BIRZEIT UNIVERSITY

Department of Electrical & Computer Engineering
ENCS5141 - Intelligent Systems Laboratory

Case Study #1—Data Cleaning and Feature Engineering for the Titanic Dataset

Prepared by: Mazen Batrawi
Instructor: Dr. Ismail Khater
Assistant: Eng. Hanan Awawdeh
Date: March 31, 2024

Abstract

This case study aims to prepare the Titanic dataset for machine learning analysis by addressing data cleaning and transformation tasks. The objective is to enhance the quality of the dataset and optimize its suitability for predictive modeling. The study employs various data preprocessing techniques, including handling missing values, selecting pertinent features, encoding categorical data, partitioning the dataset into training and testing sets, and applying scaling and dimensionality reduction methods. These steps are systematically executed to ensure the dataset is properly cleaned and transformed for subsequent machine learning tasks. By comparing a model trained on preprocessed data with one trained on raw data, it is observed that upon the application of scaling and dimensionality reduction techniques, there was a slight increase in accuracy from 77.6% to 78.2%, which indicates under-fitting. This accuracy is a due to the fact that the dataset does not contain much of numerical values and relations between the features, but the steps of cleaning the data still remain very important. These findings underscore the importance of carefully considering the balance between data transformation techniques and model performance when preparing datasets for predictive modeling tasks.

Contents

List of Figures

1 Introduction

In this case study, we'll undertake crucial data preprocessing tasks on the Titanic dataset, which comprises information regarding the passengers aboard the Titanic ship. Our objective is to ready the dataset for machine learning analysis. This case study encapsulates the data science component of the lab, where we'll visualize the dataset, furnish descriptive statistics, address missing data and outliers, choose relevant features, transform the data, and manage high-dimensional data.

1.1 Dataset

The Titanic dataset has emerged as a prominent resource in the domain of data science, garnering attention as a widely-used dataset for analysis and machine learning endeavors. This dataset encapsulates valuable insights into the passengers aboard the ill-fated Titanic ship and has served as a cornerstone for educational and research initiatives alike.

Comprising data from the tragic maritime event, the Titanic dataset provides a comprehensive glimpse into the demographics and characteristics of passengers onboard. From socio-economic status to cabin allocation, this dataset offers a multifaceted view of the individuals who embarked on the voyage. The dataset has been extensively utilized to unravel patterns of survival and explore factors influencing outcomes amidst the catastrophic event.

Reflecting the diversity among the passengers, the Titanic dataset encompasses a range of attributes, including age, gender, passenger class, embarkation port, and survival status. It serves as a rich repository for investigating various facets of human behavior and decision-making under dire circumstances. Through its detailed records and historical significance, the Titanic dataset continues to be a valuable asset for analytical exploration and machine learning endeavors in both educational and research contexts.

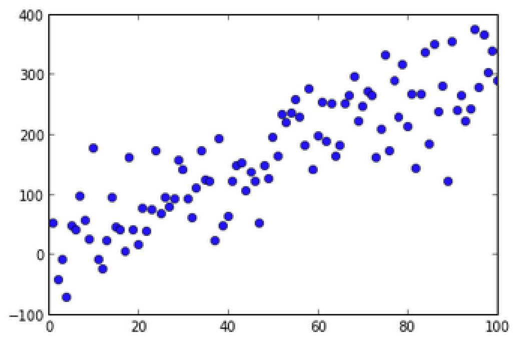
1.2 Data Visualization and Data Cleaning

This section illustrates various methodologies employed in data cleaning as an integral component of Exploratory Data Analysis (EDA). EDA serves as a pivotal step in understanding and scrutinizing datasets thoroughly.

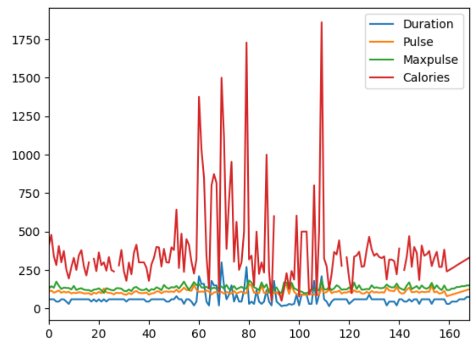
1.2.1 Data Visualization

Utilizing graphical elements like charts, graphs, and maps, data visualization provides a robust means of conveying insights derived from data. This visual portrayal simplifies complex datasets, making patterns, trends, and relationships easily discernible. By aiding in the detection of anomalies and facilitating clearer comprehension, data visualization plays a crucial role in enhancing decision-making processes and conveying research findings effectively.

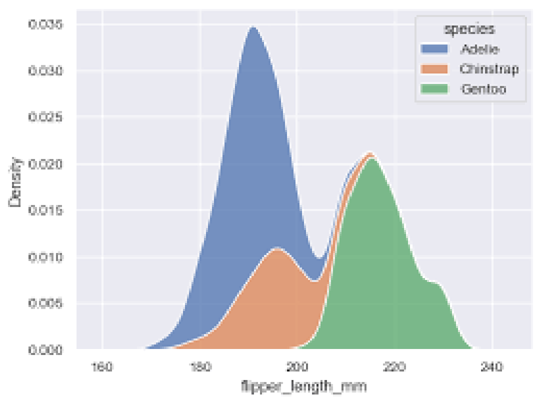
In artificial intelligence (AI), data visualization serves several key purposes. Firstly, it aids in exploring and understanding data, uncovering patterns and trends crucial for tasks like data mining and machine learning. Secondly, it communicates insights effectively to stakeholders, facilitating decision-making processes. Additionally, data visualization helps in generating hypotheses about the data, supporting research and problem-solving endeavors. Lastly, it contributes to validating AI models, ensuring their accuracy and reliability through visual inspection and comparison with actual data.



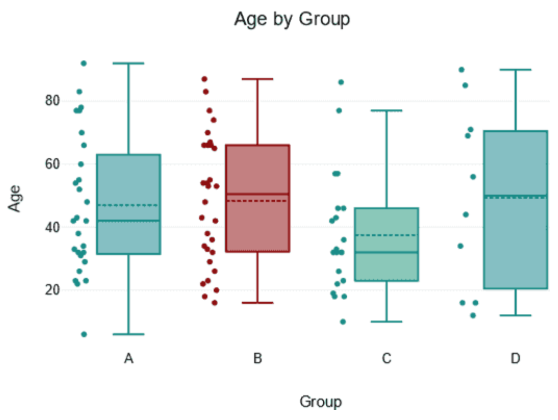
(a) Matplotlib



(b) Pandas



(c) Seaborn



(d) Boxplot

Figure 1.1: Plot examples using different libraries

1.2.2 Descriptive Statistics

In machine learning, descriptive statistics is the foundation for understanding a dataset's core characteristics. These techniques provide insights into central tendencies (typical values), variability (how data points spread), and overall distribution. This knowledge allows us to uncover patterns, identify anomalies, and diagnose potential issues that could hinder model performance. Furthermore, descriptive statistics play a critical role in data cleaning and preprocessing. By leveraging these methods, we can pinpoint and address challenges like missing values, outliers, and inconsistencies. This ensures the data fed into machine learning algorithms is reliable and well-suited for analysis.

- **Central tendency**: The following measures are employed to assess the central tendency of a distribution of data: Mean, Median, and Mode. The mean represents the average value of the data, the median corresponds to the middle value when the data is sorted, and the mode indicates the value that appears most frequently in the dataset. Pandas provides methods like `mean()`, `median()`, and `mode()` to calculate these measures.

