CS167: Machine Learning

Missing data handling

Wednesday, September 2nd, 2026

Today’s Agenda

- Notebook # 1 Released

Notebook #1 Help

- A few helpful functions for Notebook #1:
  - `max()` will return the maximum value in a Series
  - `idxmax()` will return the **index** of the maximum value

```
[ ]  #find the deck of the passenger who was the oldest on the titanic
     titanic.age.max()

     80.0

[10] ndx = titanic.age.idxmax() ## returns the name of the 'row' and NOT integer index of the row
     print(ndx)
     titanic.loc[ndx].embark_town

     630
     'Southampton'
```

Notebook #1 Help

- Other functions worth mentioning:
  - `min()` will return the minimum value in a Series
  - `mode()` will return the mode — the most common value of the dataset

```
titanic.age.min()

0.42
```

```
[12] titanic.age.mode()

0    24.0
Name: age, dtype: float64
```

Help

- Pandas Exercises Solutions (Day03 Solutions)

  - The solution will be posted on Blackboard by Wednesday. I want you to try it first and then review the solution for your own verification.

Missing Data

- Most datasets you will work with will not be in perfect shape
  - you'll need to "clean" the data before you can run any machine learning algorithms on it.

- Missing data is a pretty common thing — so much so that there's a special value for missing data:
  - `NaN`, or not a number.

Missing Data

3Ds
Detect
Determine
Decide

- The steps of cleaning data normally include:
  - Step 1: Detecting which columns have missing data
  - Step 2: Determining how much data is missing in each column
  - Step 3: Deciding what to do with the missing data:
    - drop it
    - fill it
    - let it be

Missing Data

- Notice, in the `deck` column, there are 3 instances of `NaN` we can see…
- But what about the other 800 or so rows? Do we have to go through and find them manually?

```
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton

Step 1: Detecting Missing Data

- In order to identify missing data, we will use a combination of three Pandas functions:

  - isna() https://pandas.pydata.org/docs/reference/api/pandas.isna.html
  - notna() https://pandas.pydata.org/docs/reference/api/pandas.notna.html
  - any() https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.any.html

Step 1: Detecting Missing Data

- Using `isna()` and `notna()` to find missing data:
    - isna(): will return a boolean series where it is `True` if the element is `NaN`
    - notna(): will return a boolean series where it is `True` if the element is **`not NaN`**

Input/output

General functions

pandas.melt

pandas.pivot

pandas.pivot_table

pandas.crosstab

pandas.cut

pandas.qcut

pandas.merge

pandas.merge_ordered

API reference > General functions > pandas.isna

pandas.isna

pandas.isna(*obj*) [source]

Detect missing values for an array-like object.

This function takes a scalar or array-like object and indicates whether values are missing (`NaN` in numeric arrays, `None` or `NaN` in object arrays, `NaT` in datetimelike).

Parameters:

obj : ***scalar or array-like***

Object to check for null or missing values.

https://pandas.pydata.org/docs/reference/api/pandas.isna.html

Step 1: Detecting Missing Data

```python
titanic.loc[0:4]
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton

- Now, let's call `isna()`, and see what we get as an output

```python
titanic.loc[:4].isna()
#look at the 'deck' column...
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	False	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True	False

Step 1: Detecting Missing Data

- `isna()` is pretty nifty but there should be better way to summarize this.
    - any()

pandas.DataFrame.any

`DataFrame.any(*, axis=0, bool_only=False, skipna=True, **kwargs)` [source]

Return whether any element is True, potentially over an axis.

Returns False unless there is at least one element within a series or along a Dataframe axis

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.any.html

Step 1: Identifying Missing Data

- Let's use `any()` on the call to `isna()` we just did to let us know which columns have missing data:

```
titanic.isna().any()

survived       False
pclass         False
sex            False
age             True
sibsp          False
parch          False
fare           False
embarked        True
class          False
who            False
adult_male     False
deck            True
embark_town     True
alive          False
alone          False
dtype: bool
```

- Several columns are missing data: `age`, `embarked`, `deck`, and `embark_town`.
- Wouldn't it be great to know how much data is missing in each of those columns?

Missing Data

3Ds
Detect
Determine
Decide

- The steps of cleaning data normally include:
  - Step 1: Detecting which columns have missing data
  - Step 2: Determining how much data is missing in each column
  - Step 3: Deciding what to do with the missing data:
    - drop it
    - fill it
    - let it be

Step 2: How much data is missing?

- It's important to determine *how much missing data each column has* before we decide how to handle our missing data:

    - If the missing data is a small proportion of the data, we choose to drop those rows completely from the dataset

    - However, if most of the rows are missing data for a specific column, maybe it's a sign that we don't need to use that column

- There are multiple ways of determining the amount of missing data, but one of the quickest/easiest is using **`value_counts()`**

Step 2: How much data is missing?

- Great, so now that we know which columns are missing data, let's check to see how much data they are missing using `value_counts()`

pandas.Series.value_counts

`Series.value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)` [source]

Return a Series containing counts of unique values.

The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Step 2: How much data is missing?

- Let's apply **`value_counts()`** on the various columns (eg, `deck`) of Titanic dataset

```
titanic.deck.value_counts(dropna=False)
#688 missing values
```

```
NaN    688
C       59
B       47
D       33
E       32
A       15
F       13
G        4
Name: deck, dtype: int64
```

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Step 2: How much data is missing?

- Let's apply **`value_counts()`** on the various columns (eg, `age`) of Titanic dataset

```
titanic.age.value_counts(dropna=False)
#177 missing values
```

```
NaN      177
24.00     30
22.00     27
18.00     26
28.00     25
        ...
36.50      1
55.50      1
0.92       1
23.50      1
74.00      1
Name: age, Length: 89, dtype: int64
```

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Step 2: How much data is missing?

- Let's apply **`value_counts()`** on the various columns (eg, `embarked`) of Titanic dataset

```
titanic.embarked.value_counts(dropna=False)
#2 missing values
```

```
S      644
C      168
Q       77
NaN      2
Name: embarked, dtype: int64
```

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Step 2: How much data is missing?

- Let's apply **`value_counts()`** on the various columns (eg, `embark_town`) of Titanic dataset

```
titanic.embark_town.value_counts(dropna=False)
#2 missing values

Southampton    644
Cherbourg      168
Queenstown      77
NaN              2
Name: embark_town, dtype: int64
```

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Step 2: How much data is missing?

- So, here is our results using **`value_counts()`**

Column	Num Rows Missing
deck	688
age	177
embarked	2
embark_town	2

- Now with this new information, it's up to us to decide what to do with these missing values

https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

Missing Data

3Ds
Detect
Determine
Decide

- The steps of cleaning data normally include:
  - Step 1: Detecting which columns have missing data
  - Step 2: Determining how much data is missing in each column
  - Step 3: Deciding what to do with the missing data:
    - **drop it:** drop the missing data from the dataset (either col or row)
    - **fill it:** fill the missing data with a suitable replacement
    - **let it be:** let it be and cross our fingers

Option 1: Drop it using `dropna()`

- If there isn't much missing data, and/or you have a very large dataset, dropping the row that includes the missing data is a viable option.

```
print("before: ", titanic.shape)
titanic.dropna()
print("after: " , titanic.shape)

before:  (891, 15)
after:  (891, 15)
```

- We know that there's missing data, why didn't the shape change?

Option 1: Drop it using `dropna()`

- Pandas is trying to protect you, and rather than dropping the rows "in place", it is returning a `DataFrame` with the rows dropped--as written, we're just not saving it's return. There are two ways to fix this:
  - save what `dropna()` is returning in a variable

```
print("before dropna(): ", titanic.shape)
no_missing_data = titanic.dropna()
print("after dropna(): ", no_missing_data.shape)

before dropna():  (891, 15)
after dropna():  (182, 15)
```

  - add the parameter `inplace=True` to the function call, and it will drop the rows in the original dataset (be careful with this one)

```
print("before dropna(): ", titanic.shape)
titanic.dropna(inplace=True)
print("after dropna(inplace=True): ", titanic.shape)

before dropna():  (891, 15)
after dropna(inplace=True):  (182, 15)
```

Option 1: Drop it using `dropna()`

- That's better, but wow, most of our dataset is gone now if we drop all of the rows that have missing data. If this happens to you, you'll probably want to re-load your data to have the full dataset to work with.

