Data Exploration

To work with a dataset is important to know the structure of the data and getting some statics that helps with identifying potential issues.

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	890.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.227883
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.716342
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.903100
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

The describe() method generates statistics for numeral columns in our dataset, this helps with getting some basic information quickly before working with the data, this includes statistics such as count, mean, standard deviation, minimum, Q1, Q2(median), Q3 and maximum for each column.

```
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         890 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
```

The info() gives us a bit of metadata for the different columns, it tells us which datatypes are used as well as how many rows contain non null values. This gives an insight into which columns we might need to process for the machine to understand.

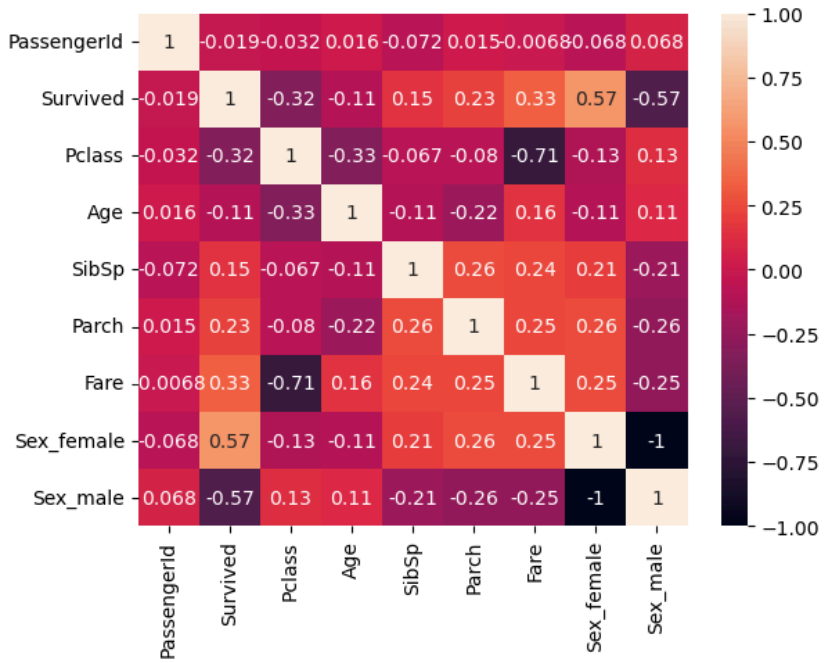
Data cleaning

Interpolation

The titanic dataset includes in total 891 rows, where most of the columns are filled. Through the df.isnull().sum() method, we can see which rows do not contain data.