- **Variation**: In assessing the spread of data distribution, various metrics are employed, including variance and standard deviation. Variance measures the extent to which data points diverge from the mean, while standard deviation, derived as the square root of the variance, signifies the dispersion of the data. Pandas, a Python library, offers convenient functions like **var()** and **std()** to calculate the variance and standard deviation of columns within a DataFrame. These functions facilitate efficient computation of these crucial statistical measures, aiding in the analysis and interpretation of data distributions.

- **Shape of distribution**: Skewness and kurtosis are statistical measures that provide insights into the shape of a distribution, where the skewness measures the asymmetry if the distribution of the data, and the kurtosis measures the peakdness of the distribution of the data.

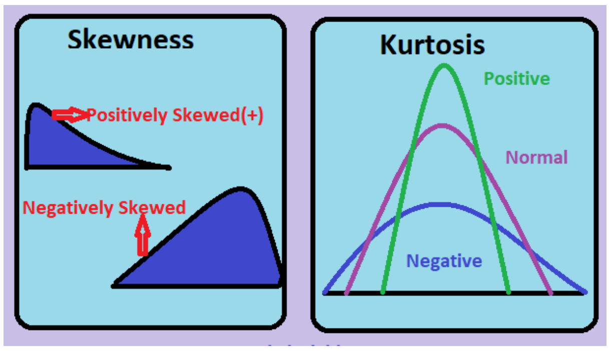


Figure 1.2: Demonstrating the cases of skewness and kurtosis

  - **Skewness** $\approx$ **0**: The distribution is approximately symmetric.

  - **Skewness** $<$ **0**: The tail is longer on the left side (left-skewed).

  - **Skewness** $>$ **0**: The tail is longer on the right side (right-skewed).

  - **Kurtosis** $\approx$ **3**: Tthe distribution has similar tails as a normal distribution.

  - **Kurtosis** $<$ **3**: A lighter-tailed distribution has fewer extreme values, meaning outliers are less common compared to distributions with heavier tails. This results in a flatter peak and a lower probability of observing values far from the mean.

  - **Kurtosis** $>$ **3**: A heavy-tailed distribution implies a greater likelihood of extreme values, as the tail decays more slowly than in a normal distribution, resulting in a sharper peak.

- **Quantiles**: Percentiles represent values where a given percentage of data lies, while the Interquartile Range (IQR) measures the spread between the 25th and 75th percentiles, capturing the central dispersion. Pandas' quantile() method calculates percentiles, such as quantile(0.75) for the 75th percentile and quantile(0.25) for the 25th percentile. Subtracting quantile(0.25) from quantile(0.75) yields the IQR. These tools are essential for assessing data distribution and variability.

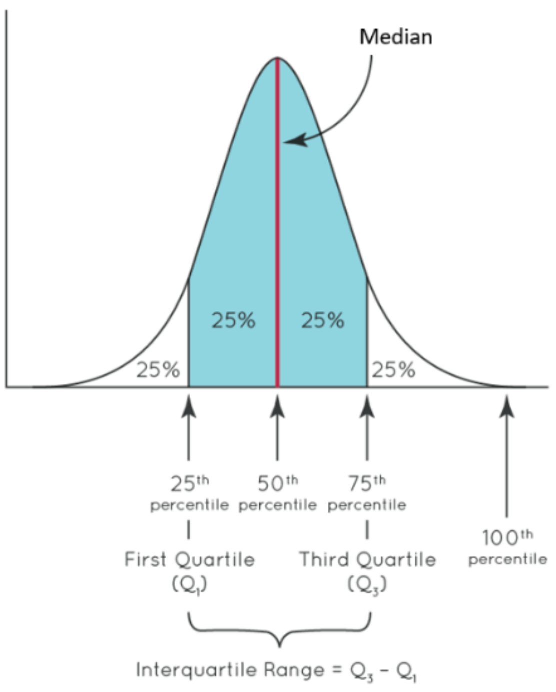


Figure 1.3: Demonstration of percentiles and IQR

1.2.3 Handling Missing Data

Missing data is data that is not available for one or more observations in a dataset. It can occur for a variety of reasons, such as human error, equipment malfunction, or deliberate omission.

Missing data can have a significant impact on the quality and accuracy of a dataset. If not handled properly, it can lead to biased results and inaccurate conclusions.

There are a number of methods for handling missing data. The best method to use will depend on the specific circumstances and the goals of the analysis. Some common methods include:

- **Dropping/Deletion**: This method is used if the number of missing values in a row is relatively small, and you believe these rows do not carry essential information for your analysis. Use the `dropna()` method in pandas to remove rows with missing values from your dataset.

- **Create a Separate Category**: If missing values represent a distinct category or carry specific meaning in your analysis, assign a special label or category to missing values. This makes them a distinct class in your analysis.

- **Imputation**: The imputation of missing values is used if the proportion of missing values is significant, and you believe these data points are valuable for your analysis. You can impute missing values using various techniques:

  - **Mean, Median, or Mode Imputation**: Replace missing values with the mean, median, or mode of the observed values in the respective column.
  - **Regression Imputation**: Train a regression model to predict missing values based on other features in the dataset.
  - **K-Nearest Neighbors (KNN) Imputation**: Impute missing values by averaging the values of the K-nearest neighbors in feature space.

- **Modeling**: When you have sufficient data with complete values and believe that missing values can be predicted accurately. Treat predicting missing values as a separate machine learning task. Train a model to predict missing values based on other features in your dataset. This can be a more complex method, but it can also be more accurate than imputation.

- **Use Domain Knowledge**: Employ domain expertise to make informed choices regarding missing data. Utilize your specialized knowledge to decide the most suitable approach for managing missing values, taking into account the dataset's context.

The impact of missing data on a dataset will depend on the following factors:

- **The amount of missing data**: The more missing data there is, the greater the impact it will have.

- **The type of missing data**: Missing data can be either missing completely at random (MCAR), missing at random (MAR), or missing not at random (MNAR). MCAR is the least harmful type of missing data, while MNAR is the most harmful.

- **The goals of the analysis**: The goals of the analysis will also affect the impact of missing data. If the analysis is sensitive to missing data, then it is important to use a method that will minimize its impact.

1.2.4 Handling Outliers

Removing outliers from a dataset is a critical data preprocessing step to improve the quality and reliability of your data analysis and modeling.

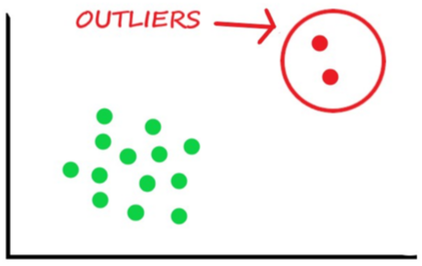


Figure 1.4: Data points with outliers

Here's a general approach on how to remove outliers from a dataset:

1. **Identify the Outliers**: Visualize your data using box plots, histograms, or scatter plots to spot potential outliers. You can also use statistical methods like the z-score or the interquartile range (IQR) to detect outliers. Data points with z-scores beyond a certain threshold or lying outside the IQR are often considered outliers.

2. **Choose a Removal Strategy**: There are several strategies to handle outliers such as Deletion, Transformations, Capping or Winsorizing, Imputation, and Model-Based, and the choice depends on your specific use case and dataset.

3. **Apply the Chosen Strategy**: Implement the chosen outlier removal strategy to clean your dataset.

4. **Document and Justify**: Document all the changes you made to your dataset, including which outliers were removed and why. This documentation is crucial for transparency and reproducibility.

5. **Reevaluate**: After removing outliers, re-evaluate your data to ensure it aligns with your analysis goals and that the removal process did not introduce any unintended biases.

