Part 2

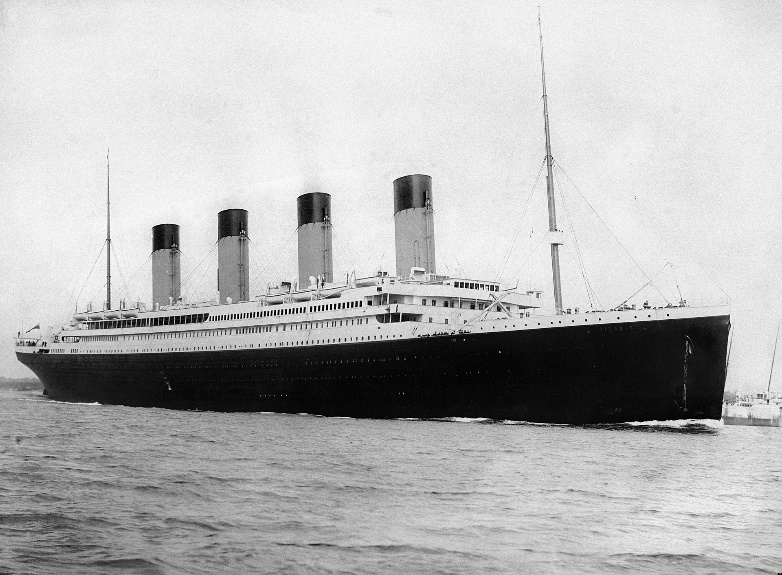
Section 1: Exploratory Data Analysis of The Titanic’s Dataset

**Preface**

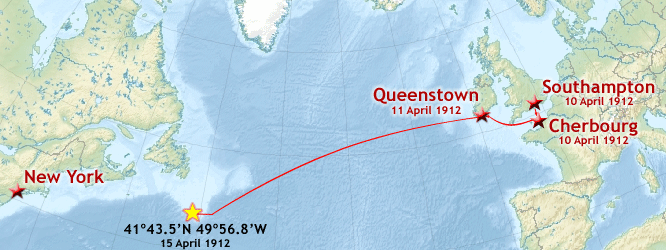
This EDA is part of an assignment. As such, we use citations and have included a References section, although they are not part of an actual EDA.

**1. Historical Data**

The Titanic was an Olympic-class ocean liner belonging to White Star Line, a British shipping company. An engineering miracle, her construction lasted three years (1909 – 1912) and she was the largest ship of her time, with a length of 268.1m., height of 175m. and weight of 52.310 tons. The first half of her planned itinerary was from Southampton, England, to New York, USA, with intermediate stops in Cherbourg, France, and Queenstown, Ireland. Her maiden voyage would also be her last, though, as on April the 15th, 1912, only five days in her journey, she hit an iceberg and sank southeast of Newfoundland, Canada, killing 1.522 of the approximately 2.227 passengers aboard (Wikipedia, 2023b; Smithsonian, 2017).



**Figure 1**: The Titanic (Source: Wikipedia, 2023b)



**Figure 2**: The Titanic’s itinerary and shipwreck coordinates (Source: Wikipedia, 2023b)

The first reason so many people died was that the number of emergency lifeboats was not adequate and could not accommodate all of the passengers (although it must be mentioned that it complied with contemporary maritime safety regulations). The second was that The Titanic’s crew was not trained on an evacuation plan and was not aware of the lifeboats’ maximum capacity, resulting to many of them leaving the ship with very few persons inside (Wikipedia, 2023b).

Since then, The Titanic has acquired a legendary status and has been the subject of numerous publications (books, movies, documentaries etc.). Being the most famous shipwreck of all time, it has secured a place in humanity’s history.

**2. EDA Background and Methodology**

The dataset this EDA uses is provided by Kaggle (2019), as part of its running “Titanic - Machine Learning from Disaster” competition, whose goal is to use ML methods to predict which passengers would survive the accident and which not.

Our general intention is to visualize the data in ways that make their interpretation easy and evident and to attempt to impute any missing values in a correct and logical manner, so that the dataset becomes as valid and complete as possible. Parts of the code we have used were inspired by Shin (2020), Nair (2019), Prajwal (2022) and Nadia (2021) and the Python libraries we imported were NumPy, pandas, Matplotlib, seaborn and statistics.

**3. Acquainting Ourselves with the Data**

Kaggle provides us with two datasets: a training and a test one. After importing them, we combine them into one, so that our EDA is performed on all available data. The resulting Dataframe object is comprised of 1309 rows and 12 columns:

#Imports the two datasets.trainingData = pd.read\_csv("/kaggle/input/titanic/train.csv")

testData = pd.read\_csv("/kaggle/input/titanic/test.csv")

#Combines the two datasets into one.

datasetsToBeJoined = [trainingData, testData]

combinedData = pd.concat(datasetsToBeJoined, sort = False, ignore\_index = True)

Combined data rows and columns: (1309, 12)

After consulting Kaggle’s (2019) “data dictionary” for information on the dataset’s variables, we familiarize ourselves with them by running a number of functions.

“nunique” displays every column’s **number of unique values**:

#Total unique values in each column.

combinedData.nunique(axis = 0)

PassengerId 1309

Survived 2

Pclass 3

Name 1307

Sex 2

Age 98

SibSp 7

Parch 8

Ticket 929

Fare 281

Cabin 186

Embarked 3

dtype: int64

There certainly exist missing values in *Age* column, because its contents are too few for 1309 different rows.

“dtypes” shows us each column’s **variable type**, which is useful for choosing the appropriate function to work with our data or for future type conversions:

#Data type of each column.

combinedData.dtypes

PassengerId int64

Survived float64

Pclass int64

Name object

Sex object

Age float64

SibSp int64

Parch int64

Ticket object

Fare float64

Cabin object

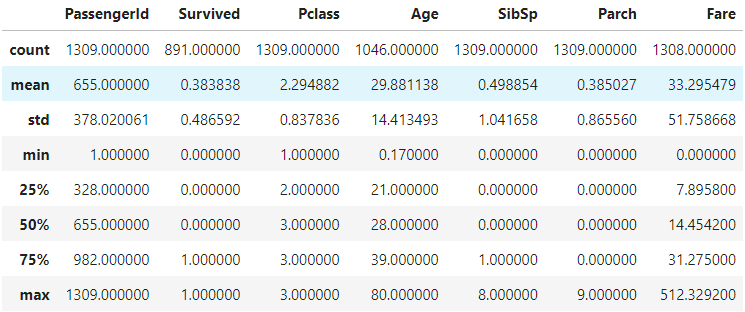
Embarked object

dtype: object

“describe” returns a more detailed **description of our numerical data**:

#"Summarizes the count, mean, standard deviation, min, and max for numeric variables".

combinedData.describe().apply(lambda s: s.apply(lambda x: format(x, 'f')))



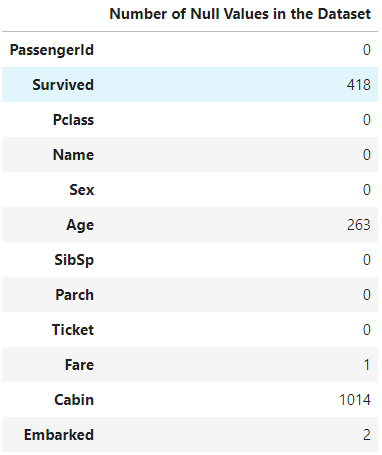
Half (~650) of the total passengers were 28 years old or less, only 25% of them bought 2nd and 1st Class tickets and very few travelled with siblings, spouses, children or parents. We can also see the minimum and maximum values of every column consisting of numerical data, as well as its Standard Deviation and mean value.

Lastly, “isna” shows us the number of each column’s **null** (NaN) (missing) values:

#Displays number of null entries for each column.

nullValues = combinedData.isna().sum()

pd.concat([nullValues], axis = 1, sort = False, keys = ['Number of Null Values in the Dataset'])



*Name* has two sets of duplicate entries, because it has no missing values but we previously saw that it had 1307 out of 1309 unique values.

**4. Data Preprocessing and Visualization**

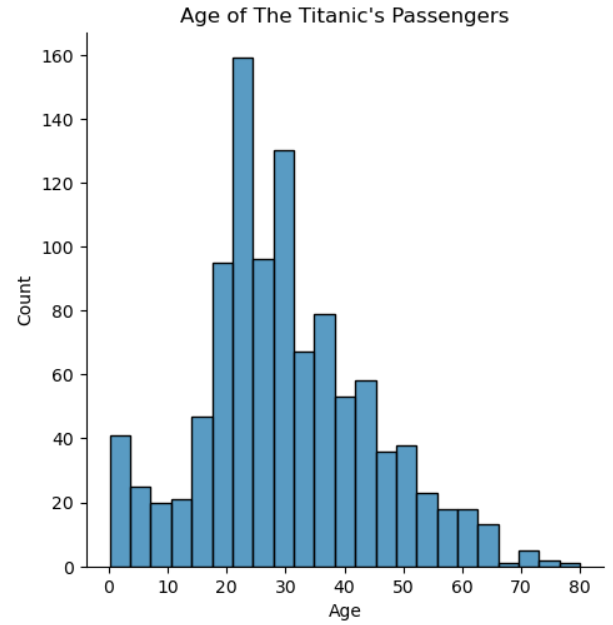
We begin by visualizing the columns that contain missing values, so that we optimize our dataset as soon as possible and proceed with plotting all remaining columns.

