) (339,)
(305, 13) (305,)
MAE: 3.356
```


One-Class SVM

The support vector machine, or SVM, algorithm developed initially for binary classification can be used for one-class classification.

When modeling one class, the algorithm captures the density of the majority class and classifies ex[...]s outliers. This modification of SVM is referred to as One-Class SVM.

> *… an algorithm that computes a binary function that is supposed to capture regions in input […] support), that is, a function such that most of the data will live in the region where the functio[…]*

— Estimating the Support of a High-Dimensional Distribution, 2001.

Although SVM is a classification algorithm and One-Class SVM is also a classification algorithm, it c[…]both regression and classification datasets.

The scikit-learn library provides an implementation of one-class SVM in the OneClassSVM class.

The class provides the "nu" argument that specifies the approximate ratio of outliers in the dataset, which defaults to 0.1. In this case, we will set it to 0.01, found with a little trial and error.


```
...
# identify outliers in the training dataset
ee = OneClassSVM(nu=0.01)
yhat = ee.fit_predict(X_train)
```


Tying this together, the complete example of identifying and removing outliers from the housing data[…]ow.

```
# evaluate model performance with outliers removed using one class SVM
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.svm import OneClassSVM
from sklearn.metrics import mean_absolute_error
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
df = read_csv(url, header=None)
# retrieve the array
data = df.values
# split into input and output elements
X, y = data[:, :-1], data[:, -1]
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# summarize the shape of the training dataset
print(X_train.shape, y_train.shape)
# identify outliers in the training dataset
ee = OneClassSVM(nu=0.01)
yhat = ee.fit_predict(X_train)
# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train[mask, :], y_train[mask]
# summarize the shape of the updated training dataset
print(X_train.shape, y_train.shape)
# fit the model
model = LinearRegression()
model.fit(X_train, y_train)
# evaluate the model
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```


Running the example fits and evaluates the model, then reports the MAE.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

In this case, we can see that 67 outliers were identified and removed and the model achieved a MAE of about 3.431, which is not better than the baseline model that achieved 3.417. Perhaps better performance can be achieved with more tuning.

```
(339, 13) (339,)
(336, 13) (336,)
MAE: 3.431
```

Further Reading

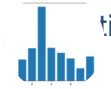
This section provides more resources on the topic if you are looking to go deeper.

Related Tutorials

- How to Choose a Feature Selection Method For Machine Learning
- One-Class Classification Algorithms for Imbalanced Datasets
- How to Remove Outliers for Machine Learning

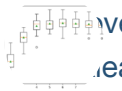
Papers

- Data Preparation for Machine Learning (7-Day Mini-Course)
- Isolation Forest, 2008.
- Minimum Covariance Determinant and Extensions, 2017.
- LOF: Identifying Density-based Local Outliers, 2000.
- Estimating the Support of a High-Dimensional Distribution, 2001.



- How to Calculate Feature Importance With Python

APIs



- Recursive Feature Elimination (RFE) for Feature Selection in Python
- Novelty and Outlier Detection, scikit-learn user guide.
- sklearn.covariance.EllipticEnvelope API.
- sklearn.svm.OneClassSVM API.
- sklearn.neighbors.LocalOutlierFactor API.

- How to Remove Outliers for Machine Learning
- sklearn.ensemble.IsolationForest API.

Summary

In this tutorial, you discovered how to use automatic outlier detection and removal to improve machine learning predictive modeling performance.

Specifically, you learned:

- Automatic outlier detection models provide an alternative to statistical techniques with a larger number of input variables with complex and unknown
- How to correctly apply automatic outlier detection and removal to the training dataset only to a

- How to evaluate and compare predictive modeling pipelines with outliers removed from the training dataset.

Never miss a tutorial:

Do you have any questions?

Ask your questions in the comments below and I will do my best to answer.

Picked for you:

How to Choose a Feature Selection Method For Machine Learning

Data Preparation for Machine Learning (7-Day Mini-Course)

How to Calculate Feature Importance With Python

Recursive Feature Elimination (RFE) for Feature Selection in Python

How to Remove Outliers for Machine Learning

Loving the tutorials?

The Data Preparation EBook is
where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on

View all posts by Jason Brownlee →

‹ How to Use Feature Extraction on Tabular Data for Machine Learning
6 Dimensionality Reduction Algorithms With Python ›

38 Responses to *4 Automatic Outlier Detection Algorithms in Python*

Joseph July 8, 2020 at 7:00 pm #

REPLY

Hi Jason, thanks for one more great article!

I question if removal of outliers is needed for algorithms (RF, XGboost). Does it really change model outcomes in real life to delete outliers in this case? Things change over time, that's why I've this question.

