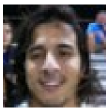
Finding the Percentage of Missing Values in a Pandas DataFrame

medium.com/dunder-data/finding-the-percentage-of-missing-values-in-a-pandas-dataframe-a04fa00f84ab

Ted Petrou November 23, 2018



Ted Petrou

Nov 23

In this tutorial, we will cover an efficient and straightforward method for finding the percentage of missing values in a Pandas DataFrame. This tutorial is available as a video on YouTube. I plan on making video tutorials for the entire library soon, so subscribe if you want to stay updated.

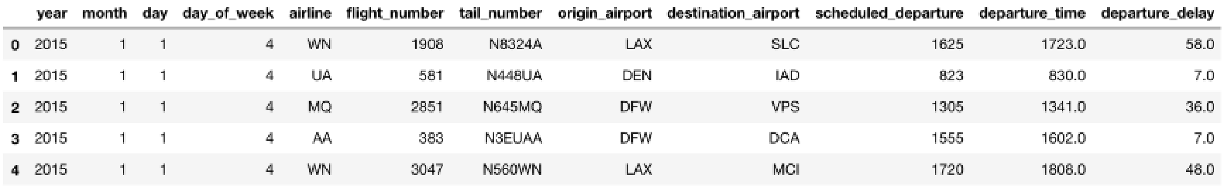
Non-intuitive Solution

The final solution to this problem is not quite intuitive for most people when they first encounter it. We will slowly build up to it and also provide some other methods that get us a result that is close but not exactly what we want.

Flights Dataset

We begin by reading in the flights dataset, which contains US domestic flight information during the year 2015. Pandas defaults the number of visible columns to 20. Since there are 31 columns in this DataFrame, we change this option below.

```
>>> import pandas as pd
>>> pd.options.display.max_columns = 100
>>> pd.read_csv('flights.csv')
>>> flights.head()
```



	year	month	day	day_of_week	airline	flight_number	tail_number	origin_airport	destination_airport	scheduled_departure	departure_time	departure_delay
0	2015	1	1	4	WN	1908	N8324A	LAX	SLC	1625	1723.0	58.0
1	2015	1	1	4	UA	581	N448UA	DEN	IAD	823	830.0	7.0
2	2015	1	1	4	MQ	2851	N645MQ	DFW	VPS	1305	1341.0	36.0
3	2015	1	1	4	AA	383	N3EUAA	DFW	DCA	1555	1602.0	7.0
4	2015	1	1	4	WN	3047	N560WN	LAX	MCI	1720	1808.0	48.0