6. **Sensitivity Analysis**: Perform sensitivity analysis to assess how different outlier removal methods impact your results. This can help you choose the most appropriate method for your specific analysis.

Remember that the decision to remove outliers should be made carefully and with a deep understanding of your data and the problem you're trying to solve. Outliers may contain valuable information or indicate data quality issues, so it's essential to strike the right balance between data integrity and data cleaning.

- **Detecting Outliers by Using Interquartile Range and Boxplots**: The boxplot is a graphical representation of the distribution of a dataset that can help you detect outliers. It provides a visual summary of the data's central tendency, spread, and potential outliers. The data points beyond the whiskers are considered potential outliers.

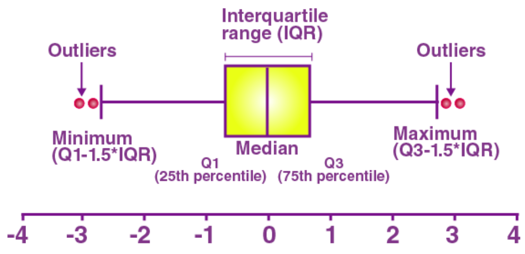


Figure 1.5: Detecting outliers using boxplot

- **Statistical Outlier Detection Using Z-Score**: Z-Score is a statistical measure used to identify and potentially remove outliers from a dataset. By calculating the Z-Score for each data point, you can determine how far away each point is from the mean of the dataset in terms of standard deviations. Typically, data points with Z-Scores that exceed a certain threshold are considered outliers and can be removed. Here's how to use the Z-Score method to remove outliers:

  1. **Calculate the Z-Score for Each Data Point**: For each data point in the dataset, calculate the Z-Score using the formula:

     $$Z = \frac{X - \mu}{\sigma} \tag{1}$$

     Where:

     – X is the data point you want to standardize.

     – μ is the mean (average) of the dataset.

     – σ is the standard deviation of the dataset.

  2. **Set a Threshold for Outliers**: Determine a Z-Score threshold beyond which data points are considered outliers. Commonly used thresholds are ± 2, ± 2.5, or ± 3 standard deviations from the mean. The choice of threshold depends on the desired level of sensitivity to outliers.

  3. **Identify Outliers**:

     – Data points with Z-Scores that exceed the chosen threshold are considered outliers.

     – If the Z-Score is greater than the threshold (in absolute value), it is an outlier.

4. **Handling Detected Outliers**: There are several strategies to handle detected outliers, and the choice depends on your specific use case and dataset. Here are some common approaches:

   (a) **Deletion**: Simply remove the outliers from your dataset. This can be done by excluding the rows containing outliers. However, this approach can lead to a loss of information.

   (b) **Transformations**: Apply mathematical transformations to your data to make it less sensitive to outliers. Common transformations include taking the logarithm, square root, or cube root of the data.

   (c) **Capping or Winsorizing**: Cap the extreme values by setting a threshold beyond which values are considered as outliers. You can replace these extreme values with the threshold value itself or with the nearest non-outlying data point.

   (d) **Imputation**: Instead of removing outliers, you can replace them with more reasonable values. This could be the mean, median, or a value predicted from a regression model. Imputation is often preferred when you don't want to lose too much data.

   (e) **Model-Based**: If you're working with predictive modeling, you can use robust models that are less sensitive to outliers, such as support vector machines, random forests, or robust regression techniques.

   The interpretation of outliers and the choice of handling them should be done carefully, considering the context of the data and the specific goals of the analysis. Not all outliers are errors, and they can sometimes provide valuable insights into the dataset. When the proportion of a feature outliers is small (<0.8%), the records corresponding to these outliers can be dropped.

1.3 Feature Selection and Feature Engineering

1.3.1 Feature Selection

Feature selection is a critical data preprocessing technique in machine learning that involves choosing a subset of the most relevant features (variables or columns) from a dataset. The goal of feature selection is to improve the performance of machine learning models by reducing the dimensionality of the data, eliminating irrelevant or redundant features, and enhancing model interpretability and generalization. There are several common feature selection methods, which can be categorized into three main types: filter methods, wrapper methods, and embedded methods.

- **Filter Methods** involves using statistical measures to identify and select the most relevant features from a dataset. Some common filter methods for feature selection include:

  - **Variance**: This method removes features that have a low variance, which means that they do not vary much across the data. This can help to remove noise and irrelevant features.

  - **Information gain**: This method measures the reduction in entropy that is achieved by knowing the value of a feature. It can be used to identify features that are most informative for predicting the target variable.

  - **Chi-squared test**: This method is used to test the independence of two variables. It can be used to identify features that are correlated with the target variable, which can be useful for classification and regression tasks.

  - **Correlation coefficient**: This method measures the linear relationship between two variables. It can be used to identify features that are highly correlated with each other, which can be removed to reduce multicollinearity.

1.3.2 Data Transformation

Data Transformation is a crucial step in the data preprocessing phase of machine learning. It involves a series of operations aimed at preparing raw data into a format that is suitable for training and evaluating machine learning models. Data transformation encompasses several key processes, including scaling, normalization, discretization, and encoding, each serving its purpose in enhancing the quality and usability of the data. Here's a description of these data transformation techniques:

- **Scaling**: This method involves transforming numerical features to a common scale without changing their relative relationships. The Min-Max scaling is a commonly used method in which features are scaled to a specific range, often [0, 1]. The standardization is another type of scaling that transforms numerical features to have a zero mean and unity variance (mean=0, standard deviation=1).

- **Discretization**: This method involves converting continuous numerical data into discrete categories or bins. This is particularly useful when you want to transform numeric data into categorical or ordinal data. Discretization can simplify complex numerical data and make it suitable for algorithms that work with categorical or ordinal features. Some common techniques include equal-width binning (dividing the data into equal-width intervals) and equal-frequency binning (ensuring each bin contains approximately the same number of data points).

- **Encoding**: Encoding is the process of converting categorical data (text or labels) into a numerical format that machine learning models can understand. It allows algorithms to work with non-numeric data Some common encoding methods include **one-hot encoding** (creating binary columns for each category), **label encoding** (assigning a unique integer to each category), and **ordinal encoding** (mapping ordinal categories to numerical values).

1.3.3 Handling High-Dimensional Data

There are two prominent methods for dimensionality reduction: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

- **Principal Component Analysis (PCA)**: PCA is one of the most widely used techniques for reducing the dimensionality of data. It transforms the original features into a new set of orthogonal features called principal components. These components capture the maximum variance in the data, allowing for dimensionality reduction while retaining as much information as possible.

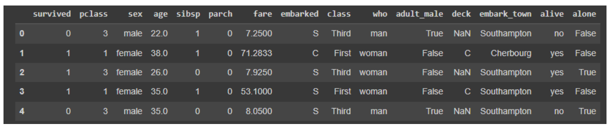
- **Linear Discriminant Analysis (LDA)**: LDA is used when the goal is not just dimensionality reduction but also class separation. It finds a linear combination of features that maximizes the distance between different classes while minimizing the variance within each class.

Both PCA and LDA are dimensionality reduction techniques, PCA is unsupervised and aims to capture maximum variance, whereas LDA is supervised and focuses on maximizing class separation. The choice between PCA and LDA depends on the specific goals of your machine learning task and whether you are dealing with a classification problem.

2 Procedure and Discussion

2.1 Loading The Titanic Dataset

In this step, the Titanic dataset is loaded using the provided code snippet in the Jupyter Notebook file. Subsequently, the header of the dataset is displayed.