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          687
Embarked         2
```

For the numerical values that are null (Fare, Age) we can interpolate a value. We have done so in two different ways.
For the age column we have chosen to use the mean of all the age values. The reason behind this choice is that we can see through the use of a heatmap that the age value does not strongly correlate with any other column:

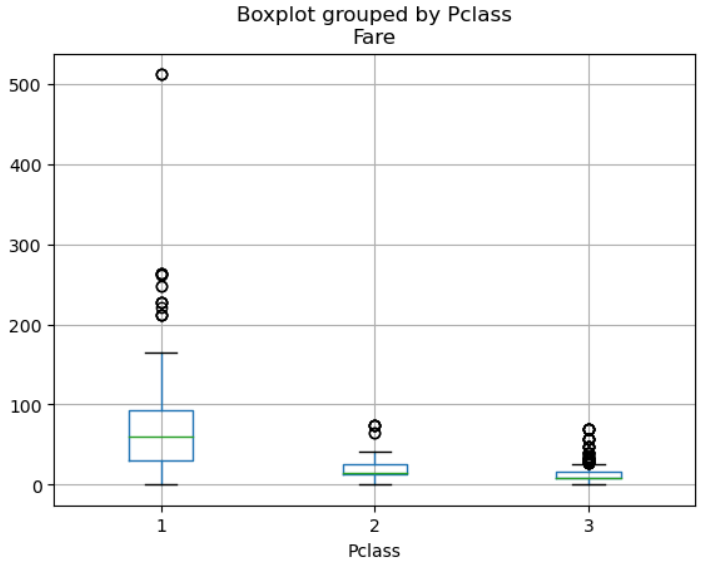


For the single missing fare value we have chosen to go with the mean of the fare grouped by class. This gives us a better value for the row than just choosing the mean of all fare values, as the mean is quite different in each of the classes.

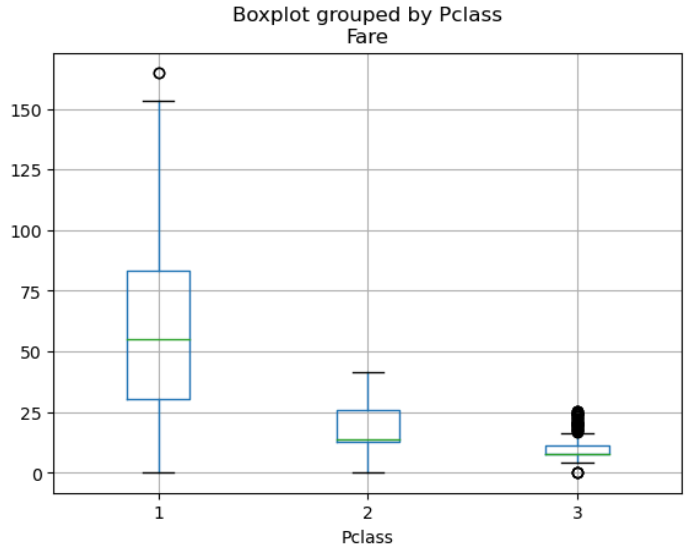
```
Mean fare first class 64.24102857142856
Mean fare second class 18.66859717514124
Mean fare third class 10.297031750267317
```

Outliers

The standard deviation of the fare column is 49, the min is 0, the max is 512 and the mean is 32. This all points to the fact that there are outliers in this column, as the spread is quite high. As a high value in third class might be the mean value of the second class, we have again chosen to group fare by class. Through the use of boxplots we can then visualize the outliers in the different classes



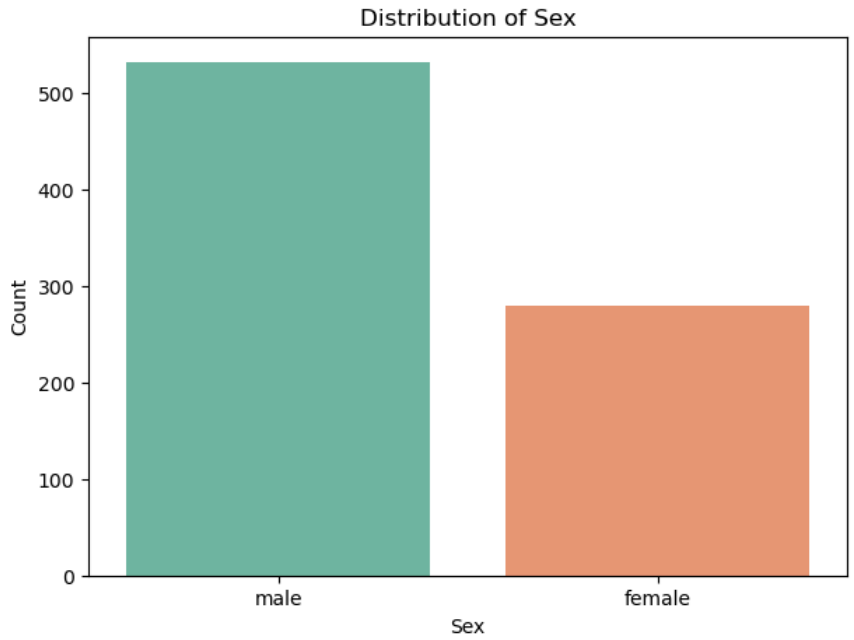
Once we removed the outliers by class, using the IQR method, our boxplot looked like this:



While we still have outliers in two out of three classes, we have removed the extreme outliers.

Data visualization

The visualizations helped to provide a better understanding of various aspects of the Titanic dataset, including the distribution of categorical variables such as sex, class, and survival status, as well as the distribution of numerical variables such as age and fare.



The distribution of sex among the passengers can be seen in Figure X, which illustrates that there were almost twice as many male passengers as female passengers.

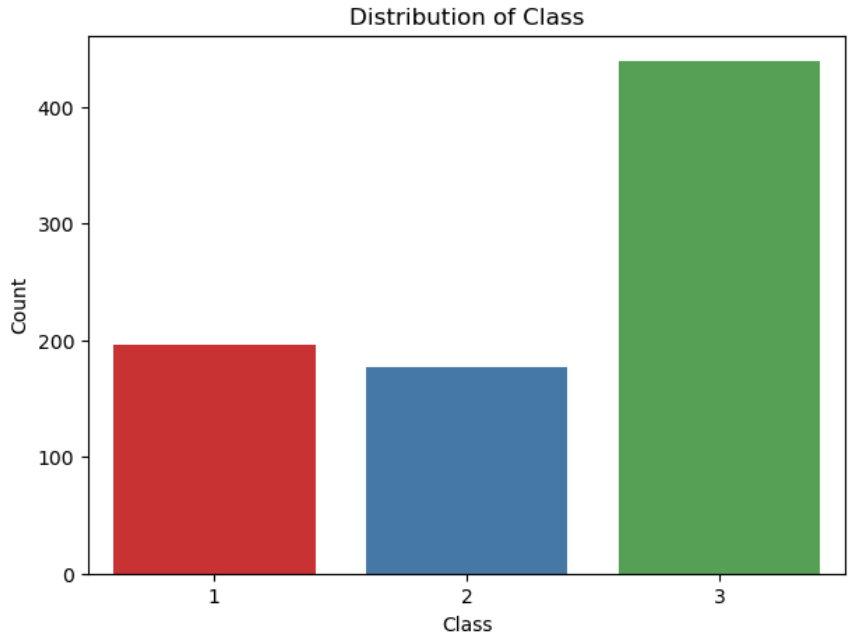
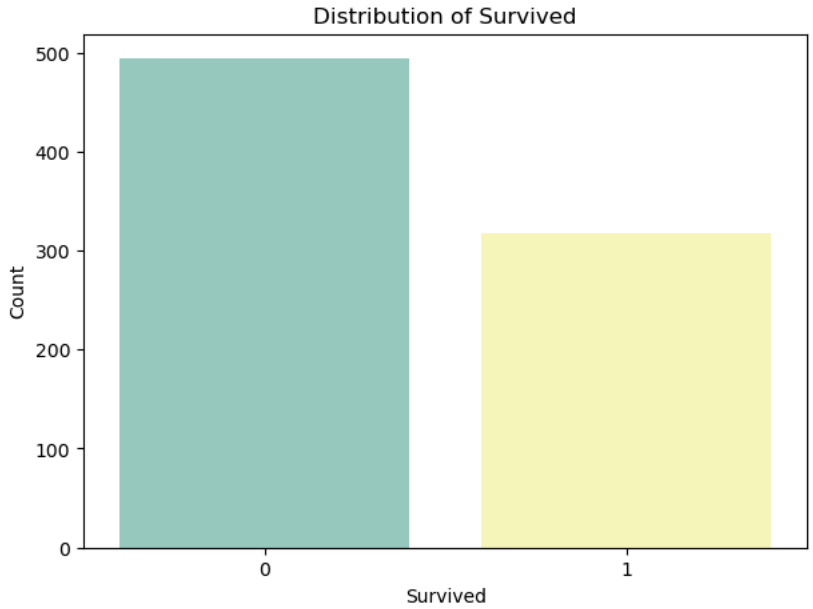


Figure x illustrates the distribution of passenger classes ('Pclass'). It shows that the majority of passengers belonged to class 3, followed by class 1 and then class 2, this gives an insight of the passengers economy.



The distribution of survival status among the passengers is shown in Figure x. It shows that a larger number of passengers did not survive the Titanic disaster compared to those who survived.

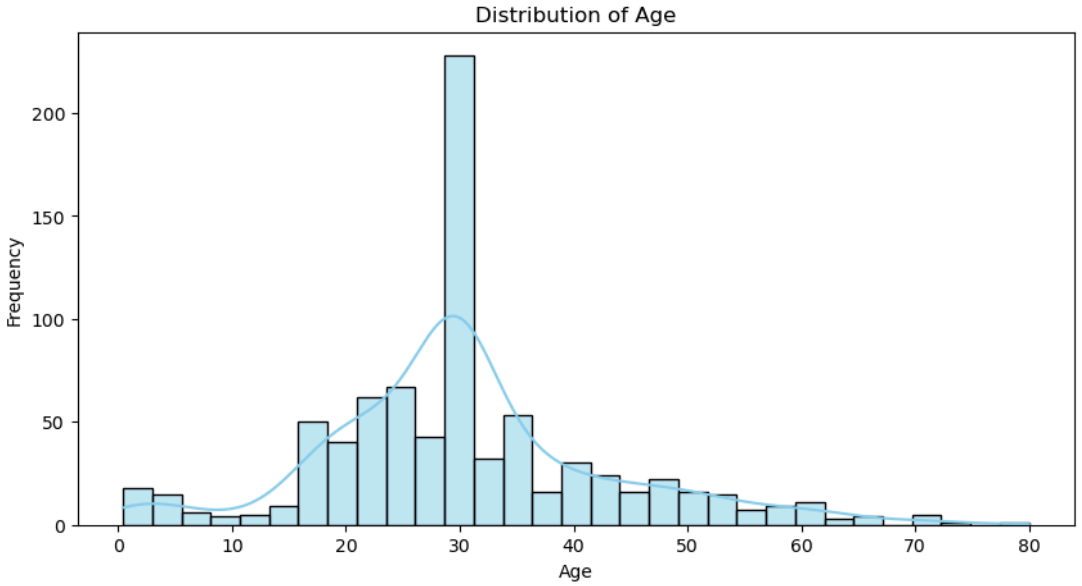


Figure X illustrates the distribution of the ages of the passengers on board and the frequency of each age, providing insight into the primary age groups on the Titanic, we can see that most of the passengers are around 30 years old.

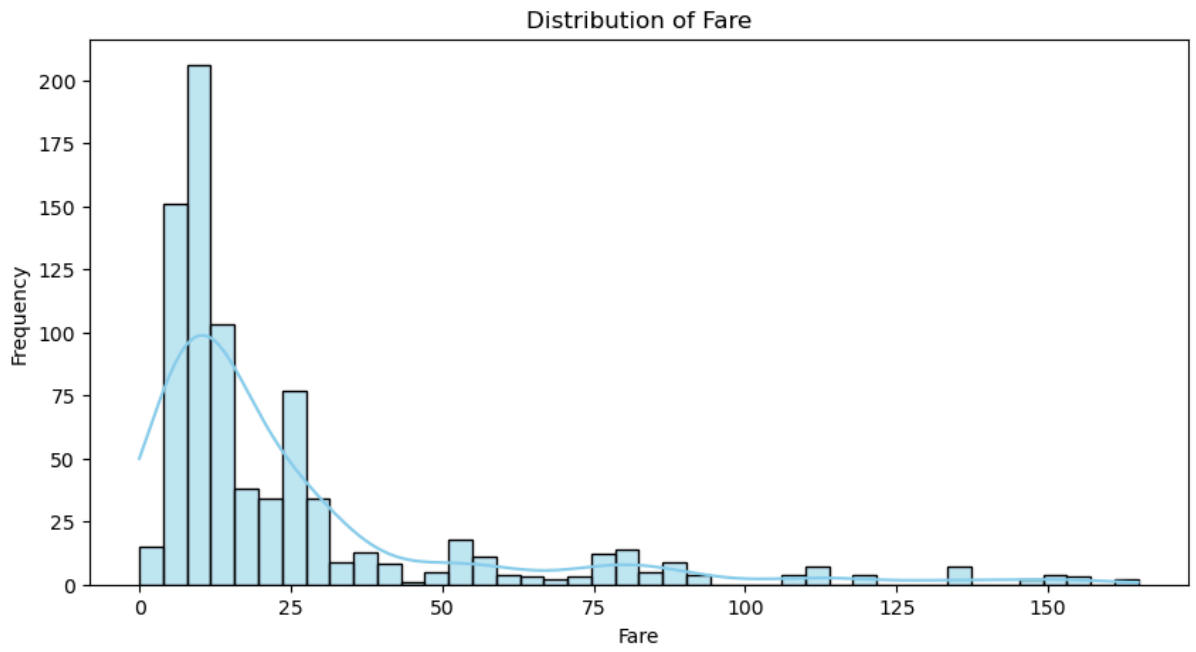