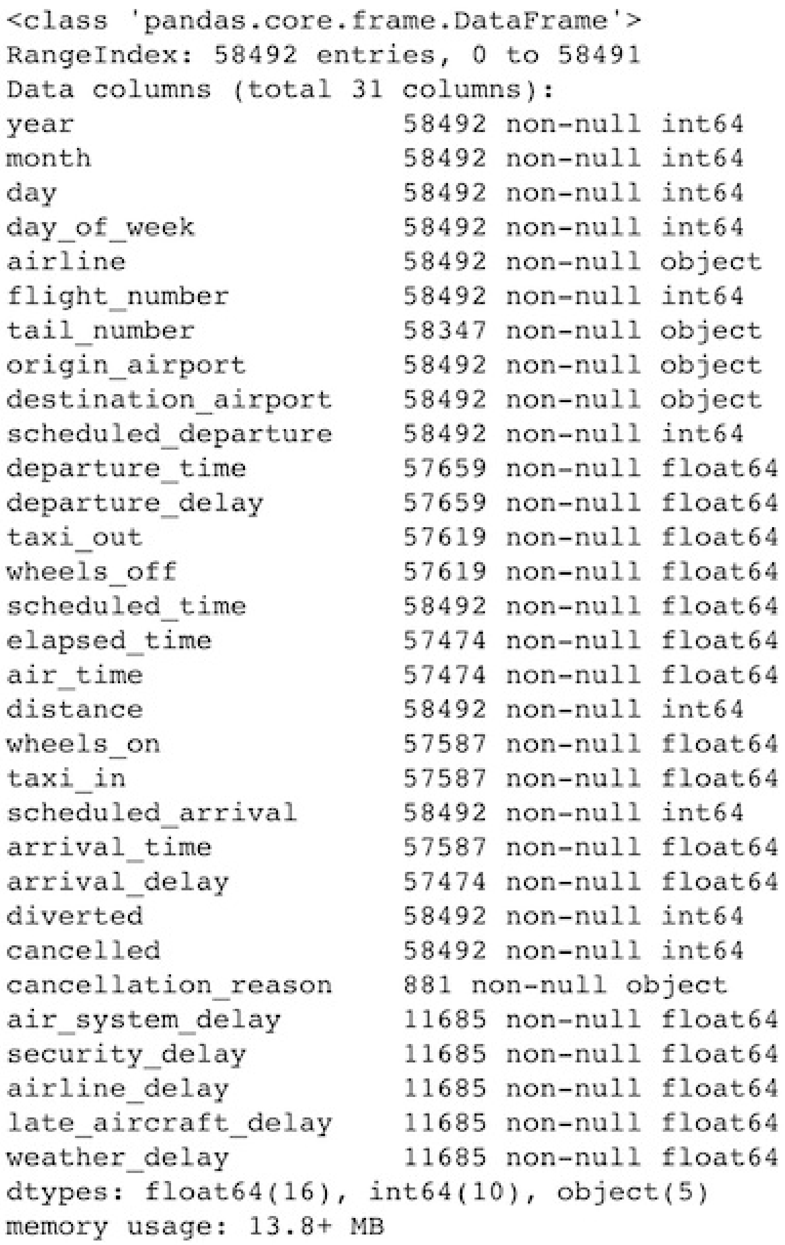
After inspecting the first few rows of the DataFrame, it is generally a good idea to find the total number of rows and columns with the `shape` attribute.

```
>>> flights.shape
(58492, 31)
```

The info method

Pandas comes with a couple methods that get us close to what we want without getting us all the way there. The `info` method prints to the screen the number of non-missing values of each column, along with the data types of each column and some other meta-data.

```
>>> flights.info()
```

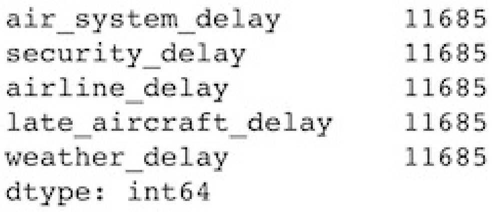


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58492 entries, 0 to 58491
Data columns (total 31 columns):
year                     58492 non-null int64
month                    58492 non-null int64
day                      58492 non-null int64
day_of_week              58492 non-null int64
airline                  58492 non-null object
flight_number            58492 non-null int64
tail_number              58347 non-null object
origin_airport           58492 non-null object
destination_airport      58492 non-null object
scheduled_departure      58492 non-null int64
departure_time           57659 non-null float64
departure_delay          57659 non-null float64
taxi_out                 57619 non-null float64
wheels_off               57619 non-null float64
scheduled_time           58492 non-null float64
elapsed_time             57474 non-null float64
air_time                 57474 non-null float64
distance                 58492 non-null int64
wheels_on                57587 non-null float64
taxi_in                  57587 non-null float64
scheduled_arrival        58492 non-null int64
arrival_time             57587 non-null float64
arrival_delay            57474 non-null float64
diverted                 58492 non-null int64
cancelled                58492 non-null int64
cancellation_reason      881 non-null object
air_system_delay         11685 non-null float64
security_delay           11685 non-null float64
airline_delay            11685 non-null float64
late_aircraft_delay      11685 non-null float64
weather_delay            11685 non-null float64
dtypes: float64(16), int64(10), object(5)
memory usage: 13.8+ MB
```

The `count` method

The `count` method returns the number of non-missing values for each column or row. By default, it operates column-wise. It doesn't give us any more information that is already available with the `info` method. Below, we just output the last 5 values.