	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

Figure 2.1: Titanic dataset header

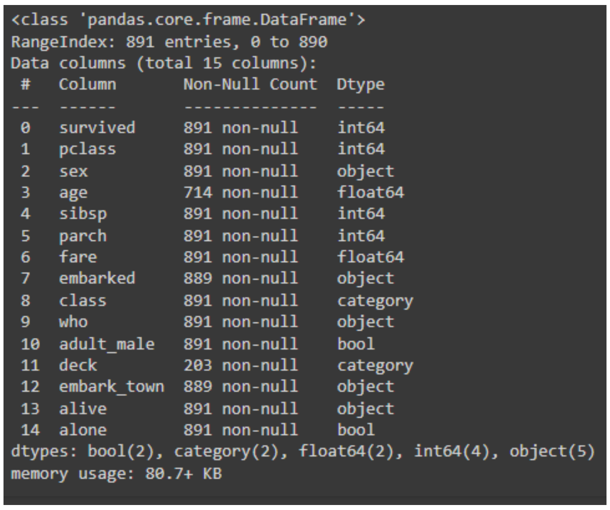
Initially, the dataset has 15 columns (features): survived, pclass, sex, age, sibsp, fare, embarked, class, who, adult_male, deck, embark_town, alive, alone. Also, there is missing values in the dataset such as row 1 in column deck.

2.2 Data Exploration

The aim of this step is to understand the dataset's structure, features, summarize its statistics, and gain insights into the data.

2.2.1 Dataset Structure

To understand the dataset structure, info() method is used. This method displays: the number of entries and columns, each column with the number of non-null entries and its type.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Figure 2.2: Dataset information

The features we have in the dataset: survived, pclass, sex, age, sibsp, parch, fare, embarked, class, who, adult_male, deck, embark_town, alive, alone.

The number of **features** is 15.

The number of **entries** is 891.

2.2.2 Data Visualization

This part is to understand the features and their relationship by visualizing them.

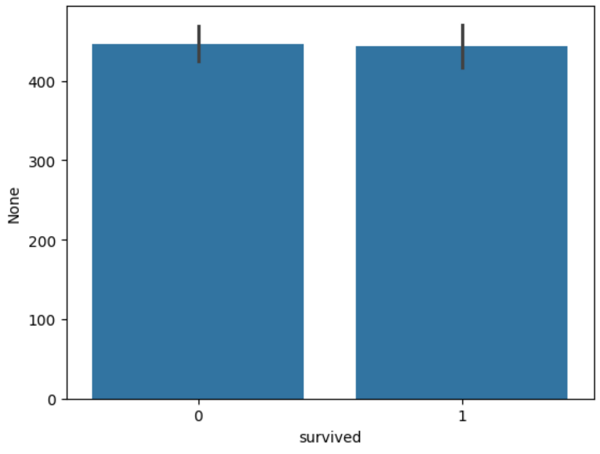


Figure 2.3: Distribution of survived feature

This is the distribution of **survived**. It consists of two types: 0, and 1. 0 means did not survive and 1 means survived. We can see that the number of survivors is almost the same as the number of the ones who did not survive.

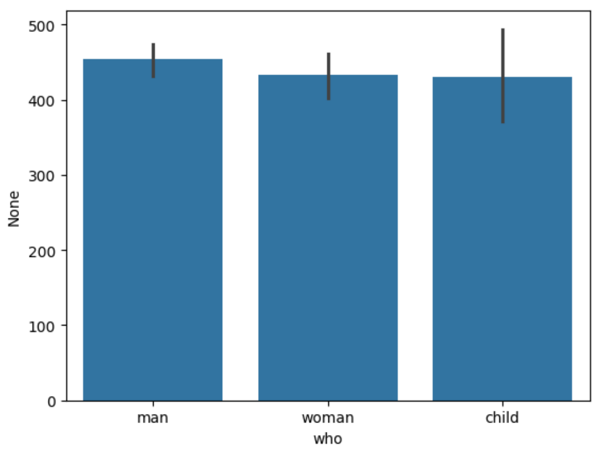


Figure 2.4: Distribution of who feature

Figure 2.4 shows the distribution of **who**. It consists of three types: man, woman, and child. The number of men is the most frequent.

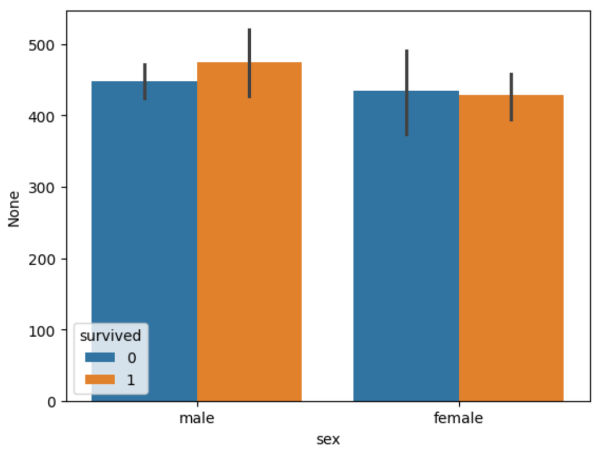


Figure 2.5: Distribution of survival state for each sex

Figure 2.5 shows the distribution of the of **sex according to survival**. We an see that more than of the half of the males survived, while more than the half of the females did not survive.

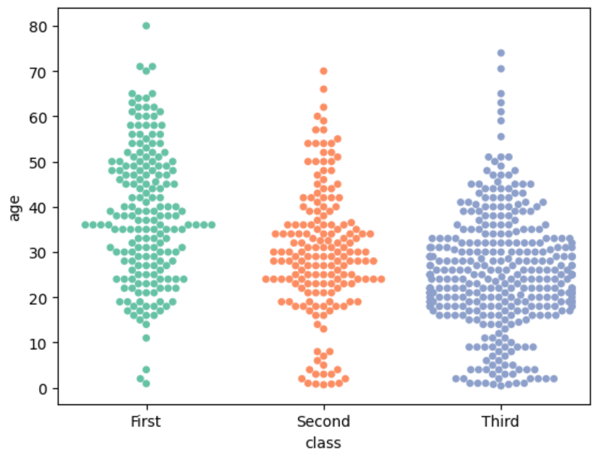


Figure 2.6: Distribution of ages among classes

Figure 2.6 shows the distribution of **ages for each class**. We can see that most of the ages are in the range of 20 to 40.

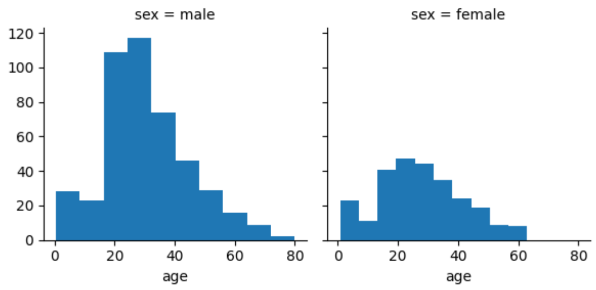


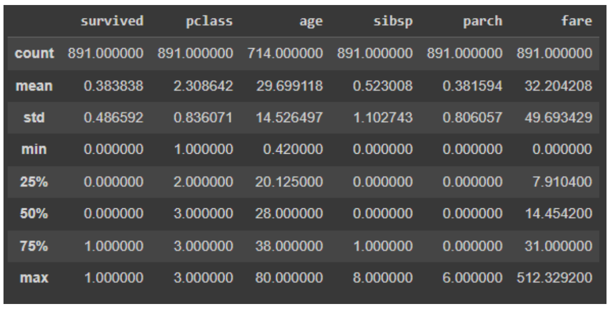
Figure 2.7: Distribution of ages among sex

Figure 2.7 shows the distribution of **ages for each sex**. We can see that males have higher range of age than females.