```
path = '/content/drive/MyDrive/cs167_fall24/datasets/vehicles.csv'
vehicles = pd.read_csv(path)
vehicles.head()
```

Option 2: Fill it using `fillna( )`

- If dropping all of the data will make your dataset too sparse, consider filling the missing values with something else.

- What do you think we should use to fill in the missing data in the `age` column?
  - we probably don't want to throw off our statistics...

```
titanic.head(7)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True

Option 2: Fill it using `fillna()`

- What do you think we should use to fill in the missing data in the `age` column?
  - we probably don't want to throw off our statistics...

```
print("before: ", titanic['age'].isna().any())
age_mean = titanic['age'].mean()
titanic['age'].fillna(age_mean, inplace=True)
print("after: ", titanic['age'].isna().any())
titanic.head(7)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.000000	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.000000	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.000000	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.000000	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.000000	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
5	0	3	male	29.699118	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True
6	0	1	male	54.000000	0	0	51.8625	S	First	man	True	E	Southampton	no	True

Option 3: Let it be

- What's so bad about missing data? Why do we care if some data is missing?
- What happens if we try to do math with `NaN`? Try it out for yourself:
    - Go to the bottom of the [Day04_Handling_Missing_Data.ipynb]() and try out

Summary: Missing Data

- The steps of cleaning data normally include:
  - Step 1: Detecting which columns have missing data
  - Step 2: Determining how much data is missing in each column
  - Step 3: Deciding what to do with the missing data:
    - drop it
    - fill it
    - let it be

Summary: Missing Data Functions

- `isna()`: returns True for any missing data with `NaN`
- `notna()`: returns True for any data that is **not** `NaN`
- `any()`: returns true if any of the elements in a Series is True
- `value_counts()`: returns a list of the values in a Series, use `dropna=False` to see `NaN` values
- `dropna()`: drops rows or columns (specify which axis, 1 or 0) that have missing data. Don't forget to either save the result of the call or add `inplace=True` as a parameter
- `fillna(new_val, inplace=True)`: replaces missing data with a given value (generally 0 or the mean)

Quick Overview

- Quick simple statistics review:
  - **mean:** the average, take the sum of elements, and divide by the number of total elements
  - **median**: The middle number; found by ordering all data points and picking out the one in the middle (or if there are two numbers in the middle, taking the mean of those two numbers)
  - **mode**: The most frequent number; the number that occurs the highest number of times

Poll

- Participate in the following poll:
  - https://forms.gle/Xq1Mx9wrFY3aUCuz6

Machine Learning Variations

- We are going to learn about a lot of different types of machine learning in CS167. Here are a few categories to look out for:
  - **classification:** identify which category it goes in. Examples: Spam or ham? Reza or Eric? Fish, amphibian, reptile, bird, or mammal
  - **regression**: real-valued labels. Examples: price of Bitcoin, tomorrow's temperature, etc
  - **supervised learning:** data has labels, goal is to predict the labels of new instance
  - **unsupervised learning:** data does not have a label, the goal is to analyze/cluster the examples
  - **other issues:** missing data, sequential data, outlier anomaly detetion, and many more

Terminology Alert!

- Load the "Iris" Dataset on the Colab notebook for this lecture
- Let's take a look at a couple rows of the "Iris" Dataset:

```
iris.head(2)
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

Terminology Alert!

- Each row in the table represents a **training example**, a previously-seen, known instance of the thing we are trying to model
- Each column in the table represents a **feature**, some attribute or variable that each training example has a value for
- **Target variable:** the 'feature' we will try to predict (`species` in this case) — it's value is unknown for any new cases not in the training data
- **Predictor variables:** (or just predictors), the features that will be used to make predictions of the target variable e.g., `sepal length`, `petal length`, `sepal width`, `petal width`

```
iris.head(2)
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

Predictor Predictor Predictor Predictor Target

Terminology Alert!

- Remember this question from Day01?

Imagine you found this beautiful flower while on a walk and took the following measurements:

5.1 cm petal length
7.2 cm sepal length

What species do you think it is?