```
>>> flights.count().tail()
```



```
air_system_delay       11685
security_delay         11685
airline_delay          11685
late_aircraft_delay    11685
weather_delay          11685
dtype: int64
```

The returned objects of the info and count methods

Although the `count` method doesn't yield us any extra information over the `info` method, it returns a Pandas `Series` which can be used for further processing. The `info` method prints its output to the screen and returns the object `None`. Let's verify this below:

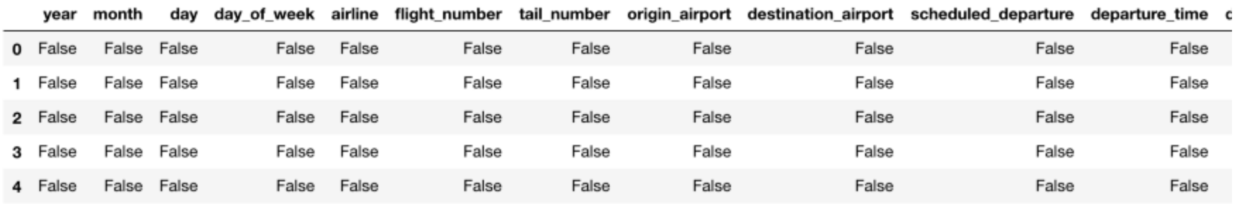
```
>>> info_return = flights.info()
>>> count_return = flights.count()
>>> info_return is None
True

>>> type(count_return)
pandas.core.series.Series
```

The `isna` method

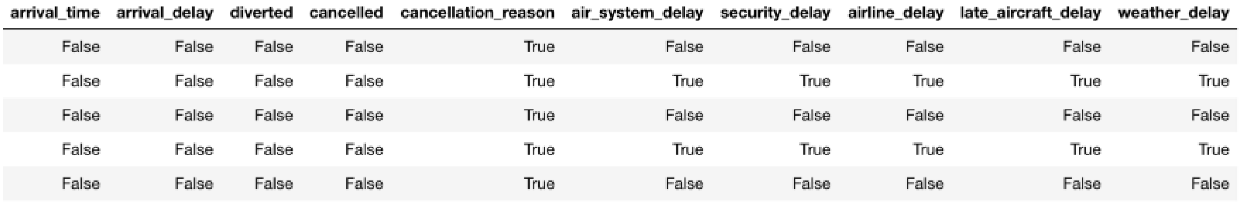
The `isna` method returns a DataFrame of all boolean values (True/False). The shape of the DataFrame does not change from the original. Each value is tested whether it is missing or not. If it is, then its new value is `True` otherwise it is `False`.

```
>>> flights_missing = flights.isna()
>>> flights_missing.head()
```



	year	month	day	day_of_week	airline	flight_number	tail_number	origin_airport	destination_airport	scheduled_departure	departure_time	c
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	

The first several columns don't have any missing values in their first few rows, but if we scroll to the end, we can see many missing values do exist.



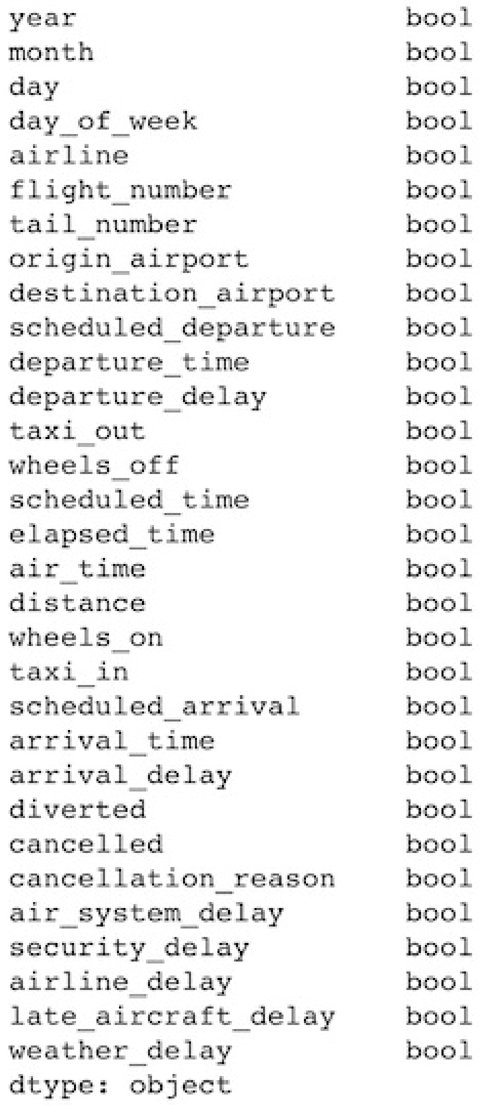
arrival_time	arrival_delay	diverted	cancelled	cancellation_reason	air_system_delay	security_delay	airline_delay	late_aircraft_delay	weather_delay
False	False	False	False	True	False	False	False	False	False
False	False	False	False	True	True	True	True	True	True
False	False	False	False	True	False	False	False	False	False
False	False	False	False	True	True	True	True	True	True
False	False	False	False	True	False	False	False	False	False

Ensuring we have all boolean columns

We can verify that result of calling the `isna` method is indeed a new DataFrame with all columns having the boolean data type. Let's call the `dtypes` attribute.