***Age*** has too many values to be correctly displayed, so we use a histogram, which automatically bins values:

#Histogram of Age column.

sns.displot(combinedData['Age']).set(title = 'Age of The Titanic\'s Passengers')

plt.show()

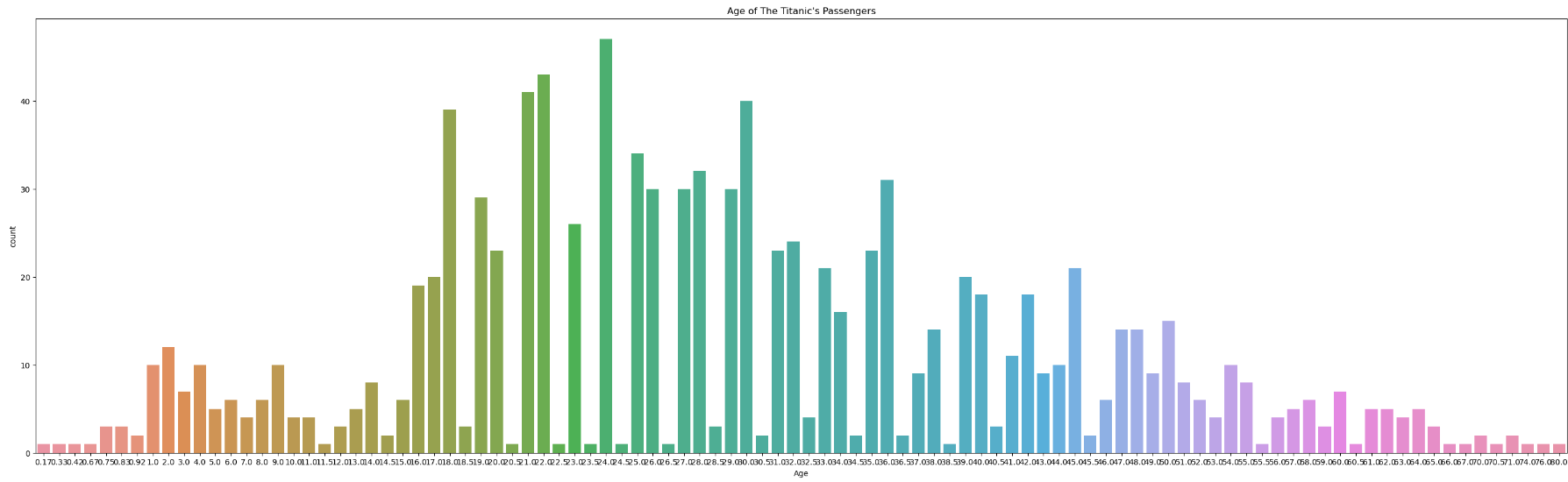


As noted before, most of the passengers were 20 – 30 years old. We also create a count plot, which does not bin values, to ensure that our data is right-skewed:

#Count plot of Age column.

plt.figure(figsize = (35, 10))

sns.countplot(x = combinedData["Age"]).set(title = 'Age of The Titanic\'s Passengers')



Missing data of right-skewed distributions can be replaced with the column’s median value (Zimmerman, 2023). In this case it is the number 28. After the replacement, we replot the data:

#Replaces null values in column Age with the value "28" (median).

np.nan\_to\_num(combinedData["Age"], False, 28)

nullValues = combinedData.isna().sum()

pd.concat([nullValues], axis = 1, sort = False, keys = ['Number of Null Values in the Dataset'])

#Histogram of Age column.

sns.displot(combinedData['Age']).set(title = 'Age of The Titanic\'s Passengers')

plt.show()

#Count plot of Age column.

plt.figure(figsize = (35, 10))

sns.countplot(x = combinedData["Age"]).set(title = 'Age of The Titanic\'s Passengers')

#Line graph of Age column.

ageCount = pd.Series(combinedData.Age).value\_counts()

ageLabels = ['0.42', '0.67', '0.75', '0.83', '0.92', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '14.5', '15', '16', '17', '18', '19', '20', '20.5', '21', '22', '23','23.5', '24','24.5','25','26', '27', '28', '28.5', '29', '30', '30.5', '31', '32', '32.5', '33', '34', '34.5', '35', '36', '36.5', '37', '38', '39', '40', '40.5', '41', '42', '43', '44', '45', '45.5', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '55.5', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '70', '70.5', '71', '74', '80']

sizes = [ageCount.get(0.42), ageCount.get(0.67), ageCount.get(0.75), ageCount.get(0.83), ageCount.get(0.92), ageCount.get(1), ageCount.get(2), ageCount.get(3), ageCount.get(4), ageCount.get(5), ageCount.get(6), ageCount.get(7), ageCount.get(8), ageCount.get(9), ageCount.get(10), ageCount.get(11), ageCount.get(12), ageCount.get(13), ageCount.get(14), ageCount.get(14.5), ageCount.get(15), ageCount.get(16), ageCount.get(17), ageCount.get(18), ageCount.get(19), ageCount.get(20), ageCount.get(20.5), ageCount.get(21), ageCount.get(22), ageCount.get(23), ageCount.get(23.5), ageCount.get(24), ageCount.get(24.5), ageCount.get(25), ageCount.get(26), ageCount.get(27) ,ageCount.get(28), ageCount.get(28.5), ageCount.get(29), ageCount.get(30), ageCount.get(30.5), ageCount.get(31), ageCount.get(32), ageCount.get(32.5), ageCount.get(33), ageCount.get(34), ageCount.get(34.5), ageCount.get(35), ageCount.get(36), ageCount.get(36.5), ageCount.get(37), ageCount.get(38), ageCount.get(39), ageCount.get(40), ageCount.get(40.5), ageCount.get(41), ageCount.get(42), ageCount.get(43), ageCount.get(44), ageCount.get(45), ageCount.get(45.5), ageCount.get(46), ageCount.get(47), ageCount.get(48), ageCount.get(49), ageCount.get(50), ageCount.get(51), ageCount.get(52), ageCount.get(53), ageCount.get(54), ageCount.get(55), ageCount.get(55.5), ageCount.get(56), ageCount.get(57), ageCount.get(58), ageCount.get(59), ageCount.get(60), ageCount.get(61), ageCount.get(62),  ageCount.get(63), ageCount.get(64), ageCount.get(65), ageCount.get(66), ageCount.get(70), ageCount.get(70.5), ageCount.get(71), ageCount.get(74), ageCount.get(80)]

fig, ax = plt.subplots()

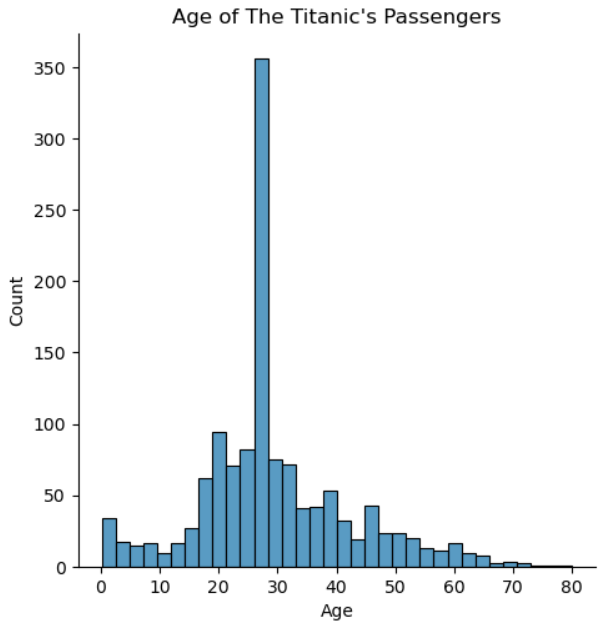
ax.plot(ageLabels, sizes)

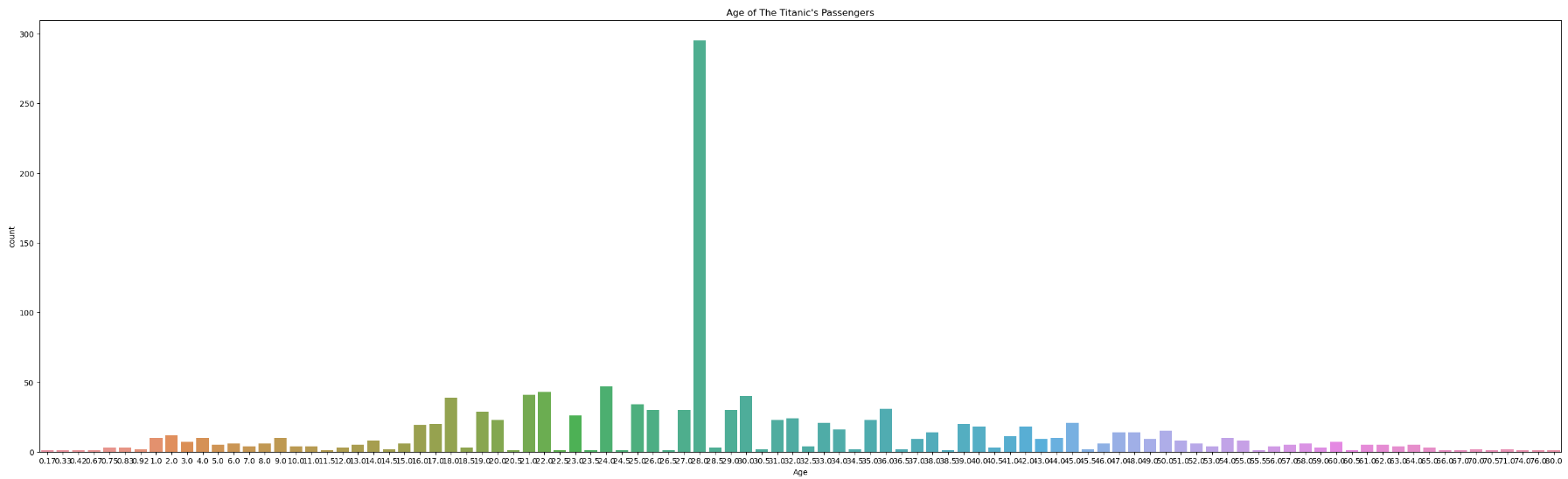
ax.set\_ylabel('Number of passengers')