Figure x illustrates the frequency of the fare prices that the passengers have paid to board the Titanic, this shows that most of the passengers paid around 13 dollars.

Feature Engineering

For a computer to understand the dataset, the values must be numerical, this is where feature engineering comes in handy. We have both implemented one hot encoding for the sex column, which gives us two columns sex_male and sex_female, containing either 0 or 1.
We have also used binning to turn the age column into three categories: young, adult and senior.

Data Selection and Wrangling

Our hypothesis was that the women and young people had a greater chance of survival. as “women and children first” is heard when talking about ships sinking. We also had a hypothesis that first class passengers had a higher chance of survival, simply because they paid more money.

Through the use of groupby() we can see that all of this seems to be true.

```
[27]: df.groupby(['AgeCategory'])['Survived'].mean()
```

```
[27]: AgeCategory
      young     0.607477
      adult     0.362573
      senior    0.238095
      Name: Survived, dtype: float64
```

```
[28]: df.groupby(['Sex_female'])['Survived'].mean()
```

```
[28]: Sex_female
      0    0.18985
      1    0.77500
      Name: Survived, dtype: float64
```

```
[29]: df.groupby(['Pclass'])['Survived'].mean()
```

```
[29]: Pclass
      1    0.622449
      2    0.480226
      3    0.252847
      Name: Survived, dtype: float64
```