```
>>> flights_missing.dtypes
```

This guarantees us that every single value in the entire is either True of False.

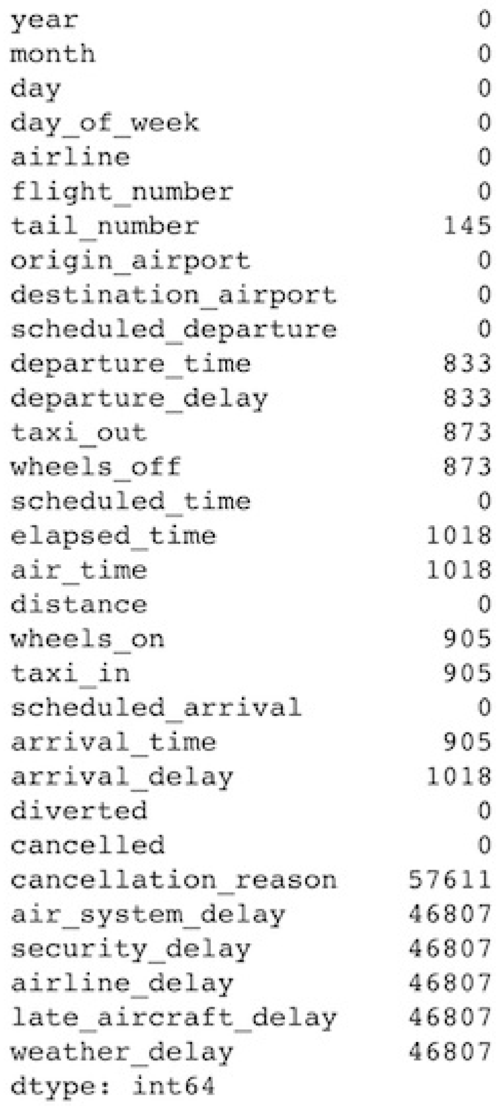


```
year                   bool
month                  bool
day                    bool
day_of_week            bool
airline                bool
flight_number          bool
tail_number            bool
origin_airport         bool
destination_airport    bool
scheduled_departure    bool
departure_time         bool
departure_delay        bool
taxi_out               bool
wheels_off             bool
scheduled_time         bool
elapsed_time           bool
air_time               bool
distance               bool
wheels_on              bool
taxi_in                bool
scheduled_arrival      bool
arrival_time           bool
arrival_delay          bool
diverted               bool
cancelled              bool
cancellation_reason    bool
air_system_delay       bool
security_delay         bool
airline_delay          bool
late_aircraft_delay    bool
weather_delay          bool
dtype: object
```

Summing a boolean DataFrame

Booleans in Python are treated as numeric when doing arithmetic operations. False evaluates as 0 and True evaluates as 1. Therefore, we can call the `sum` method on the DataFrame, which by default sums each column independently.

