# Missing Data

Datasets can often have missing data, which can cause problems when we perform machine learning. Missing data can be hard to spot at a first glance.

In our scenario, we obtained a list of passengers on the failed maiden voyage of the Titanic. We'd like to know which factors predicted passenger survival. For our first task, which we'll perform here, we'll check whether our dataset has missing information.

## The required 'graphing' library package

This tutorial relies on the 'graphing' library package resource. Depending on the environment you use to execute the code in this notebook, you might need to install that package to proceed. To install the 'graphing' package, uncomment and execute this notebook cell code:

```
# %pip install graphing
```

## Preparing data

Let's use Pandas to load the dataset and take a cursory look at it:

```
import pandas as pd
!pip install missingno

# Load data from our dataset file into a pandas dataframe
!wget https://raw.githubusercontent.com/MicrosoftDocs/mslearn-introduction-to-machine-learning
!wget https://raw.githubusercontent.com/MicrosoftDocs/mslearn-introduction-to-machine-learning
dataset = pd.read_csv('titanic.csv', index_col=False, sep=",", header=0)
```

```
# Let's take a look at the data
dataset.head()
```

Now, we'll see how many samples and columns we have:

```
# Shape tells us how many rows and columns we have
print(dataset.shape)
```

We have data for 891 passengers, each described by 12 different variables.

# Finding Missing Data

Do we have a complete dataset?

No. We know from history that there were more than 2000 people on the Titanic, so we know straight away that we are missing information on more than 1000 people!

How can we tell if the data we have available is complete?

We could print the entire dataset, but this could involve human error, and it would become impractical with this many samples.

A better option would use `pandas` to report the columns that have "empty" cells:

```
# Calculate the number of empty cells in each column
# The following line consists of three commands. Try
# to think about how they work together to calculate
# the number of missing entries per column
```

```
# the number of missing entries per column
missing_data = dataset.isnull().sum().to_frame()

# Rename column holding the sums
missing_data = missing_data.rename(columns={0:'Empty Cells'})

# Print the results
print(missing_data)
```

It looks like we don't know the age of 177 passengers, and we don't know if two of them even embarked.

Cabin information for a whopping 687 persons is also missing.

# Missing Data Visualizations

Sometimes it can help if we can see if the missing data form some kind of pattern.

We can plot the absence of data in a few ways. One of the most helpful is to literally plot gaps in the dataset:

```
# import missingno package
import missingno as msno

# Plot a matrix chart, set chart and font size
msno.matrix(dataset, figsize=(10,5), fontsize=11)
```

The white bars in the graph show missing data. Here, the patterns aren't visually clear, but maybe many passengers with missing `Age` information are also missing `Cabin` information.

# Identifying Individual Passengers with Missing Information.

Let's use pandas to get a list of passengers of unknown age:

```
# Select Passengers with unknown age
# Notice how we use .isnull() rows with no value
unknown_age = dataset[dataset["Age"].isnull()]

# Print only the columns we want for the moment (to better fit the screen)
# limit output to 20 rows
unknown_age[["PassengerId","Name", "Survived", "Age"]][:20]
```

This technique lists the passengers with missing `Cabin` or `Embarked` information as well. Let's combine these using an `AND`, to see how many passengers are missing both Cabin and Age information

```
# Find those passengers with missing age or cabin information
missing_age = dataset["Age"].isnull()
missing_cabin = dataset["Cabin"].isnull()

# Find those passengers missing both
unknown_age_and_cabin = dataset[missing_age & missing_cabin]
print("Number of passengers missing age and cabin information:", len(unknown_age_and_cabin))
```

Our suspicions were correct - most passengers missing age information are also missing cabin information.

Normally, from here, we would want to know *why* we have this issue. A good hypothesis is that information was not collected carefully enough for the passengers who used the cheap tickets.

Let's plot a histogram of ticket classes, and another of just people missing information.

```
import graphing

# The 'graphing' library is custom code we use to make graphs
# quickly. If you don't run this notebook in the sandbox
# environment, you might need to formally install this library
# in the environment you use. See the first cell of this notebook
# for more information about installation of the 'graphing'
# library.
#
# To review the 'graphing' library in detail, find it in our
# GitHub repository

graphing.histogram(dataset, 'Pclass', title='Ticket Class (All Passengers)', show=True)
graphing.histogram(unknown_age_and_cabin, 'Pclass', title='Ticket Class (Passengers Missing Cal
```

It seems that those passengers with missing information typically used the cheaper tickets. These sorts of biases might cause problems in real-world analyses.