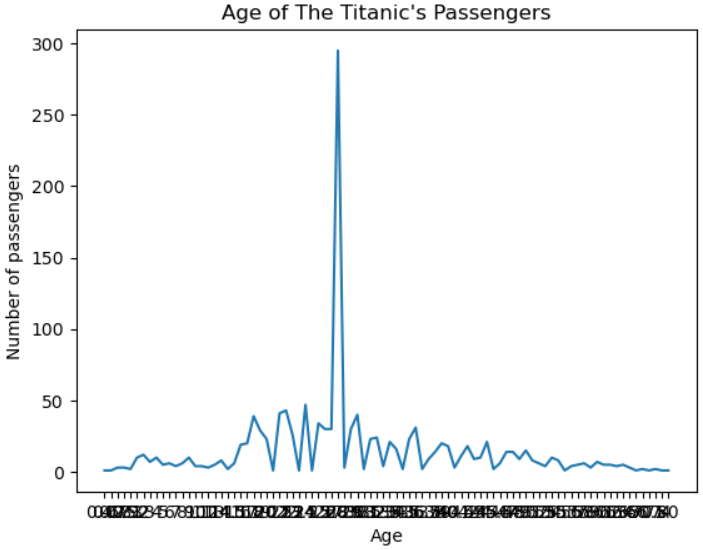
ax.set\_xlabel('Age')

ax.set\_title('Age of The Titanic\'s Passengers')

plt.show()





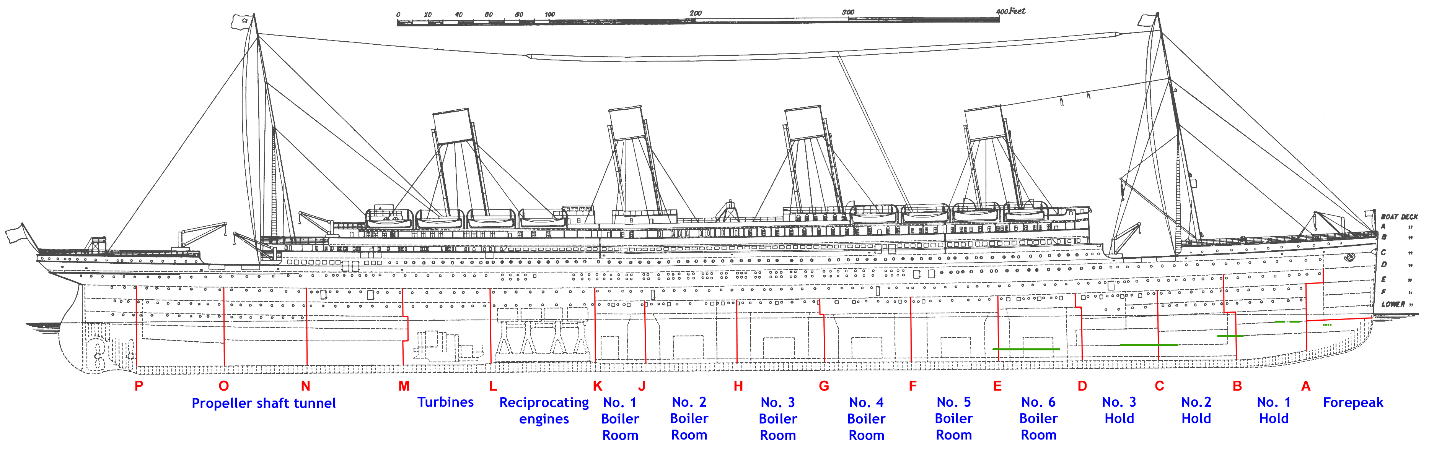


The distribution is still right-skewed, but more symmetric than before the imputation. Line graphs are another convenient way to check a graph’s skeweness.

*Cabin*’s values are named “A20”, “B24”, “C70” etc. We transform them into plain “A”, “B” etc. and save them in a new column named *CabinOnlyLetter*, because the numerical part is irrelevant; as in every ship, deck A of the Titanic offered the best accommodation and was located the farthest from the engines’ noise and tremor, with the rest of the decks/cabins continuing downwards in alphabetical order (Wikipedia, 2023b). By knowing a cabin’s category, we also know the part of the ship it was located at. Additionally, dealing with fewer values (data reduction) will assist in handling and visualizing our data. Nevertheless, it still constitutes a form of data loss and to ensure the transformation does not affect the dataset, we would run next phase’s ML algorithm(s) once using the old (*Cabin*) and once the new values (*CabinOnlyLetter*).

#Splits cabin number values, keeps only the first letter and saves them in a new column.

combinedData['CabinOnlyLetter'] = combinedData['Cabin'].str.split('', expand = True)[1]

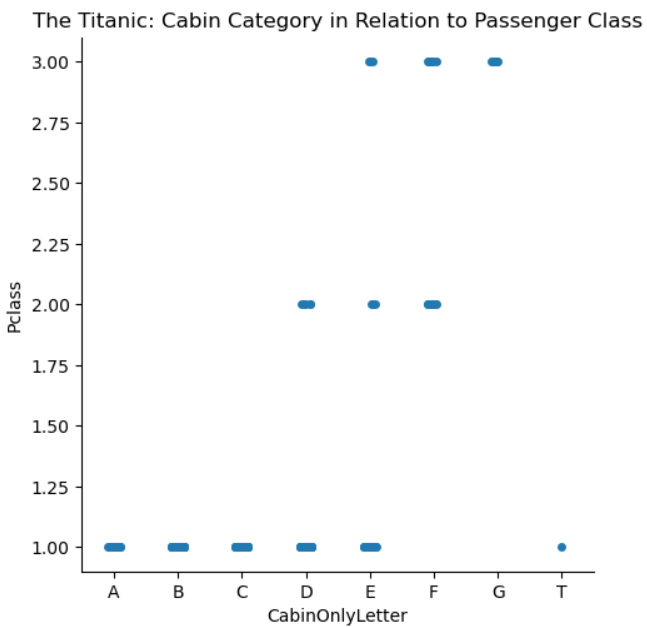


**Figure 3**: Location of The Titanic’s decks (Source: Wikipedia, 2023a)

Relationships between two or more values can be clearly displayed using strip plots. Here is one visualizing ***CabinOnlyLetter*** **and** ***Pclass*** (Passenger Class):

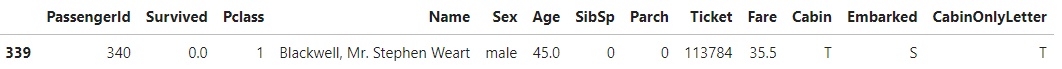
#Strip plot of CabinOnlyLetter and Pclass columns.

sns.catplot(data = combinedData, x = "CabinOnlyLetter", y = "Pclass", order = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'T']).set(title = 'The Titanic: Cabin Category in Relation to Passenger Class')



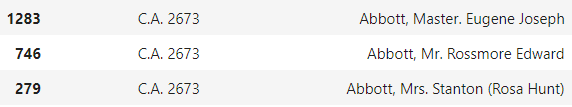
Contrary to expectations, 1st-class cabins were not restricted to the ship’s higher parts.

The **cabin in Deck T**, was occupied by a 1st Class passenger and most probably belonged to the Boat Deck (Wikipedia, 2023b), which was located above Deck A. Because “T” Deck does not appear in Wikipedia’s website, our source of information about the ship’s cabin categories, we are going to change its category into A:



As there exist cabin categories that are common in more than one passenger class (D, E and F), imputing the former with the help of the latter is unfeasible. We, instead, display ***Ticket*** **and** ***CabinOnlyLetter*** and ***Ticket* and *Name*** and consult them:



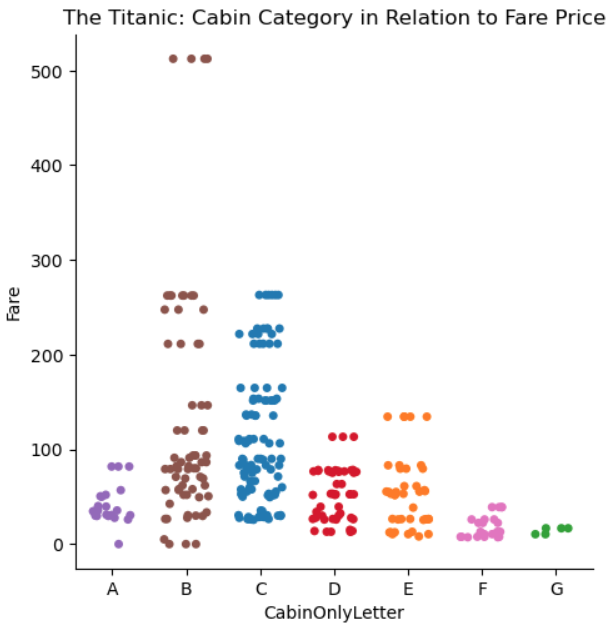


This allows us to replace some of the missing values, reducing their number to 998, because most family members were in possession of identically-numbered tickets and logic dictates that identically-numbered tickets belonged to the same cabin category/deck.