```
>>> flights_num_missing = flights_missing.sum()
>>> flights_num_missing
```



```
year                       0
month                      0
day                        0
day_of_week                0
airline                    0
flight_number              0
tail_number              145
origin_airport             0
destination_airport        0
scheduled_departure        0
departure_time           833
departure_delay          833
taxi_out                 873
wheels_off               873
scheduled_time             0
elapsed_time            1018
air_time                1018
distance                   0
wheels_on                905
taxi_in                  905
scheduled_arrival          0
arrival_time             905
arrival_delay           1018
diverted                   0
cancelled                  0
cancellation_reason    57611
air_system_delay       46807
security_delay         46807
airline_delay          46807
late_aircraft_delay    46807
weather_delay          46807
dtype: int64
```

Interpreting this output

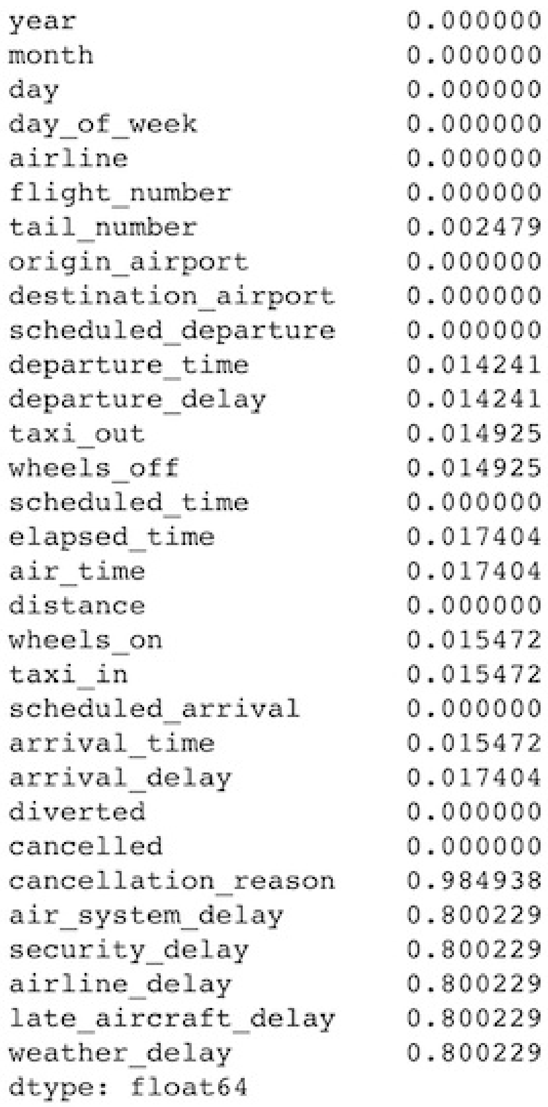
We summed up each column in the boolean DataFrame, which is summing up just False and True values. This result simply returns the number of values that are True. In our case the True values represent missing values in our original DataFrame, so we now have the number of missing values in each column.

Turning this result into a percentage

Now that we have the total number of missing values in each column, we can divide each value in the Series by the number of rows. The built-in `len` function returns the number of rows in the DataFrame.