To shorten the report, the most important plots were introduced above. There are many plots available on the jupyter file with their disunion.

2.2.3 Descriptive Statistics

The **describe()** method is used to describe the dataset as in figure 2.8 below.



	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Figure 2.8: Dataset statistics

It provides statistical values like: count, mean, std, min, percentiles, and max. Each value represents the following:

- **Count**: the number of values in the column
- **Mean**: the average value of the data.
- **Standard Deviation (STD)**: The square root of the variance, indicating the spread of data.
- **Min**: finds the smallest value in the column.
- **Max**: finds the greatest value in the column.
- **Percentiles**: Values below which a given percentage of data falls, 25%, 50%, 75%.

To gain some further details on the dataset, lets analyze it using the following:

1. **Central tendency** mean, median, and mode, lets see figure 2.9 below:

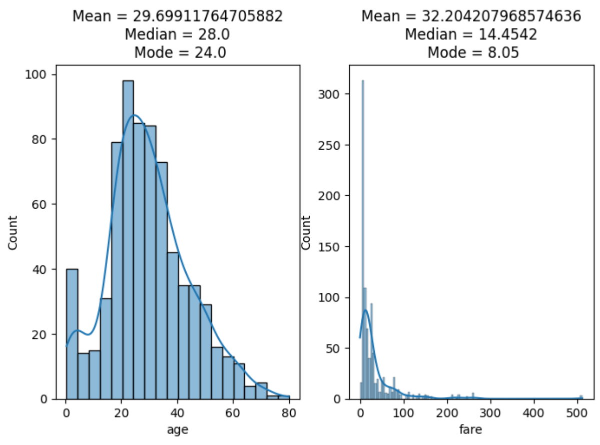


Figure 2.9: The distribution of features with their central tendency

- **Age**: it is close to a **normal** distribution, but the **mode** is slightly far from **mean & median**.

- **Fare**: from the measures, it is not likely to be a normal distribution.

With these measures (mean, median, mode), we are unable to confirm that the distributions are normal (Gaussian), we should use other measures.

2. **Variation** The subsequent metrics are utilized to evaluate the dispersion of data distribution:

   - **Variance**: A measure of how much the data points deviate from the mean.

   - **Standard Deviation**: The square root of the variance, indicating the spread of data.

   Lets take a look at figure 2.10 below:

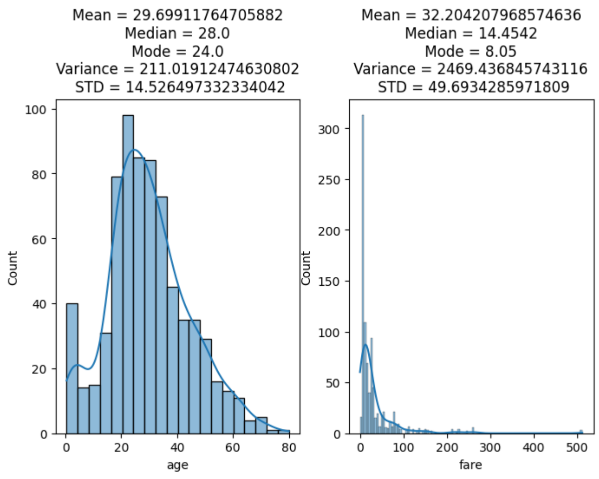


Figure 2.10: The distribution of features with their variation

   From the above figure, we can notice the following:

   - **Age**: It has a high variance and low standard deviation, indicating more spread around the mean.

   - **Fare**: It has a significantly higher variance and standard deviation, suggesting considerable variability or dispersion around the mean, possibly due to outliers or a non-normal distribution.

   Overall, these statistical values provide insights into the central tendency, spread, and symmetry of the distributions for each variable, helping to understand their characteristics and potential patterns in the data.

3. **Shape of distribution** Skewness and Kurtosis are measures to determine the shape of the distribution.

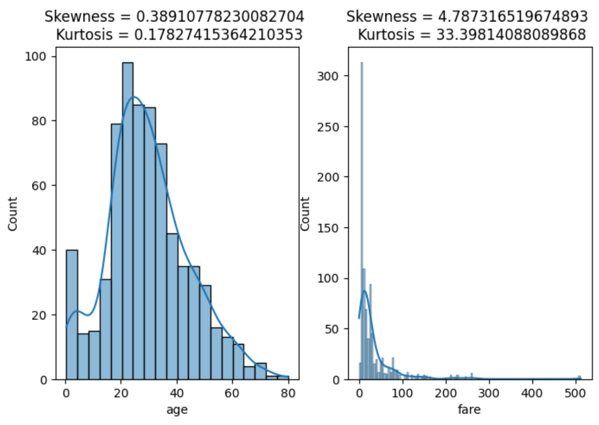


Figure 2.11: Distribution of ages among sex

From figure 2.11 above, we can see that both variables have positive kurtosis values, suggesting that they have heavier tails than a normal distribution. And we can see that both variables are positively skewed. These values collectively describe the shape and characteristics of the distributions for each variable. Positive kurtosis indicates relatively more extreme values compared to a normal distribution, while skewness helps understand the asymmetry in the data distribution. These values can assist in identifying deviations from normality and understanding the nature of the data distributions.

4. **Quantiles**

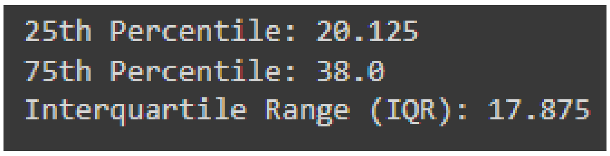

```
25th Percentile: 20.125
75th Percentile: 38.0
Interquartile Range (IQR): 17.875
```

Figure 2.12: Distribution of ages among sex

From figure 2.12 above, we can notice the following:

- The 25th percentile (Q1) of 20.125 indicates that a quarter of the dataset's values lie below this point.
- The 75th percentile (Q3) at 38.0 shows that three-quarters of the dataset's values are less than or equal to this value.
- The IQR of approximately 17.875 suggests that the middle 50% of the data falls within this range, showing the spread of the central data points.

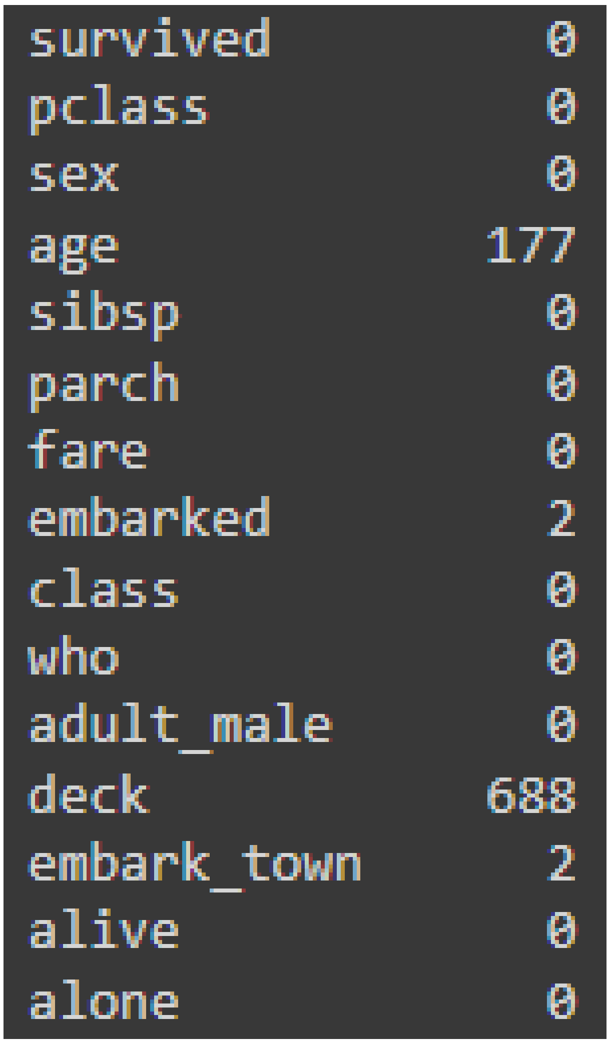
In summary, these quartile values and the IQR help understand the spread and dispersion of the central portion of the dataset, offering insights into its variability without being influenced by extreme values (outliers).