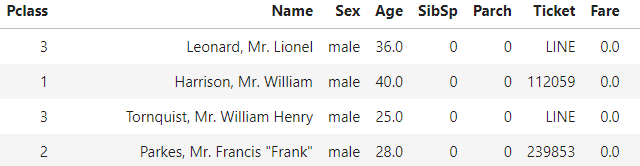
Continuing, we create a strip plot of ***CabinOnlyLetter*** **and** ***Fare***:

#Strip plot of CabinOnlyLetter and Fare columns.

sns.catplot(data = combinedData, x = "CabinOnlyLetter", y = "Fare", jitter = 0.3, hue = "CabinOnlyLetter", order = ['A', 'B', 'C', 'D', 'E', 'F', 'G']).set(title = 'The Titanic: Cabin Category in Relation to Fare Price')



There were cheap tickets available for every cabin category, regardless of the passenger’s class. The £500+ fares are way more expensive than all others but it is known that The Titanic was a very luxurious ship and that some of its passengers were among the richest people alive (ibid.), so we do not mark them as outliers. Similarly, zero-cost tickets could have been won as a prize and their value will not be altered – there is at least one such ticket in every passenger class, as can be seen below. The low cost of all A-category tickets is the only set of values we consider unusual.

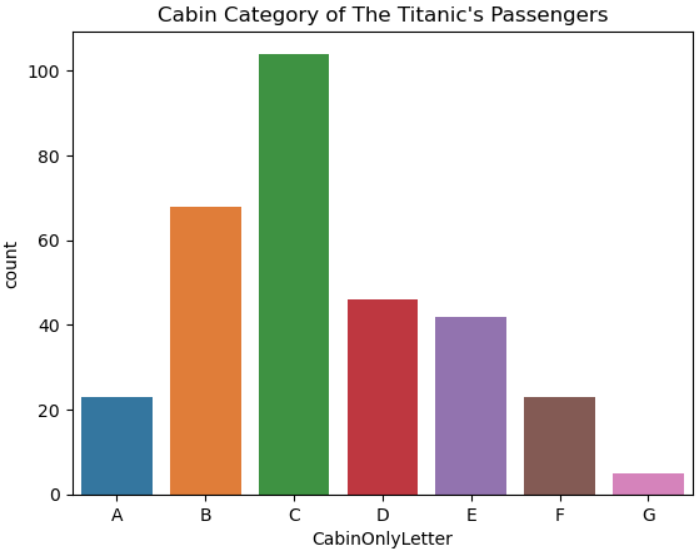


As in the *Cabin*-*Pclass* combination, we cannot impute any of the remaining missing *Cabin* values, because no cabin category contains fare prices exclusive to it.

The following step is to create a count plot of ***CabinOnlyLetter***, to form an initial opinion about it:

#Count plot of CabinOnlyLetter column.

sns.countplot(x = combinedData["CabinOnlyLetter"], order = ['A', 'B', 'C', 'D', 'E', 'F', 'G']).set(title = 'Cabin Category of The Titanic\'s Passengers')



Most of the cabins are in B and C decks, which means that the majority of missing values is from passenger classes 2 and 3 and decks/cabins D through G, as we are about to confirm.

A good way to estimate missing categorical data is the mode value, the most frequently-occurring value in a set (Myatt and Johnson, 2014). We find that “C” is the *CabinOnlyLetter* mode of Class 1 passengers and use it, to replace some of the missing values.

The *CabinOnlyLetter* mode of Class 2 and 3 passengers is the same: F. We believe that the fact that the missing *CabinOnlyLetter* values are so numerous (254 out of 277 in Class 2 and 693 out of 709 in Class 3) renders the mode of the existing ones unreliable and so decide to alternately fill the missing values with each cabin letter belonging to each class (D, E and F for Class 2 and E, F and G for Class 3), lest we create a *CabinOnlyLetter* column extremely biased towards “F”. As before, in a real-life scenario we would run next phase’s ML algorithms once for every situation and then decide which choice is the best: 1) Imputing only the values related to *Ticket*, 2) Imputing only the values related to *Ticket* and passenger class “1”, 3) Imputing all values as we do in the current situation and 4) Imputing the values related to *Ticket* and passenger class 1 and replacing all Class 2 and 3 values with F.

Changes the CabinOnlyLetter value of null rows with Pclass == 2.

combinedData.iat[9,12] = 'D'

combinedData.iat[15,12] = 'E'

combinedData.iat[17,12] = 'F'

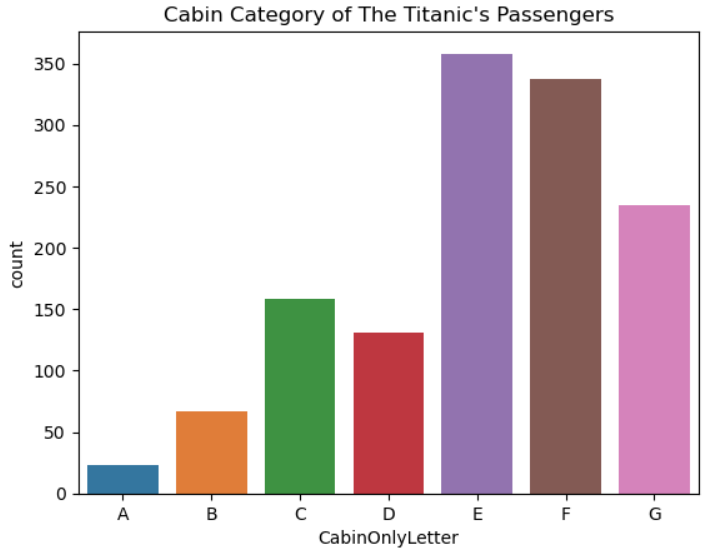
combinedData.iat[20,12] = 'D'

[... ... ...]

There are, currently, no missing values in ***CabinOnlyLetter*** and its updated count plot appears as follows:

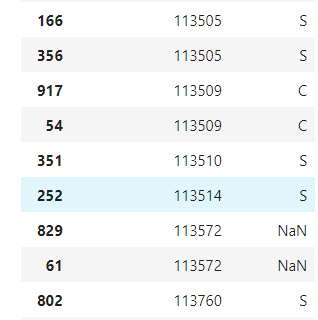
#Count plot of CabinOnlyLetter column.

sns.countplot(x = combinedData["CabinOnlyLetter"], order = ['A', 'B', 'C', 'D', 'E', 'F', 'G']).set(title = 'Cabin Category of The Titanic\'s Passengers')



As was expected, cabins in Decks E, F and G (3rd Class) were the most numerous.

Moving on to ***Embarked***, we try estimating its 2 missing values by crosschecking them with ***Ticket***, but they both belong to the same ticket number:



Moreover, we cannot impute them by taking into account the rest of the tickets starting with “113”, because they too have different values. Hence, we replace them with ***Embarked***’s mode: “S”.

We, next, visualize a bar chart, which has the same functionality as a count plot. Both are convenient, when counting a small number of unique variables:

#Bar chart of Embarked column.

embarkedCount = pd.Series(combinedData.Embarked).value\_counts()

fromSouthampton = embarkedCount.get('S')

fromQueenstown = embarkedCount.get('Q')

fromCherbourg = embarkedCount.get('C')

embarkedLabels = ['Southampton, England', 'Queenstown, Ireland', 'Cherbourg, France']

sizes = [fromSouthampton, fromQueenstown, fromCherbourg]

bar\_colors = ['royalblue', '#45cea2', 'lightgray']

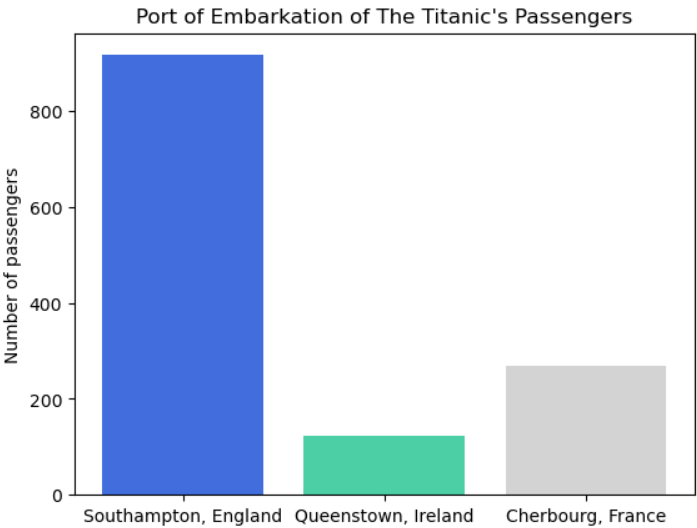
fig, ax = plt.subplots()

ax.bar(embarkedLabels, sizes, color = bar\_colors)

ax.set\_ylabel('Number of passengers')

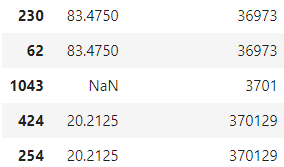
ax.set\_title('Port of Embarkation of The Titanic\'s Passengers')

plt.show()

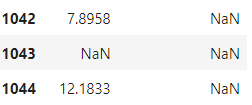


As we can see, almost all passengers embarked at Southampton, The Titanic’s departure port.