```
>>> len(flights)
58492
```

```
>>> flights_num_missing / len(flights)
```



```
year                   0.000000
month                  0.000000
day                    0.000000
day_of_week            0.000000
airline                0.000000
flight_number          0.000000
tail_number            0.002479
origin_airport         0.000000
destination_airport    0.000000
scheduled_departure    0.000000
departure_time         0.014241
departure_delay        0.014241
taxi_out               0.014925
wheels_off             0.014925
scheduled_time         0.000000
elapsed_time           0.017404
air_time               0.017404
distance               0.000000
wheels_on              0.015472
taxi_in                0.015472
scheduled_arrival      0.000000
arrival_time           0.015472
arrival_delay          0.017404
diverted               0.000000
cancelled              0.000000
cancellation_reason    0.984938
air_system_delay       0.800229
security_delay         0.800229
airline_delay          0.800229
late_aircraft_delay    0.800229
weather_delay          0.800229
dtype: float64
```

What simple operation did we just do?

There is a name for the operation that we just completed. Summing up all the values in a column and then dividing by the total number is the **mean**.

Using the `mean` method directly

Instead of calling the `sum` method and dividing by the number of rows, we can call the `mean` method directly on the `flights_missing` DataFrame. This produces the exact same result as the last output.

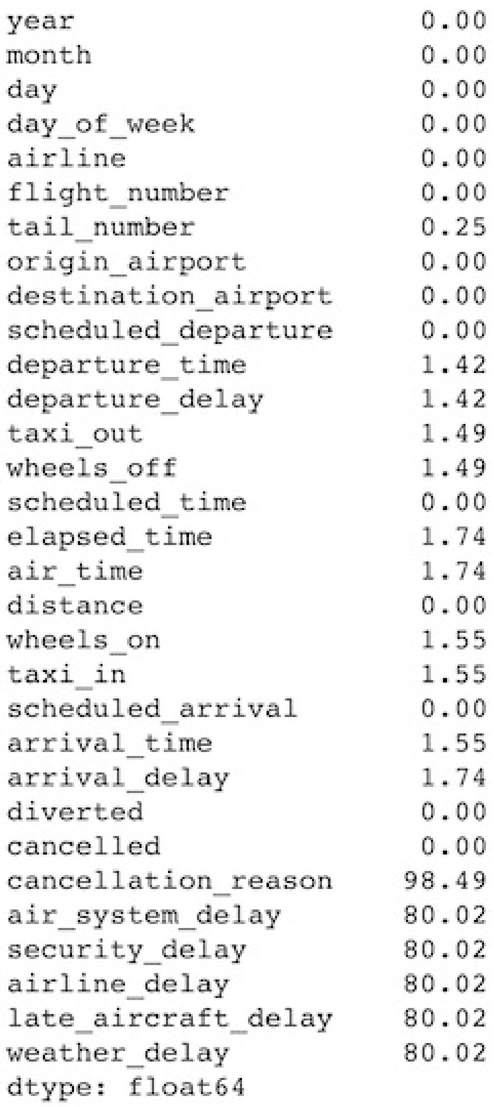
```
>>> flights_missing.mean()
```

The output isn’t shown because it is the exact same as the last.

A one-line solution

We can put all our work together in a single line of code beginning with our flights DataFrame. We remove excess decimal noise by rounding and then multiply each value by 100 to get a percentage.

```
>>> flights.isna().mean().round(4) * 100
```



```
year                     0.00
month                    0.00
day                      0.00
day_of_week              0.00
airline                  0.00
flight_number            0.00
tail_number              0.25
origin_airport           0.00
destination_airport      0.00
scheduled_departure      0.00
departure_time           1.42
departure_delay          1.42
taxi_out                 1.49
wheels_off               1.49
scheduled_time           0.00
elapsed_time             1.74
air_time                 1.74
distance                 0.00
wheels_on                1.55
taxi_in                  1.55
scheduled_arrival        0.00
arrival_time             1.55
arrival_delay            1.74
diverted                 0.00
cancelled                0.00
cancellation_reason     98.49
air_system_delay        80.02
security_delay          80.02
airline_delay           80.02
late_aircraft_delay     80.02
weather_delay           80.02
dtype: float64
```

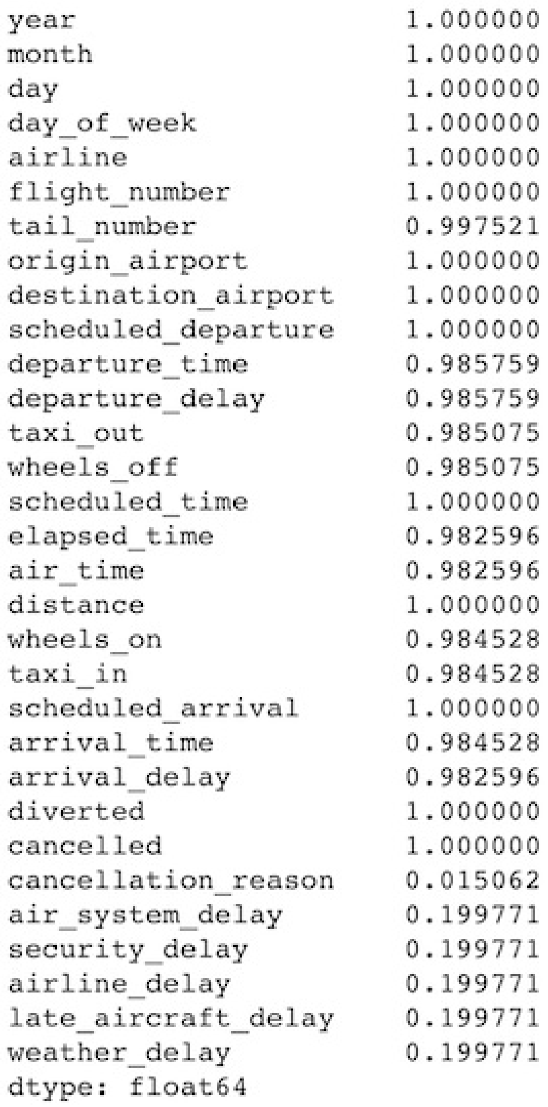
An alternative with the `count` method

Alternatively, we can get the same result by taking the result of the `count` method and dividing by the number of rows. This gives us the percentage of non-missing values in each column.