2.3 Data Quality Issues

This section focuses on identifying and addressing missing values and potential outliers within the dataset. Additionally, it involves detecting any outliers present and implementing appropriate handling measures if necessary.

2.3.1 Detecting Missing Values

Lets start by finding the missing values for each feature.



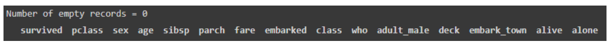
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
```

Figure 2.13: Features with the number of missing values

Figure 2.13 above shows the number of missing values for each feature.

- **Age** has 177 missing values.
- **Embarked** and **Embark_town** has 2 missing values each.
- **Deck** has 688 missing values.

Some datasets may include completely empty entries, so we must check for it as in figure 2.14 below:



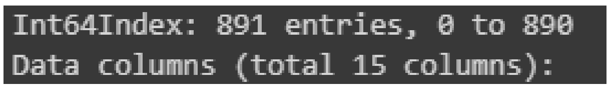
```
Number of empty records = 0
   survived  pclass  sex  age  sibsp  parch  fare  embarked  class  who  adult_male  deck  embark_town  alive  alone
```

Figure 2.14: Addressing fully empty rows

The dataset doesn't include any empty rows.

2.3.2 Handling Missing Values

1. **Handling missing data through dropping** At first, we will try to drop entries that has at least 80% of the features as nulls if their proportion of the dataset is not a lot.

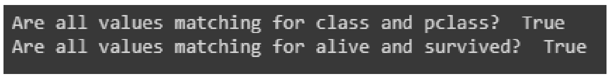


```
Int64Index: 891 entries, 0 to 890
Data columns (total 15 columns):
```

Figure 2.15: Entries after dropping entries that has at least 80% of the features as nulls

From the figure 2.15 above, we can see that the number of the entries did not change (891), so it means that there are no entries with much empty values.

We can notice some similarities between some of the columns, like pclass and class, and survived and alive, so we will check if these pairs are the same.



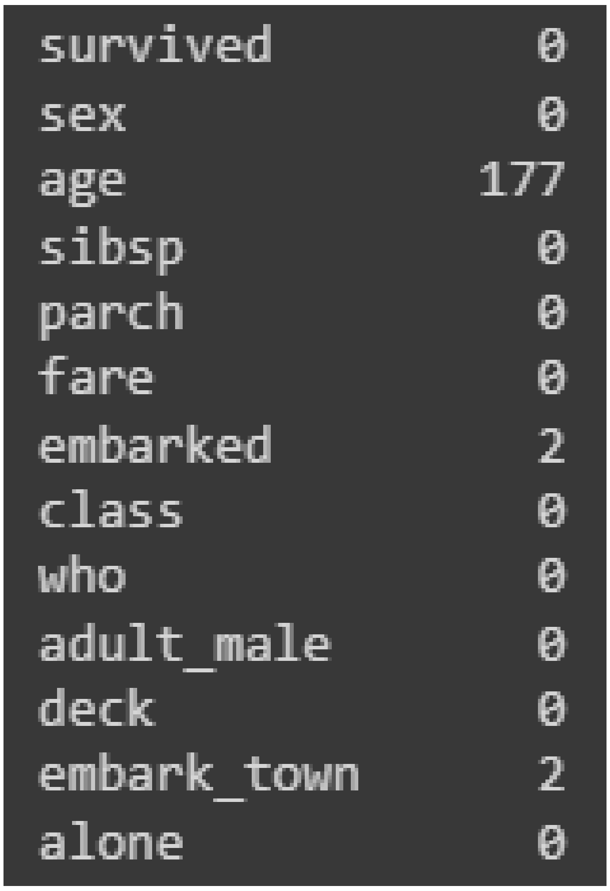
```
Are all values matching for class and pclass?  True
Are all values matching for alive and survived?  True
```

Figure 2.16: Entries after dropping entries that has at least 80% of the features as nulls

We can see that these two pairs are the same, so the existence of multi equal features is not good for the training, so we will drop one from each pair.

From figure 2.13, we can see that the deck feature has a lot of empty values (688), dropping the feature can be a solution, but after trying it, the accuracy got worse. And if we try filling with the mode, a huge proportion of this feature will be the same value, so it is not a good option. So, we used forward filling, that is, if the value is null, set it as the one before it, and it was kind of equally distributed.

2. **Handling Missing Data Through Auto Filling**



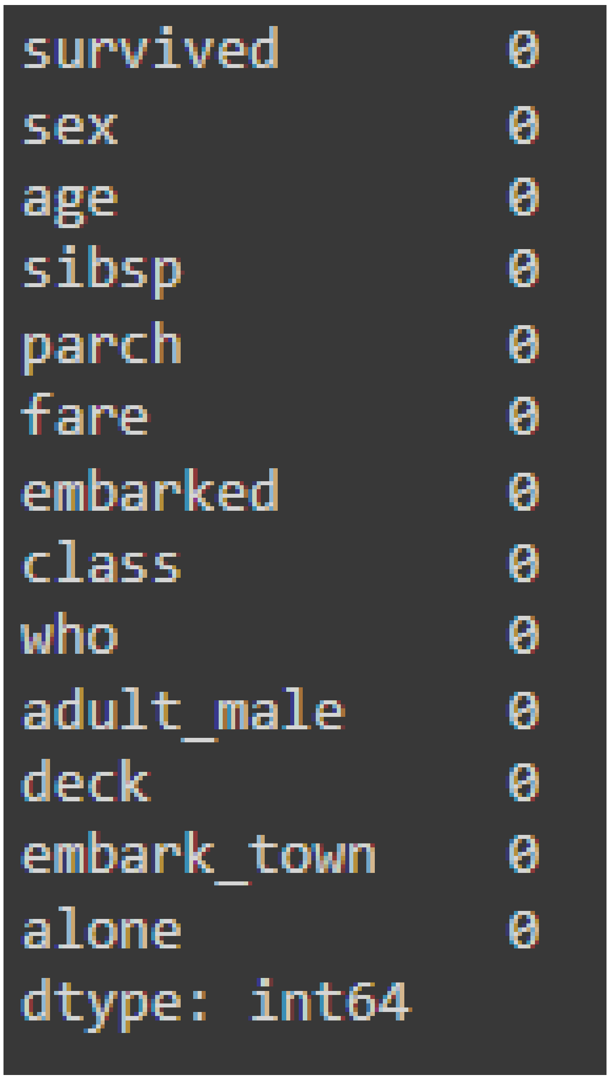
```
survived         0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck             0
embark_town      2
alone            0
```

Figure 2.17: Remaining missing values

From the figure 2.17 above, we can see that there are 3 features with missing values, 2 of them are categorical, and one is numerical (age). We will fill the missing values of the age using the median, because the mean can introduce a new value and sensitive to outliers, so it is not a good measure.

For the other two categorical values, we will fill them using the mode, which is the best option for these categorical features since there is not that much of null values (only 2 per column).

3. **Handling Missing Data Through Auto Filling**