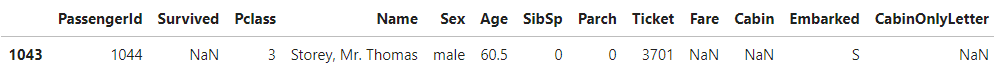
To find the missing *Fare* value, we display ***Fare*** **and** ***Ticket***, but the fare belongs to a ticket number that is unique, meaning that we cannot impute it using this method:



We try ***Fare*** **and** ***CabinOnlyLetter***, but the result is the same, since both columns are null:



We, finally, display all of the ***Fare* missing entry**’s attributes:



His passenger class is 3, so we can impute his fare price by examining the rest of the **Class 3 fare prices**. To do this, we first have to check their distribution. We plot both a histogram (that bins values) and a count plot (that lists every value), to ensure we judge correctly:

#Creates a new dataset comprised of passengers with third class tickets and displays a histogram of the fare price they paid.

thirdClassPassengers = combinedData[(combinedData.Pclass == 3)]

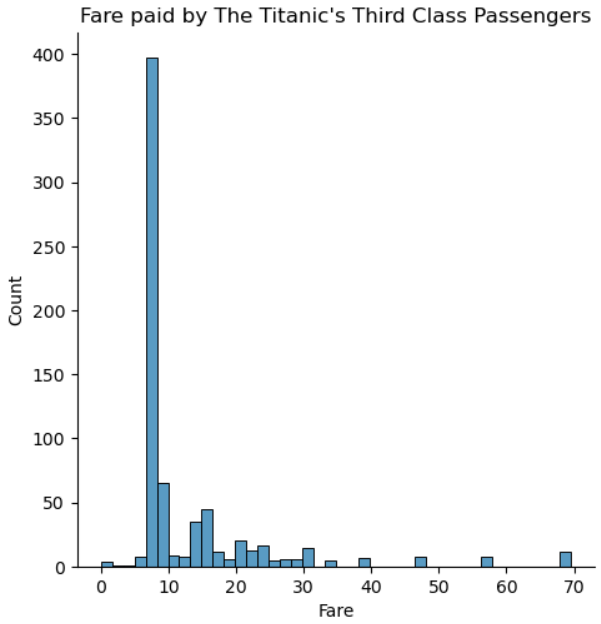
sns.displot(thirdClassPassengers['Fare']).set(title = 'Fare paid by The Titanic\'s Third Class Passengers')

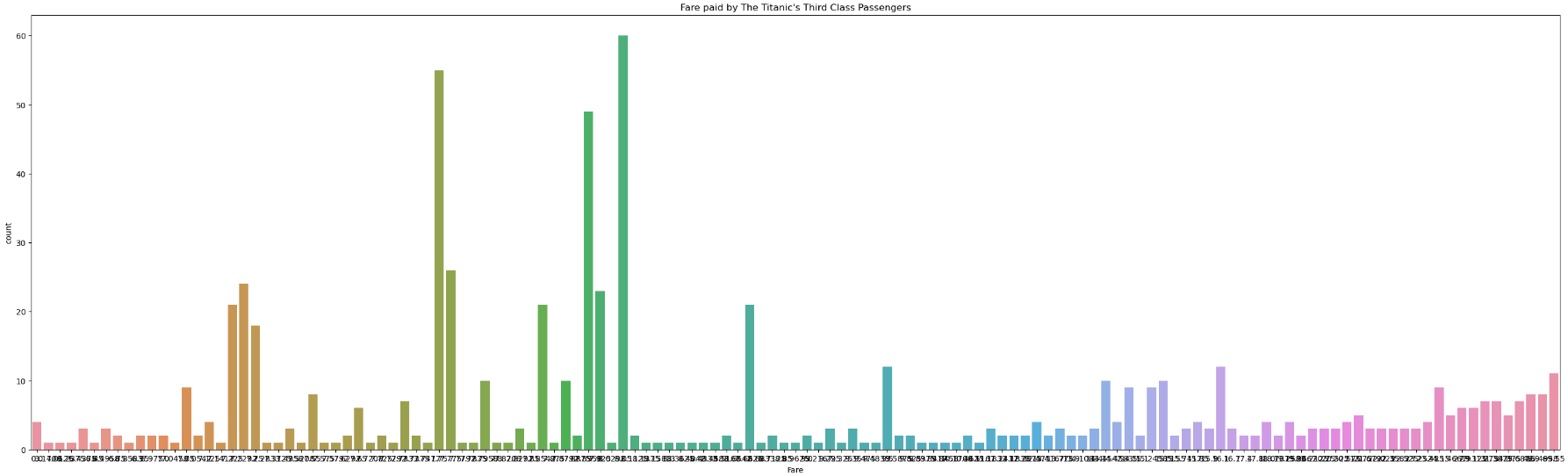
plt.show()

#Count plot of the fare paid by travel class 3 passengers.

plt.figure(figsize = (35, 10))

sns.countplot(x = thirdClassPassengers["Fare"]).set(title = 'Fare paid by The Titanic\'s Third Class Passengers')





They are both right-skewed, so we replace the missing value with the median of Class 3 passengers’ fare price.

We can, now, plot ***Fare*** as a whole. We create a histogram, a strip plot and a box plot:

#Histogram of Fare column.

sns.displot(combinedData['Fare']).set(title = 'Fare paid by The Titanic\'s Passengers')

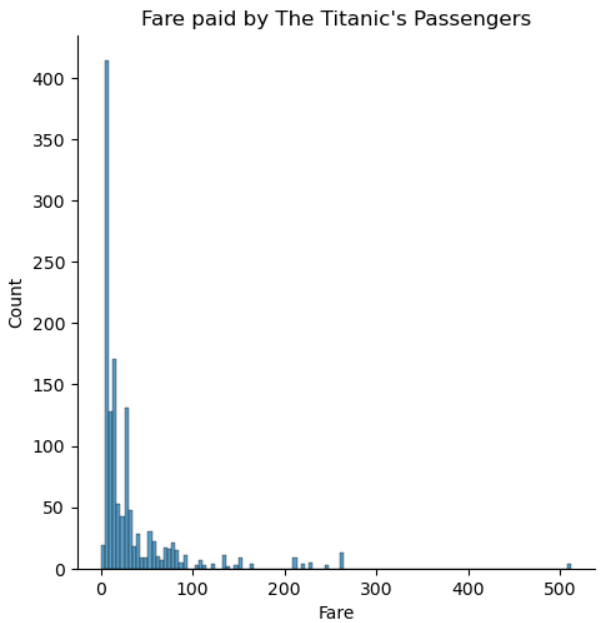
plt.show()

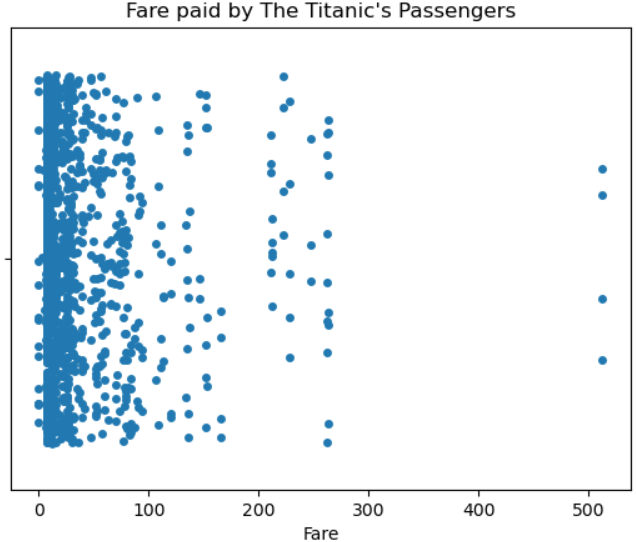
#Strip plot of Fare column.

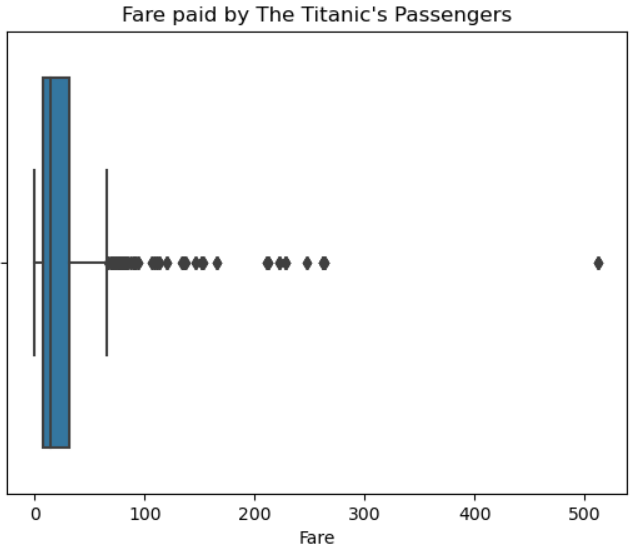
sns.stripplot(data = combinedData, x = "Fare", jitter = 0.3)

#Box plot of the Fare column.

sns.boxplot(x = combinedData["Fare"])







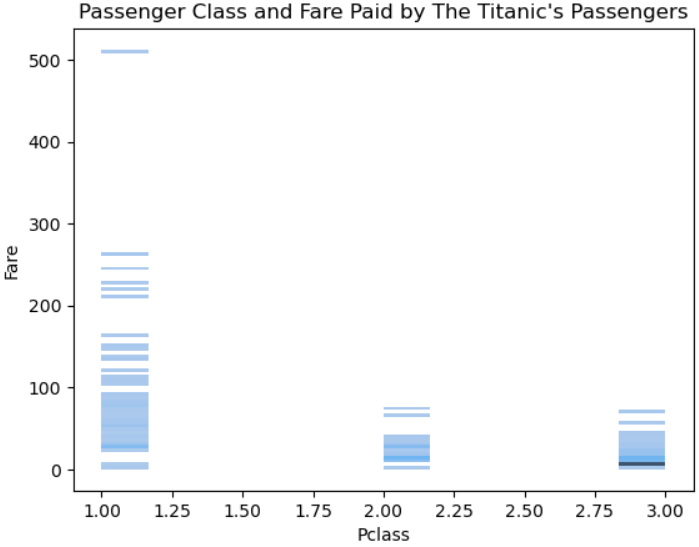
Box plots are very useful, because they can display many pieces of information at once: the minimum and maximum values, or “range”, which are the leftmost vertical line and the rightmost dot respectively, the median value (the black line inside the box), outliers (all dots after the last right vertical line) and all four of the visualized dataset’s quartiles.

