

[🔗 Outline](#) > 2. Data And Features > Lecture: Wrangling Data > Dropping Data

## Dropping Data

[🔖 Bookmark this page](#)

You should always first try to fill in missing data rather than deleting it. This is so important that we've included a link in the dive deeper section that provides a very comprehensive argument and explanation for this. But if all else fails and you've given up on rectifying your nans, you can always remove the sample or column completely, so that it no longer negatively impacts your analysis. This should ever be used as a last resort:

```
df = df.dropna(axis=0) # remove any row with nans
df = df.dropna(axis=1) # remove any column with nans

# Drop any row that has at least 4 NON-NaN within it:
df = df.dropna(axis=0, thresh=4)
```

There may be cases where you want to get rid of non-nan values. For instance, if your dataset has a column you don't need:

```
# Axis=1 for columns
df = df.drop(labels=['Features', 'To', 'Delete'], axis=1)
```

You might also want to prune duplicate records if samples cannot have identical properties. Be careful though! To get rid of duplicate records, you should tell Pandas which features are to be examined, because Pandas generates indices for you automatically when you load a dataframe without specifying an index column. With each column having a unique index, Pandas won't find any 'duplicates' unless you limit your search to a subset of your dataframe's features:

```
df = df.drop_duplicates(subset=['Feature_1', 'Feature_2'])
```

Removing duplicate samples will cause gaps to occur in your index count. You can interpolate to fill those holes where appropriate, or alternatively you can reindex your dataframe:

```
df = df.reset_index(drop=True)
```

The `drop=True` parameter tells Pandas not to keep a backup copy of the original index. Most, if not all, of the above methods return a copy of your dataframe. This is useful because you can chain methods:

```
df = df.dropna(axis=0, thresh=2).drop(labels=['ColA', axis=1]).drop_duplicates(subset=['ColB', 'ColC']).reset_index()
```

However there may be times where you want these operations to work in-place on the dataframe calling them, rather than returning a new dataframe. Pass `inplace=True` as a parameter to any of the above methods to get that working.

© All Rights Reserved



🌐 English ▾

© 2012-2017 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc. | 粤ICP备17044299号-2

POWERED BY  
OPENedX®