```
survived       0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alone          0
dtype: int64
```

Figure 2.18: Ensuring missing values are gone

We can see from figure 2.18 above that there are no missing values, so we can continue to the next steps.

2.3.3 Detecting Outliers

This part is concerned about detecting outliers, there are various methods for detecting them. To summarize this part, we will use **sns.boxenplot()**, it is similar to a box plot but provides more quantiles for a deeper insight into the distribution of the data.

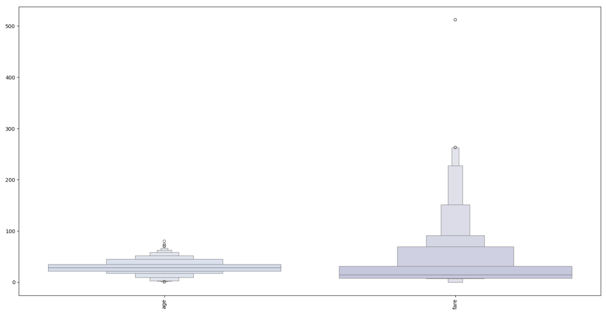
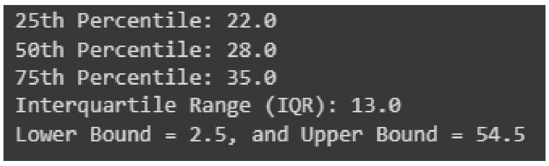


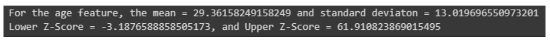
Figure 2.19: Outliers before removing them

We can see from figure 2.19 above that there are outliers for the features age and fare so we need to remove them. We will compare the scores of two methods, by checking the Z-Score and the IQR.



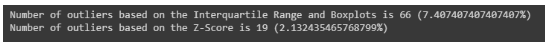
```
25th Percentile: 22.0
50th Percentile: 28.0
75th Percentile: 35.0
Interquartile Range (IQR): 13.0
Lower Bound = 2.5, and Upper Bound = 54.5
```

(a) Age percentiles, IQR, and lower and upper bounds



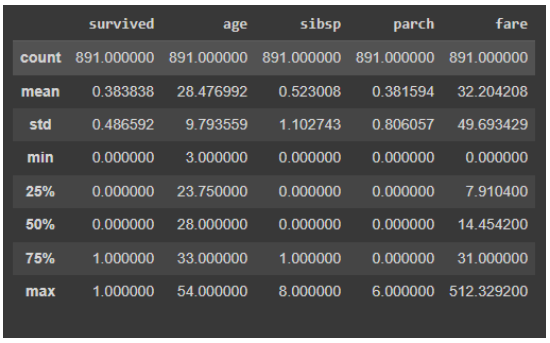
```
For the age feature, the mean = 29.36158249158249 and standard deviaton = 13.019696550973201
Lower Z-Score = -3.1876588858505173, and Upper Z-Score = 61.910823869015495
```

(b) Age mean, std, and lower and upper Z scores



```
Number of outliers based on the Interquartile Range and Boxplots is 66 (7.407407407407407%)
Number of outliers based on the Z-Score is 19 (2.132435465768799%)
```

(c) Percentage of outliers for each method

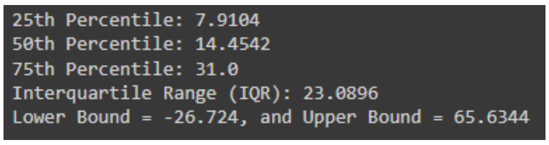


	survived	age	sibsp	parch	fare
count	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	28.476992	0.523008	0.381594	32.204208
std	0.486592	9.793559	1.102743	0.806057	49.693429
min	0.000000	3.000000	0.000000	0.000000	0.000000
25%	0.000000	23.750000	0.000000	0.000000	7.910400
50%	0.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	33.000000	1.000000	0.000000	31.000000
max	1.000000	54.000000	8.000000	6.000000	512.329200

(d) Removing age outliers result

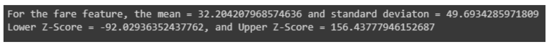
Figure 2.20: Removing age outliers

We will use the IQR values to update the outliers, since their percentage is 7% and we cant delete them. So, anything above the upper bound will be set to the median, and anything lower than the lower bound will also be set to the median. From figure 2.20d above, we can see that the range of the age is now between 3 and 54, the outliers were filled with the median of the age. Now lets process the fare the same way.



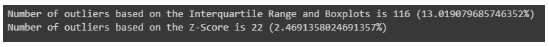
```
25th Percentile: 7.9104
50th Percentile: 14.4542
75th Percentile: 31.0
Interquartile Range (IQR): 23.0896
Lower Bound = -26.724, and Upper Bound = 65.6344
```

(a) Fare percentiles, IQR, and lower and upper bounds



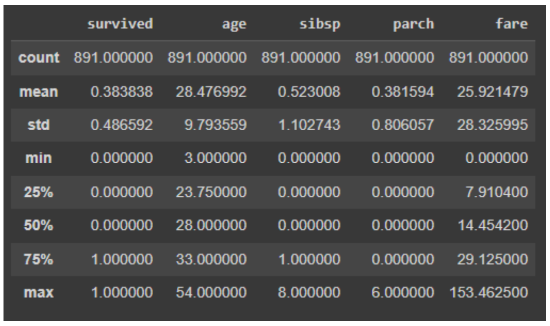
```
For the fare feature, the mean = 32.204207968574636 and standard deviaton = 49.6934285971809
Lower Z-Score = -92.02936352437762, and Upper Z-Score = 156.43777946152687
```

(b) Fare mean, std, and lower and upper Z scores



