# How To Drop Rows In Pandas DataFrames With NaN Values In Certain Columns

*DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors=’raise’)*

* **labels:**String or list of strings referring row.
* **axis:**int or string value, 0 ‘index’ for Rows and 1 ‘columns’ for Columns.
* **index or columns:**Single label or list. index or columns are an alternative to axis and cannot be used together.
* **level:**Used to specify level, in case data frame is having multiple level index.
* **inplace:**Makes changes in original Data Frame if True.
* **errors:**Ignores error if any value from the list doesn’t exists and drops rest of the values when errors = ‘ignore’

**Introduction**

In today’s short guide we are going to explore a few ways for dropping rows from pandas DataFrames that have null values in certain column(s). Specifically, we’ll discuss how to drop rows with:

* at least one column being NaN
* all column values being NaN
* specific column(s) having null values
* at least N columns with non-null values

First, let’s create an example DataFrame that we’ll reference in order to demonstrate a few concepts throughout this article.

import pandas as pddf = pd.DataFrame({  
 'colA':[None, False, False, True],   
 'colB': [None, 2, None, 4],  
 'colC': [None, 'b', 'c', 'd'],  
 'colD': [None, 2.0, 3.0, 4.0],  
})print(df)  
 *colA colB colC colD  
0 None NaN None NaN  
1 False 2.0 b 2.0  
2 False NaN c 3.0  
3 True 4.0 d 4.0*

**Drop all rows having at least one null value**

When it comes to dropping null values in pandas DataFrames, [pandas.DataFrame.dropna()](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html" \t "_blank) method is your friend. When you call dropna() over the whole DataFrame without specifying any arguments (i.e. using the default behaviour) then the method will drop all rows with at least one missing value.

**df = df.dropna()**print(df)  
 *colA colB colC colD  
1 False 2.0 b 2.0  
3 True 4.0 d 4.0*

**Drop rows having only missing values**

Now if you want to drop all the rows whose columns’ values are all null, then you need to specify how='all' argument.

**df = df.dropna(how='all')**print(df)  
 *colA colB colC colD  
1 False 2.0 b 2.0  
2 False NaN c 3.0  
3 True 4.0 d 4.0*

**Drop rows where specific column values are null**

If you want to take into account only specific columns, then you need to specify the subset argument.

For instance, let’s assume we want to drop all the rows having missing values in any of the columns colA or colC :

**df = df.dropna(subset=['colA', 'colC'])**print(df)  
 *colA colB colC colD  
1 False 2.0 b 2.0  
2 False NaN c 3.0  
3 True 4.0 d 4.0*

Additionally, you can even drop all rows if they’re having missing values in both colA and colB:

**df = df.dropna(subset=['colA', 'colB'], how='all')**print(df)  
 *colA colB colC colD  
1 False 2.0 b 2.0  
2 False NaN c 3.0  
3 True 4.0 d 4.0*

**Drop rows with at least N non-missing values**

Finally, if you need to drop all the rows that have at least N columns with non- missing values, then you need to specify the **thresh argument that specifies the number of non-missing values that should be present for each row in order not to be dropped**.

For instance, if you want to drop all the columns that have more than one null values, then you need to specify thresh to be len(df.columns) — 1

**df = df.dropna(thresh=len(df.columns)-1)**print(df)  
 *colA colB colC colD  
1 False 2.0 b 2.0  
2 False NaN c 3.0  
3 True 4.0 d 4.0*

**Final Thoughts**

In today’s short guide, we discussed 4 ways for dropping rows with missing values in pandas DataFrames.

Note that there may be many different methods (e.g. numpy.isnan() method) you can use in order to drop rows (and/or columns) other than pandas.DataFrame.dropna(),the latter has been built explicitly for pandas and it comes with an improved performance when compared against more generic methods.