The histogram shows us that the majority of the passengers paid a small fare and the strip plot confirms this in an even clearer way. The box plot shows us that the median is very low, resulting in many values being characterized as outliers. The first three quartiles’ values are very close to the median and the last quartile is the most outspread. This occurs because, as we were also able to see in the *CabinOnlyLetter* and *Fare* strip plot earlier, all eight of the cabin categories offered cheap accommodation, meaning that the majority of the passengers did not pay an expensive fare.

A histogram of ***Fare* and *Pclass*** shows us that only 1st Class passenger classes affected fare price. Besides this, every class included cheap ticket prices, as already stated:

#Histogram of Pclass and Fare columns.

sns.histplot(data = combinedData, x = "Pclass", y = "Fare").set(title = 'Passenger Class and Fare Paid by The Titanic\'s Passengers')

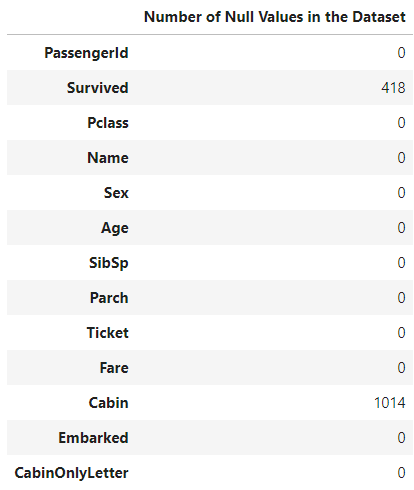


Examining ***Fare*, *Cabin* and *Ticket*** (displayed below), we noticed possibly noisy entries of people that paid less than others for the exact same cabin numbers (e.g. Ticket #11770 paid 25.7 for cabin C101 and Ticket #11769 paid 51.4), but the fact that the instances of different ticket numbers corresponding to the same cabin are too many makes us wonder if smoothing them is prudent/correct or not. We decide to use them as they are but, as stated previously, in a real situation we would execute the learning algorithm(s) once with their values as they are and once after smoothing them by using the average of their immediately neighbouring values.

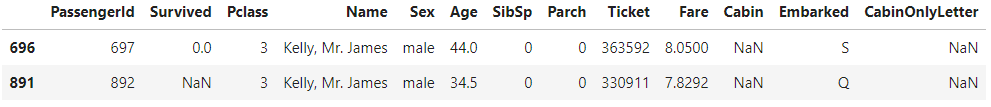


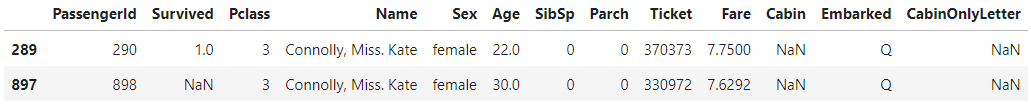


Having finished with imputing all missing data, we can display our dataset’s **null** values one last time:



We now turn to the potentially noisy ***Name*** values. We detect the **duplicate entries** and display them:





The second set’s names have more common features than those of the first one’s and draw the most chances of one of them being a false entry. Despite this, we decide to ignore all four, because we do not have any strong indication that they are false indeed and it would not be impossible for two sets out of ~2.200 people to share the same name.

***Pclass*** and most of the remaining attributes can be depicted in more than one ways, because they contain few unique values. We create a bar chart and a pie chart, which displays every value proportionally to the column’s total:

#Bar chart of Pclass column (passenger travel class).

travelClassCount = pd.Series(combinedData.Pclass).value\_counts()

firstClass = travelClassCount.get(1)

secondClass = travelClassCount.get(2)

thirdClass = travelClassCount.get(3)

travelClassLabels = ['First Class', 'Second Class', 'Third Class']

sizes = [firstClass, secondClass, thirdClass]

bar\_colors = ['gold', 'silver', '#cd7f32']

fig, ax = plt.subplots()

ax.bar(travelClassLabels, sizes, color = bar\_colors)

ax.set\_ylabel('Number of passengers')

ax.set\_title('Travel Class of The Titanic\'s Passengers')

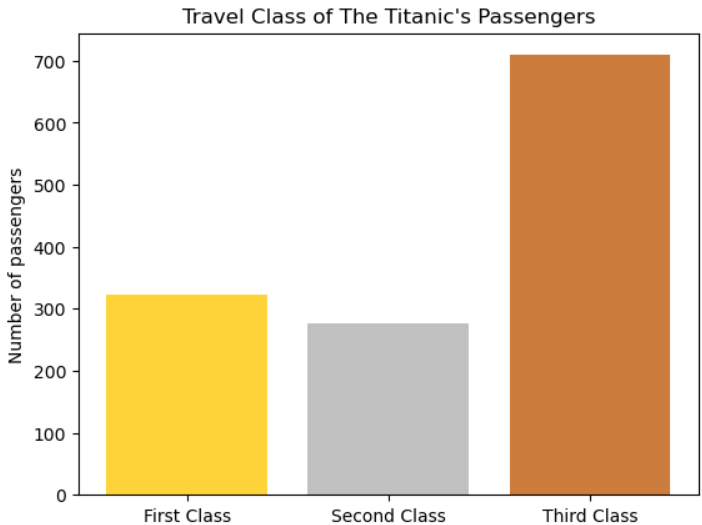
plt.show()

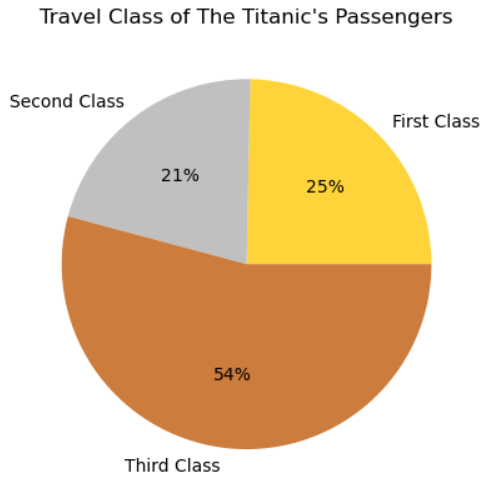
#Pie chart of Pclass column (passenger travel class).

fig, ax = plt.subplots()

ax.set\_title("Travel Class of The Titanic\'s Passengers")

ax.pie(sizes, labels=travelClassLabels, colors = bar\_colors, autopct='%1.0f%%')





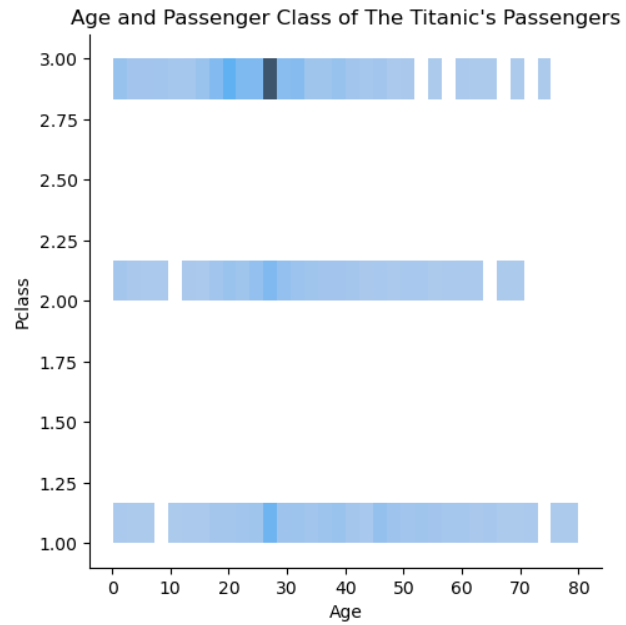
3rd Class passengers reasonably occupied half of the total number of travelers, as *CabinOnlyLetter*’s graph also showed us. Although all classes offered inexpensive tickets, these graphs also validate *Fare*’s plots, 3rd Class tickets being cheaper than their 2nd and 1st Class counterparts. 1st and 2nd Class passengers each comprise roughly one quarter of the total number.

The passengers’ age could help us find out the number of older people that were in the lower decks, away from the lifeboats, which were on Deck A (Wikipedia, 2023b). We use a histogram of ***Age*** **and** ***Pclass***, to investigate it:

#Histogram of Age and Pclass columns.

sns.displot(data = combinedData, x = "Age", y = "Pclass").set(title = 'Age and Passenger Class of The Titanic\'s Passengers')

plt.show()



People aged 60 or more were included in all three passenger classes, so no insight is gained.

***Sex*** is the next attribute to be visualized:

#Pie chart of Sex column.

sexCount = pd.Series(combinedData.Sex).value\_counts()

malePassengers = sexCount.get('male')

femalePassengers = sexCount.get('female')

sexLabels = 'Men', 'Women'

sizes = [malePassengers, femalePassengers]

pieColors = ['dodgerblue', 'hotpink']

fig, ax = plt.subplots()

ax.set\_title("Sex of The Titanic\'s Passengers")

ax.pie(sizes, labels = sexLabels, autopct = '%1.1f%%', colors = pieColors)

#Bar chart of Sex column.