# Missing as Zero

Additionally, some datasets may have missing values that appear as zero. While the Titanic dataset doesn't have this problem, let's see how that would work here.

```
import numpy as np

# Print out the average age of passengers for whom we have age data
mean_age = np.mean(dataset.Age)
print("The average age on the ship was", mean_age, "years old")

# Now, make another model where missing ages contained a '0'
dataset['Age_2'] = dataset['Age'].fillna(0)
mean_age = np.mean(dataset.Age_2)
print("The average age on the ship was", mean_age, "years old")
```

```
print( "The average age on the ship was ", mean_age, " years old")
```

What happened here? Our analyses have considered the values of `0` to not be 'missing' but rather to be actual ages.

This shows that it can be important to time the review of your raw data before you run the analyses. Another fast way to get a feel for a dataset is to graph its distribution:

```
graphing.histogram(dataset, label_x="Age_2")
```

Here, we see an unlikely number of very young children. This would be cause for further inspection of the data, to hopefully spot the fact that the missing ages appear as zeros.

# Handling Missing Data

There are many ways to address missing data, each with pros and cons.

Let's take a look at the less complex options:

# Option 1: Delete data with missing rows

When we have a model that cannot handle missing data, the most prudent thing to do is to remove rows that have information missing.

Let's remove some data from the `Embarked` column, which only has two rows with missing data.

```
# Create a "clean" dataset, where we cumulatively fix missing values
# Start by removing rows ONLY where "Embarked" has no values
print(f"The original size of our dataset was", dataset.shape)
clean_dataset = dataset.dropna(subset=["Embarked"])
```

```
clean_dataset = clean_dataset.reindex()

# How many rows do we have now?
print("The shape for the clean dataset is", clean_dataset.shape)
```

We can see that this removed the offending two rows from our new, clean dataset.

## Option 2: Replace empty values with the mean or median for that data.

Sometimes, our model cannot handle missing values, and we also cannot afford to remove too much data. In this case, we can sometimes fill in missing data with an average calculated on the basis of the rest of the dataset. Note that imputing data like this can affect model performance in a negative way. Usually, it's better to simply remove missing data, or to use a model designed to handle missing values.

Below, we impute data for the `Age` field. We use the mean `Age` from the remaining rows, given that >80% of these aren't empty:

```
# Calculate the mean value for the Age column
mean_age = clean_dataset["Age"].mean()

print("The mean age is", mean_age)

# Replace empty values in "Age" with the mean calculated above
clean_dataset["Age"].fillna(mean_age, inplace=True)

# Let's see what the clean dataset looks like now
print(clean_dataset.isnull().sum().to_frame().rename(columns={0:'Empty Cells'}))
```

The `Age` field has no longer has empty cells anymore.

```
# Assign unknown to records where "Cabin" is empty
clean_dataset["Cabin"].fillna("Unknown", inplace=True)

# Let's see what the clean dataset looks like now
print(clean_dataset.isnull().sum().to_frame().rename(columns={0:'Empty Cells'}))
```

That's it! No more missing data!

We only lost two records (where `Embarked` was empty).

That said, we had to make some approximations to fill the missing gaps for the `Age` and `Cabin` columns, and those will certainly influence the performance of any model we train on this data.

# Summary

Missing values can affect the way a Machine Learning model works in a negative way. It's important to quickly verify the existence of data gaps, and the locations of those gaps.

You can now get a "big picture" of what is missing, and select only those items that you must address, by the use of lists and charts.

In this exercise, we practiced:

- Finding and visualization of missing dataset values, using the `pandas` and `missingno` packages.
- Checking whether a dataset uses the value '0' to represent missing values.

- Handling missing data in three ways: removing of rows that contain missing values, replacment of the missing values with the mean or median of that particular feature, and creation of a new `Unknown` category, if dealing with categorical data.



## Next unit: Examine different types of data

Continue >