Jason Brownlee July 9, 2020 at 6:39 am #

I think trees are pretty robust to outliers. Test for your dataset.

JParzival July 9, 2020 at 10:42 am #

Great article!

Thank you for sharing your experience!

Jason Brownlee July 9, 2020 at 1:19 pm #

REPLY

Nagdev. A July 10, 2020 at 9:39 am #

REPLY

Two re t e lis oencoders and PCA

How to Choose a Feature Selection Method For Machine Learning

Jason Brownlee July 10, 2020 at 1:47 pm #

REPLY

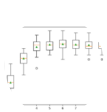
For outlier detection? How so?

Data Preparation for Machine Learning (7-Day Mini-Course)

Ali November 19, 2020 at 4:20 pm #

Actually, autoencoders can provide best performance for anomaly detection problems

How to Calculate Feature Importance With Python



Recursive Feature Elimination (RFE) for Feature Selection in Python

Jason Brownlee November 20, 2020 at 6:43 am #

Depends on the specific dataset.

How to Remove Outliers for Machine Learning

Ali November 19, 2020 at 4:27 pm #

REPLY

Both Autoencoder and PCA are dimensionality reduction techniques. Interestingly, during the process of dimensionality reduction outliers are identified.

pm #

REPLY

Hello sir,
It was a great article. Just one doubt:
MCD technique does not perform well when the data has very large dimensions like >1000. In that case, it is a good option to feed the model with principal components of the data. The paper that you mentioned in the link says:

"For large p we can still make a rough estimate of the scatter as follows. First compute the first q < p robust principal components of the data. For this we can use the MCD-based ROBPCA method53, which requires that the number of components q be set rather low."

But the MCD PCA is not available in python. Can you please tell what can be done in this case?

Thank you

Jason Brownlee July 11, 2020 at 6:13 am #

Great tip, thanks.

Perhaps find a different platform that implements the method?

Perhaps implement it yourself?

Perhaps use a different method entirely?

fabou July 10, 2020 at 11:08 pm #

Hi Jason,

as usual great educational article.

How could automatic outlier detection be integrated into a cross validation loop? Does it have to be part of a pipeline which steps would be : outlier detection > outlier removal (transformer) > modeling?

In this case, should a specific transformer "outlier remover" be created?

Thanks

Jason Brownlee July 11, 2020 at 6:16 am #

I would have run the CV loop manually and apply the method to the data prior to fitting/evaluating a model or pipeline.

It's disappointing that sklearn does not support methods in pipelines that add/remove rows. imbalanced learn can do this kind of thing…

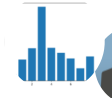
How to Choose a Feature Selection Method For Machine Learning

Chayma July 15, 2020 at 1:16 am #

Thank you for the great article.

Data Preparation for Machine Learning (7-Day Mini-Course)

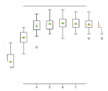
Which algorithm is the most sutible for outlier detection in time series data?



How to Calculate Feature Importance With Python

Jason Brownlee July 15, 2020 at 8:27 am #

I don't know off hand, I hope to write about that topic in the future.



Recursive Feature Elimination (RFE) for Feature Selection in Python

Nagesh August 27, 2020 at 12:33 pm #

How to Remove Outliers for Machine Learning

Thoughts on this onne ? https://github.com/arundo/adtk

Jason Brownlee August 27, 2020 at 1:36 pm #

I'm not familiar with it.

Allen21 September 1, 2020 at 5:54 am #

If anyone is getting a TypeError with X_train[mask, :], just change it to X_train[mask]. Another great article BTW

REPLY

Jason Brownlee September 1, 2020 at 6:38 am #

Sorry to hear that.

Perhaps these tips will help:

https://machinelearningmastery.com/faq/single-faq/why-does-the-code-in-the-tutorial-not-work-for-me

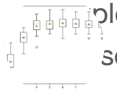
Vidya Manu Shankar September 10, 2020 at 1:51 pm #

Hi Jason .

Thanks for this post.

Couple of questions though:

1. How do we validate the output of the outlier detection algorithms mentioned in this post , whether the clusters ?



2. So , why don't we include the target variable as input to the outlier algorithms ? I missed this point .

Jason Brownlee September 11, 2020 at 5:48 am #

Thanks.

Good question, you can validate the model by either evaluating predictions on dataset with known outliers or inspecting identified outliers and using a subject matter expert to determine if they are true outliers or not.

The algorithms are one-class algorithms, no target variable is required.

Vidya September 12, 2020 at 2:38 am #

Awesome , thank you !

REPLY

Jason Brownlee September 12, 2020 at 6:18 am #

You're welcome.