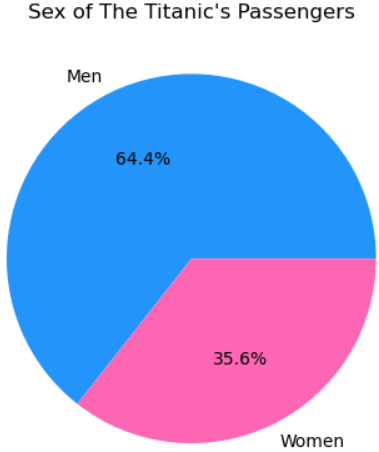
fig, ax = plt.subplots()

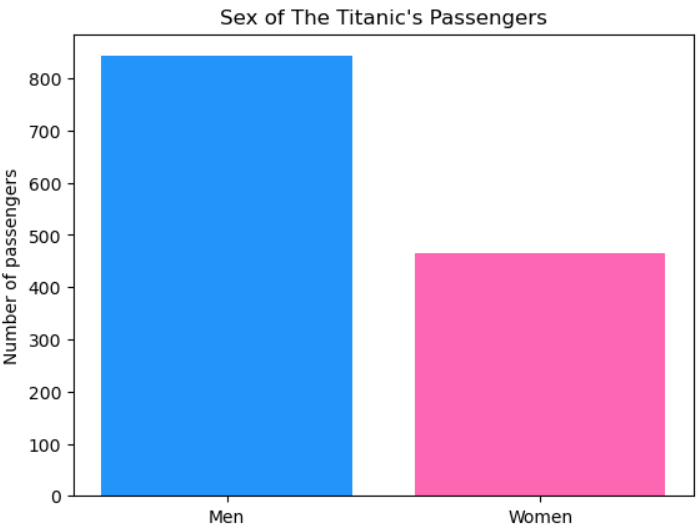
ax.bar(sexLabels, sizes, color=pieColors)

ax.set\_ylabel('Number of passengers')

ax.set\_title('Sex of The Titanic\'s Passengers')

plt.show()





A little below two thirds of the passengers were male; they were double the women.

***SibSp*** denotes the number of siblings and/or spouses that followed each passenger:

#Pie chart of SibSp column (number of siblings and spouses also aboard).

siblingsAndSpousesCount = pd.Series(combinedData.SibSp).value\_counts()

zeroSiblingsAndSpouses = siblingsAndSpousesCount.get(0)

oneSiblingAndSpouse = siblingsAndSpousesCount.get(1)

twoSiblingsAndSpouses = siblingsAndSpousesCount.get(2)

threeSiblingsAndSpouses = siblingsAndSpousesCount.get(3)

fourSiblingsAndSpouses = siblingsAndSpousesCount.get(4)

fiveSiblingsAndSpouses = siblingsAndSpousesCount.get(5)

eightSiblingsAndSpouses = siblingsAndSpousesCount.get(8)

siblingsAndSpousesLabels = 'None', 'One', 'Two', 'Three', 'Four', 'Five', 'Eight'

sizes = [zeroSiblingsAndSpouses, oneSiblingAndSpouse, twoSiblingsAndSpouses, threeSiblingsAndSpouses, fourSiblingsAndSpouses, fiveSiblingsAndSpouses, eightSiblingsAndSpouses]

explode = (0, 0, 0, 0, 0.3, 0.6, 0.9)

fig, ax = plt.subplots()

ax.set\_title("Number of Passengers\' Siblings and/or Spouses also Aboard The Titanic")

ax.pie(sizes, labels = siblingsAndSpousesLabels, autopct = '%1.0f%%', explode = explode)

#Bar chart of SibSp column (number of siblings and spouses also aboard).

barColors = ['blue', 'black', 'yellow', 'red', 'brown', 'green', 'purple']

fig, ax = plt.subplots()

ax.bar(siblingsAndSpousesLabels, sizes, color = barColors)

ax.set\_xlabel('Number of siblings/spouses')

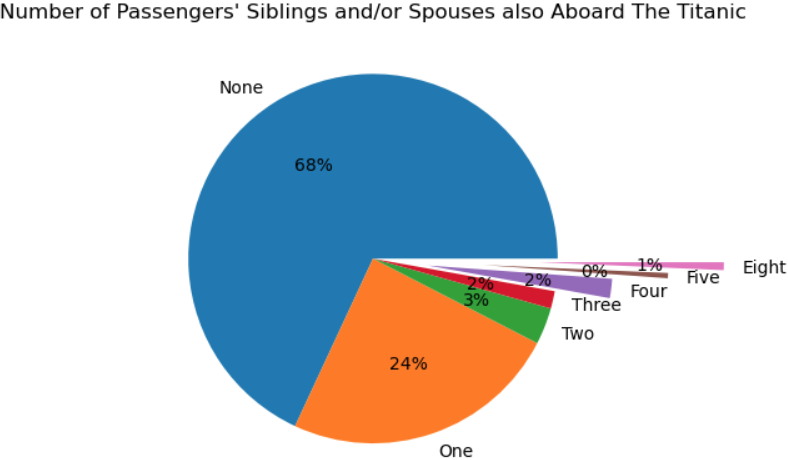
ax.set\_ylabel('Number of passengers')

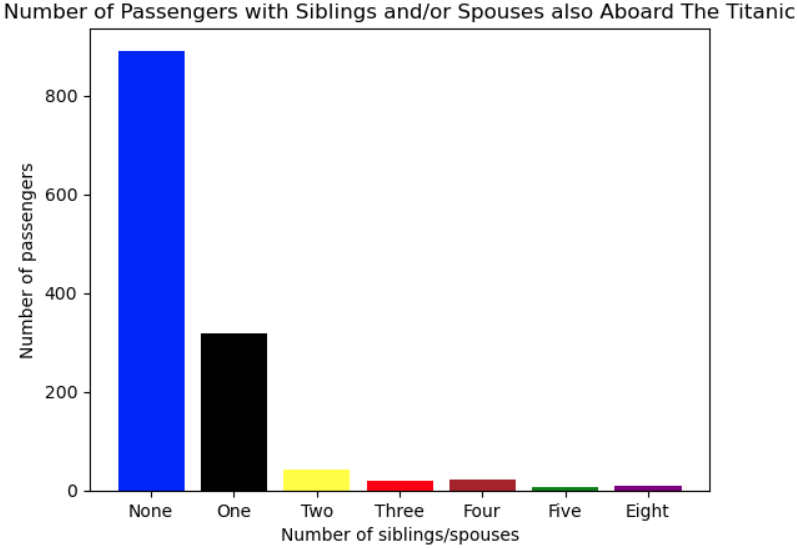
ax.set\_title('Number of Passengers with Siblings and/or Spouses also Aboard The Titanic')

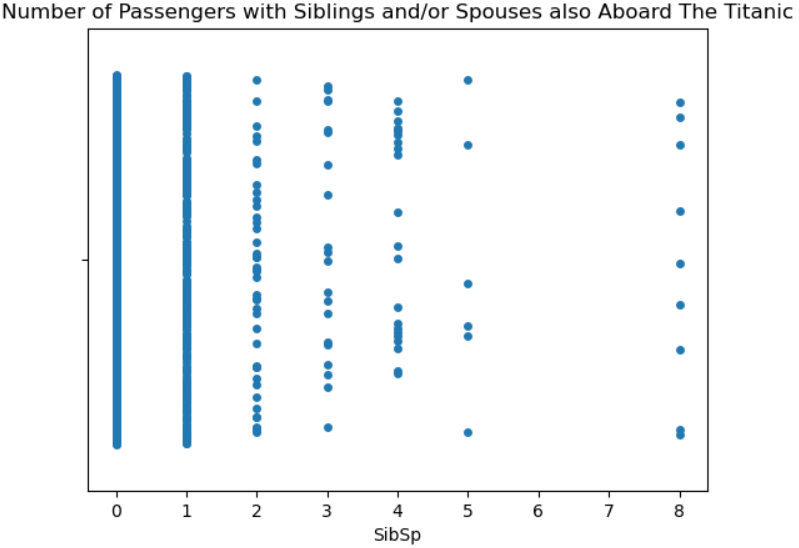
plt.show()

#Strip plot of SibSp column (number of siblings and spouses also aboard).

sns.stripplot(data = combinedData, x = "SibSp", jitter = 0.4).set(title = 'Number of Passengers with Siblings and/or Spouses also Aboard The Titanic')







The passengers’ bulk travelled without any siblings and spouses (or was just single) and a quarter of them were accompanied by one sibling/spouse. The rest cases, displayed more clearly in the strip plot, form only the 7.6% of the total.