```
>>> flights.count() / len(flights)
```

From here, we can subtract each value in the Series from 1 to get the same result as the one-line solution from above. Again, the output has not been shown.

```
>>> 1 - flights.count() / len(flights)
```



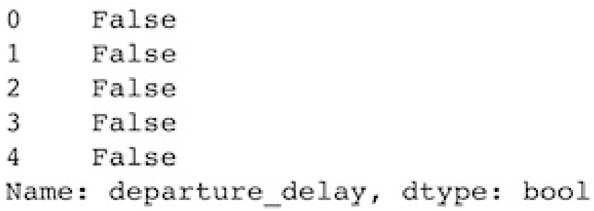
```
year                   1.000000
month                  1.000000
day                    1.000000
day_of_week            1.000000
airline                1.000000
flight_number          1.000000
tail_number            0.997521
origin_airport         1.000000
destination_airport    1.000000
scheduled_departure    1.000000
departure_time         0.985759
departure_delay        0.985759
taxi_out               0.985075
wheels_off             0.985075
scheduled_time         1.000000
elapsed_time           0.982596
air_time               0.982596
distance               1.000000
wheels_on              0.984528
taxi_in                0.984528
scheduled_arrival      1.000000
arrival_time           0.984528
arrival_delay          0.982596
diverted               1.000000
cancelled              1.000000
cancellation_reason    0.015062
air_system_delay       0.199771
security_delay         0.199771
airline_delay          0.199771
late_aircraft_delay    0.199771
weather_delay          0.199771
dtype: float64
```

Generalizing boolean operations

This same idea can be used for any boolean Series/DataFrame. Let’s see how we can find the number and percentage of flights that have arrival delays longer than 60 minutes.

First, let’s determine whether each flight has an arrival delay greater than 60 minutes by using a boolean comparison and assigning the result to a variable.

```
>>> gt_60 = flights['departure_delay'] > 60
>>> gt_60.head()
```



```
0    False
1    False
2    False
3    False
4    False
Name: departure_delay, dtype: bool
```

Use sum and mean methods to find total and percentage

Once we create a boolean Series/DataFrame, we can use the `sum` and `mean` methods to find the total and percentage of values that are True. In this case, it’s the number and percentage of flights with arrival delays greater than one hour.

```
>>> gt_60.sum()
3626

>>> gt_60.mean()
0.06199138343705122
```

We can now report that we have 3,626 flights or 6.2% that have arrival delays greater than one hour.

The two-step process of finding the total or percentage of True values

We can boil the idea down to two steps.

1. Create a boolean Series/DataFrame
2. Call `sum` to find the number of True values and `mean` to find the percentage of True values

Conclusion

An efficient and straightforward way exists to calculate the percentage of missing values in each column of a Pandas DataFrame. It can be non-intuitive at first, but once we break down the idea into summing booleans and dividing by the number of rows, it’s clear that we can use the `mean` method to provide a direct result. This idea can be generalized to any boolean Series or DataFrame.