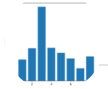
How to Choose a Feature Selection Method For Machine Learning

REPLY

arnan maipradit September 13, 2020 at 10:29 am #

Data Preparation for Machine Learning (7-Day Mini-Course)

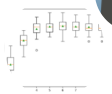
do you have any example of outlier detection using Q-learning, I found that Q-learning almost using in case of many actions (robot move up down right so it has 4 actions) but in the case of outlier detection it has only 2 actions (normal behavior an can be used on outlier detection (anomaly detection) or not ? . If you could make an example or sugges



How to Calculate Feature Importance With Python

REPLY

Jason Brownlee September 14, 2020 at 6:44 am #



cursive Feature Elimination (RFE) for Feature Selection in Python

Sorry, I do not have any examples or RL at this stage.

Thanks for the suggestion.

How to Remove Outliers for Machine Learning

REPLY

Arushi Mahajan October 14, 2020 at 1:45 pm #

Hi, amazing tutorial.

Just one question. How can you see all the rows that were dropped?

The Data Preparation EBook is where you'll find the ***Really Good*** stuff.

REPLY

ctober 14, 2020 at 1:51 pm #

What do you mean by "dropped rows"?

Never miss a tutorial:

Jason Brownlee December 7, 2020 at 6:22 am #

REPLY

don[illegible]e example only applies the automatic methods to the training dataset.

Mitra December 20, 2020 at 1:42 am #

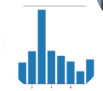
REPLY

How to Choose a Feature Selection Method For Machine Learning

Hi sir! Thank you for the amazing content, Just wanted to point out one thing. In the Isolation Forests, documentation of Scikit learn I read that the default value for contamination is no longer 0.1 and it's turned to auto. You can correct that part 🙂

Data Preparation for Machine Learning (7-Day Mini-Course)

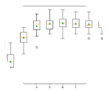
Jason Brownlee December 20, 2020 at 5:58 am #



How to Calculate Feature Importance With Python

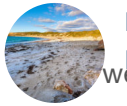
You're welcome.

Thanks for the suggestion.



Recursive Feature Elimination (RFE) for Feature Selection in Python

Mitra December 20, 2020 at 4:49 am #



How to Remove Outliers for Machine Learning

One quick note! In the Minimum Covariance Determination method, you said we can use this m[illegible] well in the dataset you're using the features don't have such shape. Most of them are skewed. I think we should first apply a transformation(log, box-cox, etc.) and then use this method on features with little or no skewness. I'm actually writing a Kaggle kernel on this and would love to hear what you think about it when it's done!

Jason Brownlee December 20, 2020 at 5:59 am #

REPLY

nend evaluating the approach with and without the data prep and use the approach that results in the best performance.

Never miss a tutorial:

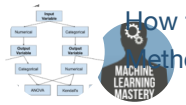
Divya Jain December 29, 2020 at 2:18 am #

REPLY

Am g tu l Si

Question- Should we always drop the rows containing outliers? Will outlier imputation work better in some cases?

Picked for you:



How to Choose a Feature Selection Method For Machine Learning

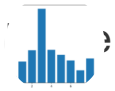
Jason Brownlee December 29, 2020 at 5:15 am #

REPLY

Thanks.

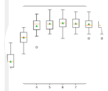
It is Data Preparation for Machine Learning (7-Day Mini-Course)

It is a prediction project.

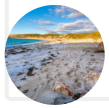


How to Calculate Feature Importance With Python

e a Reply



Recursive Feature Elimination (RFE) for Feature Selection in Python



How to Remove Outliers for Machine Learning

Name (required)

Loving the Tutorials?

The Data Preparation EBook is where you'll find the ***Really Good*** stuff.

Email (will not be published) (required)

>> SEE WHAT'S INSIDE

Website

Start Machine Learning

Never miss a tutorial:

SUBMIT COMMENT

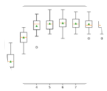
Picked for you:

How to Choose a Feature Selection Method For Machine Learning

Read more

Data Preparation for Machine Learning (7-Day Mini-Course)

How to Calculate Feature Importance With Python



Recursive Feature Elimination (RFE) for Feature Selection in Python

How to Remove Outliers for Machine Learning

Loving the Tutorials?

The Data Preparation EBook is where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE

Welcome!

I'm *Jason Brownlee* PhD and I **help developers** get results with **machine learning**.

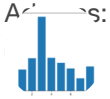
Start Machine Learning

Never miss a tutorial:

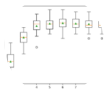
Picked for you:

How to Choose a Feature Selection Method For Machine Learning

Data Preparation for Machine Learning (7-Day Mini-Course)



How to Calculate Feature Importance With Python



Recursive Feature Elimination (RFE) for Feature Selection in Python

How to Remove Outliers for Machine Learning