```
Number of outliers based on the Interquartile Range and Boxplots is 116 (13.019079685746352%)
Number of outliers based on the Z-Score is 22 (2.4691358024691357%)
```

(c) Percentage of outliers for each method



	survived	age	sibsp	parch	fare
count	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	28.476992	0.523008	0.381594	25.921479
std	0.486592	9.793559	1.102743	0.806057	28.325995
min	0.000000	3.000000	0.000000	0.000000	0.000000
25%	0.000000	23.750000	0.000000	0.000000	7.910400
50%	0.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	33.000000	1.000000	0.000000	29.125000
max	1.000000	54.000000	8.000000	6.000000	153.462500

(d) Removing fare outliers result

Figure 2.21: Removing fare outliers

We will use the Z-Score range to update the outliers, since their range is more reasonable. So, anything above the upper bound will be set to the median, and anything lower than the lower bound will also be set to the median. From figure 2.21d above, we can see that the range of the age is now between 0 and 153.46, the outliers were filled with the median of the fare. Figure 2.22 below shows the plot after removing the outliers. We can see that most of the outliers were removed.

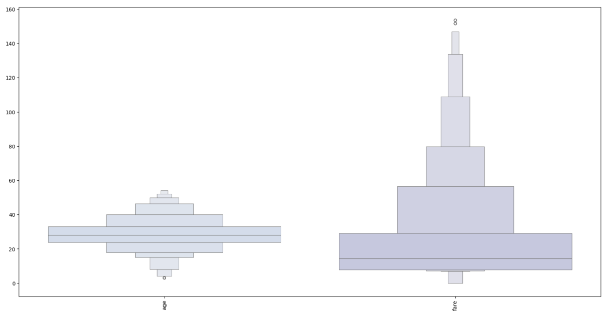
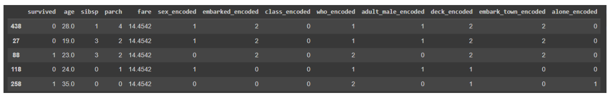


Figure 2.22: Outliers after removing them

2.4 Encoding Categorical Variables

In this case study, the label encoder was employed due to its ability to maintain the original distribution of features without increasing their count. Although One-hot encoding was experimented with, it resulted in inferior outcomes.

We have eight categorical features: sex, embarked, class, who, adult_male, deck, embark_town, and alone. Here is the result after encoding them:



	survived	age	sibsp	parch	fare	sex_encoded	embarked_encoded	class_encoded	who_encoded	adult_male_encoded	deck_encoded	embark_town_encoded	alone_encoded
438	0	28.0	1	4	14.4542	1	2	0	1	1	2	2	0
27	0	19.0	3	2	14.4542	1	2	0	1	1	2	2	0
88	1	23.0	3	2	14.4542	0	2	0	2	0	2	2	0
118	0	24.0	0	1	14.4542	1	0	0	1	1	1	0	0
258	1	35.0	0	0	14.4542	0	0	0	2	0	1	0	1

Figure 2.23: Encoding categorical features

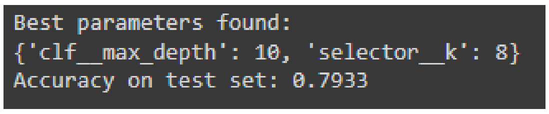
The figure 2.23 above is a snippet of the dataset after encoding the eight features.

2.5 Splitting The Dataset Into Training and Testing Subsets

This step is important for all upcoming parts. For example, feature selection can't be done without splitting the data into X, y

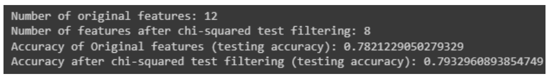
2.6 Feature Selection

In this part, we will need a filtering method and the number of features to select. We will use chi2 (chi-squared test) along with grid search to give the best K (number of features) along with the best accuracy.



```
Best parameters found:
{'clf__max_depth': 10, 'selector__k': 8}
Accuracy on test set: 0.7933
```

(a) Best parameters for chi2 along with random forest



```
Number of original features: 12
Number of features after chi-squared test filtering: 8
Accuracy of Original features (testing accuracy): 0.7821229050279329
Accuracy after chi-squared test filtering (testing accuracy): 0.7932960893854749
```

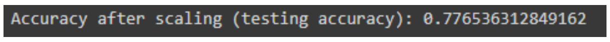
(b) Results before and after applying the best parameters

Figure 2.24: Feature selection result

We can see from figure 2.24a that the best K for chi2 is 8, and the best depth for the random forest classifier is 10, and from figure 2.24b that it gets an 79.33% accuracy. The accuracy is 79.33% after dropping the features. However, this is the best case solution because of the grid search. The accuracy without dropping is 78.21%.

2.7 Scale or Normalize The Numerical Features

In this section, the focus is on scaling the numerical features. Two scaling techniques were explored: min-max scaling and standardization. Through experimentation, it was observed that standardization gave the same result as the min-max scaling, so any of them can be chosen.



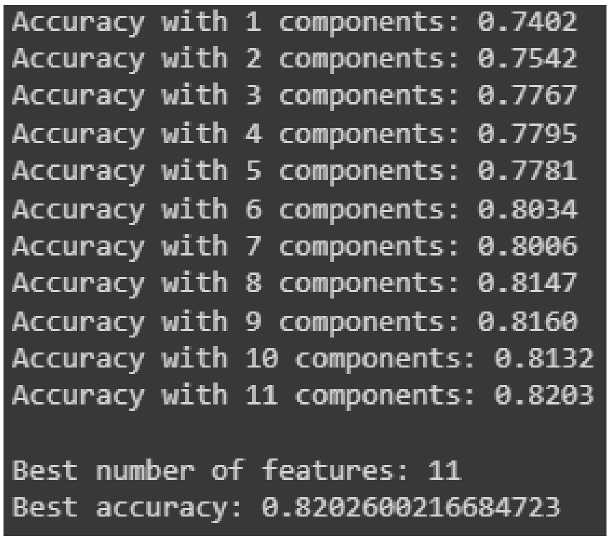
```
Accuracy after scaling (testing accuracy): 0.776536312849162
```

Figure 2.25: Result after scaling

We can see that the scaling did not improve the accuracy, it made it worse.

2.8 Apply Suitable Dimensionality Reduction Techniques

Following the completion of feature selection and scaling, the subsequent step involves reducing the dimensionality of the dataset. This entails minimizing the number of features utilized. Principal Component Analysis (PCA) is employed for this purpose, with a critical parameter known as n_components, dictating the number of components to retain.



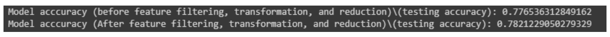
```
Accuracy with 1 components: 0.7402
Accuracy with 2 components: 0.7542
Accuracy with 3 components: 0.7767
Accuracy with 4 components: 0.7795
Accuracy with 5 components: 0.7781
Accuracy with 6 components: 0.8034
Accuracy with 7 components: 0.8006
Accuracy with 8 components: 0.8147
Accuracy with 9 components: 0.8160
Accuracy with 10 components: 0.8132
Accuracy with 11 components: 0.8203

Best number of features: 11
Best accuracy: 0.8202600216684723
```

Figure 2.26: PCA result

After applying cross validation, the best accuracy is 82% where the PCA chose all of the features (11). Figure 2.26 above demonstrates the results.

2.9 Validating Preprocessing Pipeline and Comparing The Results

The validation of the preprocessing pipeline, encompassing feature filtering, transformation, and reduction, is assessed through the accuracy measure applied to the testing set.



```
Model acccuracy (before feature filtering, transformation, and reduction)\(testing accuracy): 0.776536312849162
Model acccuracy (After feature filtering, transformation, and reduction)\(testing accuracy): 0.7821229050279329
```

Figure 2.27: Model accuracy before and after

The figure 2.27 above shows that the model's accuracy increased after the filtering, transformation, and reduction, which indicates that these operations were helpful in improving the accuracy. But the overall accuracy is not good, since it suffers from under-fitting.

3 Conclusion

In conclusion, this case study underscores the crucial significance of thorough data preprocessing in refining the quality and applicability of the Titanic dataset for predictive modeling purposes. Through systematic handling of missing values, careful selection of pertinent features, encoding of categorical data, and appropriate partitioning, we ensure that the dataset is well-prepared for analysis using machine learning techniques. Utilizing various visualization tools aids in intuitively understanding the dataset, facilitating pattern recognition and comprehension, while descriptive statistics offer an initial overview essential for identifying patterns, anomalies, and potential issues that could impact model efficacy. Dealing with missing data requires judicious decision-making based on the quantity and characteristics of the missing values, and managing outliers necessitates thoughtful considerations to preserve data integrity. Moreover, employing feature selection methods contributes to enhancing model performance by reducing dimensionality and eliminating irrelevant features. Lastly, data transformation techniques ensure that the data is appropriately formatted for utilization in machine learning models. The observed marginal improvement in accuracy, from 77.6% to 78.2%, following the implementation of scaling and dimensionality reduction techniques underscores the importance of striking a balance between data transformation and model performance considerations. Despite the dataset's limited numerical values and relationships, the meticulous steps taken in data cleaning remain indispensable. This holistic approach underscores the critical role of deliberate preprocessing steps in augmenting the quality and dependability of subsequent machine learning analyses.