# Exploring and Preprocessing the Titanic Dataset

In this notebook, I will use the Titanic dataset to explore and preprocess data. In particular, I will...

- Handle missing data
- remove duplicates,
- Label encoding and pre-processing data

For this project, I got the dataset here: https://www.kaggle.com/competitions/titanic/overview

In [61]:
```
# First, I will import all necessary libraries.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

In [62]:
```
# Importing the dataset
df = pd.read_csv("train.csv")
df.head()
```

Out[62]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 |

To clarify some points, the columns are...

- Passenger ID
- Survived: 0 = no, 1 = yes
- Pclass: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- Name
- Sex
- Age
- SibSp: number of siblings that person has on board
- Parch: number of parents that person has on board
- Ticket
- Fare
- Cabin
- Embarked: Where they left from C = Cherbourg, Q = Queenstown, S = Southampton

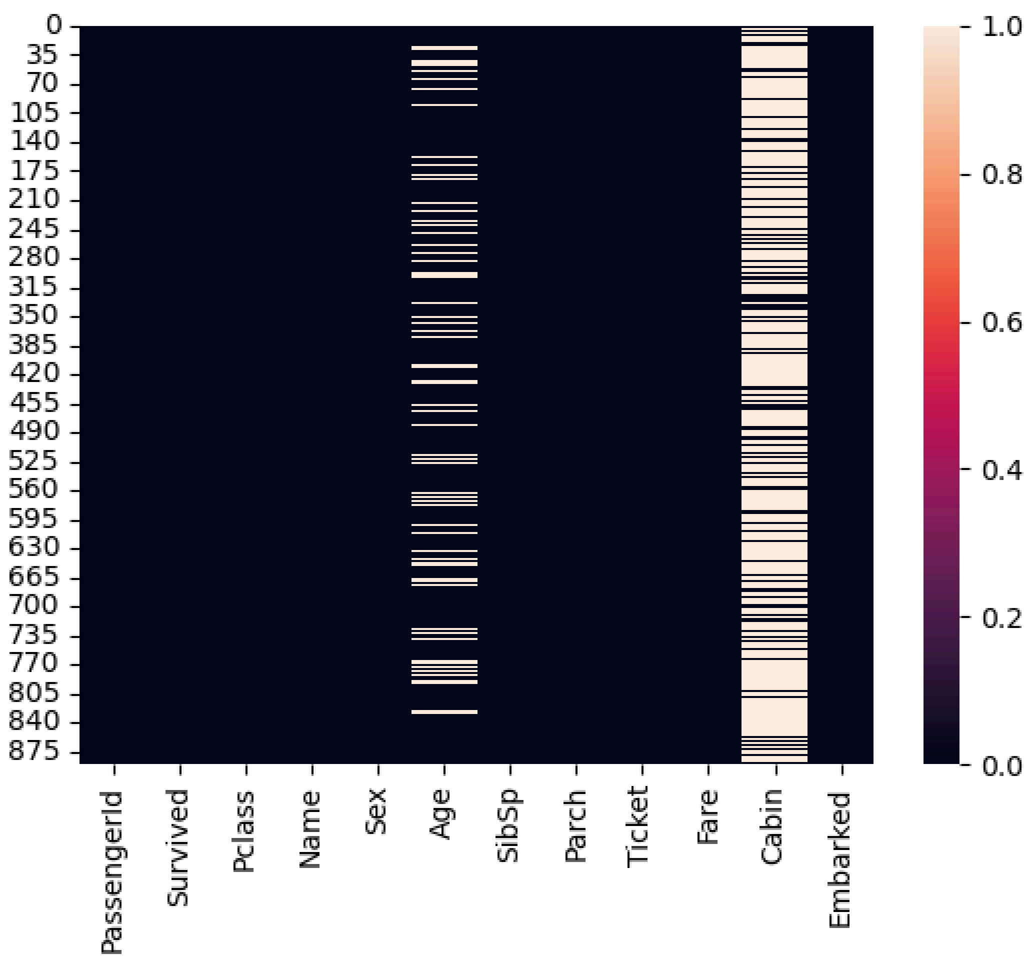
In [63]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [64]:
```
sns.heatmap(df.isnull())
```

Out[64]: <Axes: >




I included this heatmap, generated using the seaborn library, to highlight where values are missing and will utilize it for handling missing data. As we can see from the map, we will have to handle missing values from the "Age" and "Cabin" columns. It's hard to see on the map, but we are also missing information for two people in the embarked column. I will first start with the Cabin column.

In [65]:
```
Cabin_nan_count = df['Cabin'].isna().sum()
print(Cabin_nan_count / 891)
```

```
0.7710437710437711
```

In [66]:
```
#77% of the Cabin column is missing, and therefore, I will drop it because there is
df.drop(columns=['Cabin'], inplace=True)
```

In [67]:
```
#Next, to fill in the missing values for the "Age" column, I will find the average
df.describe()
```

Out[67]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.00000( |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.20420ε |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.69342ς |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.00000( |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.91040( |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.45420( |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.00000( |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.32920( |

We can see that the average age is 29.7 years

In [68]:
```
df["Age"] = df["Age"].fillna(29.7)
```

Finally, since we are only missing two entries for the embarked column, I will simply get rid of those two.

In [69]:
```
df.dropna(subset=["Embarked"], inplace=True)
```

In [70]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 889 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  889 non-null    int64  
 1   Survived     889 non-null    int64  
 2   Pclass       889 non-null    int64  
 3   Name         889 non-null    object 
 4   Sex          889 non-null    object 
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64  
 7   Parch        889 non-null    int64  
 8   Ticket       889 non-null    object 
 9   Fare         889 non-null    float64
 10  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 83.3+ KB
```

Now we don't have anymore missing values. For the next step I will check for duplicates and remove if necessary

In [71]:
```
# We have zero duplicates, so no need to get rid of anything.
df.duplicated().sum()
```

Out[71]:
```
np.int64(0)
```

Now we will change the "sex" column from string entries (male and female) to number entries (male ---> 0 and female ---> 1). We want to do this because machine learning models such as logistic regression and decision trees expect numeric data types instead of strings.

In [72]:
```
# LabelEncoder() is a tool to convert a string to a numeric
le = LabelEncoder()
#.fit_transform() changes the column values into numeric types. In this case, that
df['Sex'] = le.fit_transform(df["Sex"])
```

Finally, we also need to change the Embarked column from an object to a numeric type. The Embarked column has three entries (C = Cherbourg, Q = Queenstown, S = Southampton).

In [75]:
```
# The .get_dummies() function converts text categories of a column into numbers. It
df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

In [77]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 889 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  889 non-null    int64  
 1   Survived     889 non-null    int64  
 2   Pclass       889 non-null    int64  
 3   Name         889 non-null    object 
 4   Sex          889 non-null    int64  
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64  
 7   Parch        889 non-null    int64  
 8   Ticket       889 non-null    object 
 9   Fare         889 non-null    float64
 10  Embarked_Q   889 non-null    bool   
 11  Embarked_S   889 non-null    bool   
dtypes: bool(2), float64(2), int64(6), object(2)
memory usage: 78.1+ KB
```

It created two new columns of the bool datatype. Notice that we are missing Embarked_C. This is because it is the default value, meaning that if Embarked_Q and S are false, then Embarked_C is true. Remember that in this case, false = 0 and true = 1.

| Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |

Finally, after removing/adding missing data, checking and removing duplicates, and pre-processing the data, the dataset is ready for machine learning models. Notice that the Ticket and Name columns are still objects and not numeric. This is because either they can't become numeric types, or because that information is not needed in the machine learning models.