***Parch*** informs us about the number of parents and/or children escorting The Titanic’s passengers:

#Bar chart of Parch column (parents and children).

parentOrChildCount = pd.Series(combinedData.Parch).value\_counts()

zeroParentsOrChildren = parentOrChildCount.get(0)

oneParentOrChild = parentOrChildCount.get(1)

twoParentsOrChildren = parentOrChildCount.get(2)

threeParentsOrChildren = parentOrChildCount.get(3)

fourParentsOrChildren = parentOrChildCount.get(4)

fiveParentsOrChildren = parentOrChildCount.get(5)

sixParentsOrChildren = parentOrChildCount.get(6)

nineParentsOrChildren = parentOrChildCount.get(9)

parentOrChildLabels = 'None', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Nine'

sizes = [zeroParentsOrChildren, oneParentOrChild, twoParentsOrChildren, threeParentsOrChildren, fourParentsOrChildren, fiveParentsOrChildren, sixParentsOrChildren, nineParentsOrChildren]

barColors = ['pink', 'black', 'yellow', 'red', 'brown', 'green', 'purple', 'blue']

fig, ax = plt.subplots()

ax.bar(parentOrChildLabels, sizes, color = barColors)

ax.set\_xlabel('Number of parents/children')

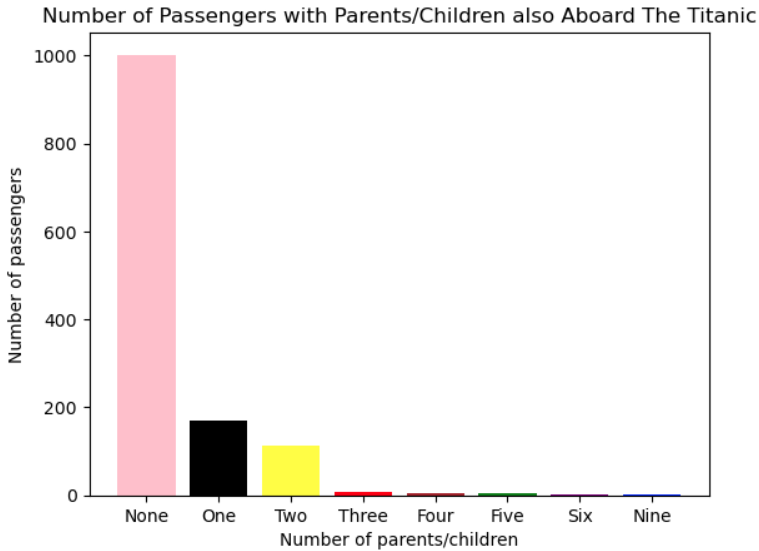
ax.set\_ylabel('Number of passengers')

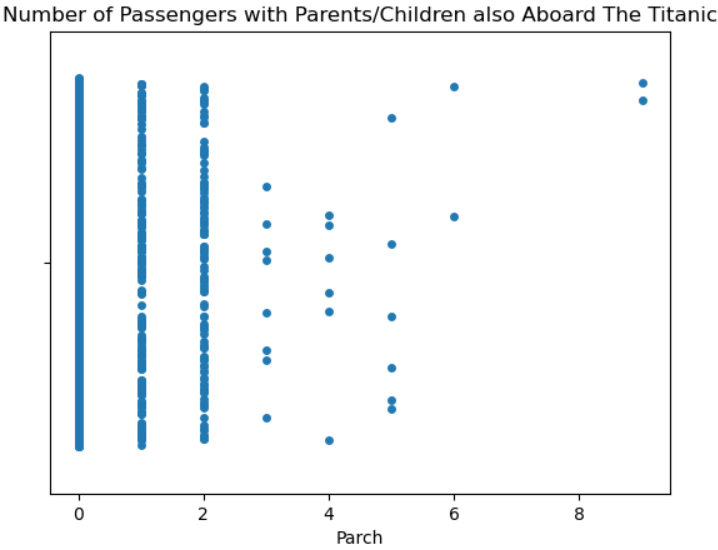
ax.set\_title('Number of Passengers with Parents/Children also Aboard The Titanic')

plt.show()

#Strip plot of Parch column (parents and children).

sns.stripplot(data = combinedData, x = "Parch", jitter = 0.4).set(title = 'Number of Passengers with Parents/Children also Aboard The Titanic')





As before, nearly all of the passengers were travelling by themselves, a few carried one or two parents/children and a very small percentage had three, four, five, six or nine family members also aboard.

***Survived*** exists only in Kaggle’s training dataset, so all graphs making use of it pertain to only 891 of the total 1309 rows. To include it in the test dataset as well, we would need to apply predicting ML algorithms, which is outside the scope of this assignment. Before utilizing it in our visualizations, we replace any missing training dataset values with those that we have already imputed by analyzing the combined dataset.

#Pie chart of Survival column.

survivalCount = pd.Series(trainingData.Survived).value\_counts()

survived = survivalCount.get(1)

died = survivalCount.get(0)

survivalLabels = 'Survived', 'Perished'

sizes = [survived, died]

pieColors = ["green", "red"]

fig, ax = plt.subplots()

ax.set\_title("The Titanic\'s mortality rate")

ax.pie(sizes, labels = survivalLabels, autopct = '%1.1f%%', colors = pieColors)

#Bar chart of Survival column.

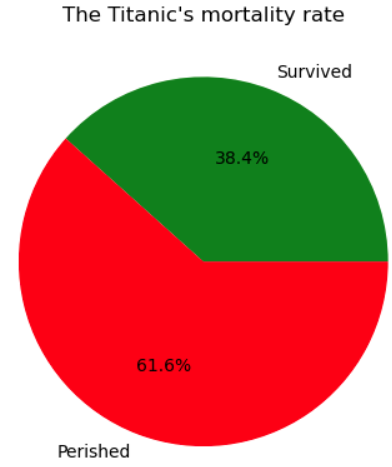
fig, ax = plt.subplots()

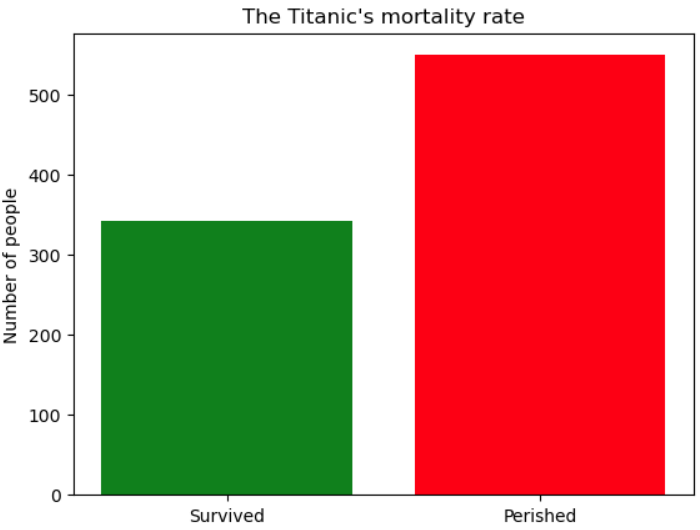
ax.bar(survivalLabels, sizes, color = pieColors)

ax.set\_ylabel('Number of people')

ax.set\_title('The Titanic\'s mortality rate')

plt.show()



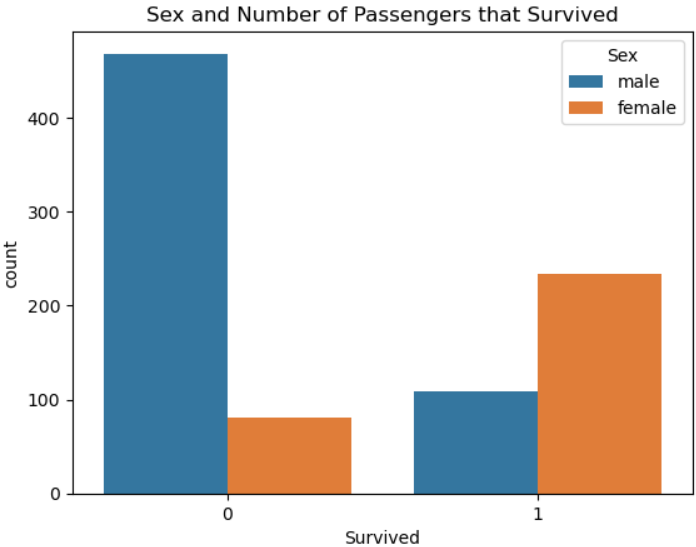


The bar and pie charts show us that more than half of the people died in the accident.

Combining ***Survived* and *Sex*** in a count plot:

#Count plot of the Survived and Sex columns.

sns.countplot(data = trainingData, x = "Survived", hue = "Sex").set(title = 'Sex and Number of Passengers that Survived')



We see something worthy of investigation: more than half of the women survived (“1”) and almost all men died (“0”).

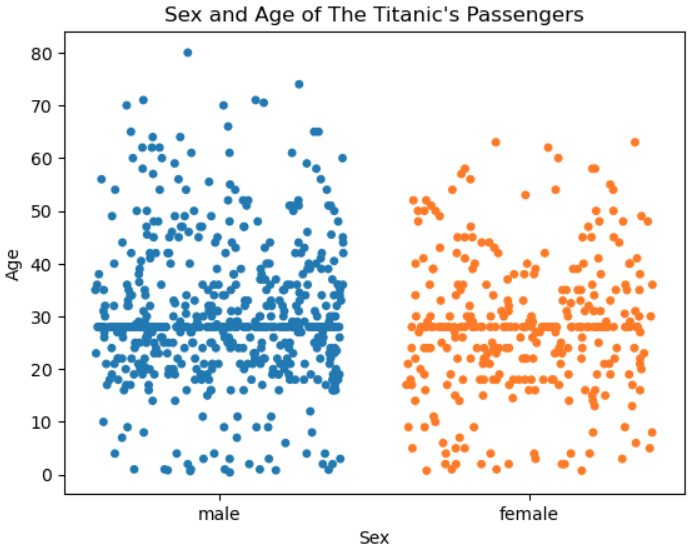
One strip and one box plot of ***Age* and *Sex*** show us that men and women of all ages were aboard The Titanic and most of them were between 20 and 35 years old. Additionally, we can safely assume that the reason more women survived was not that they were younger than men:

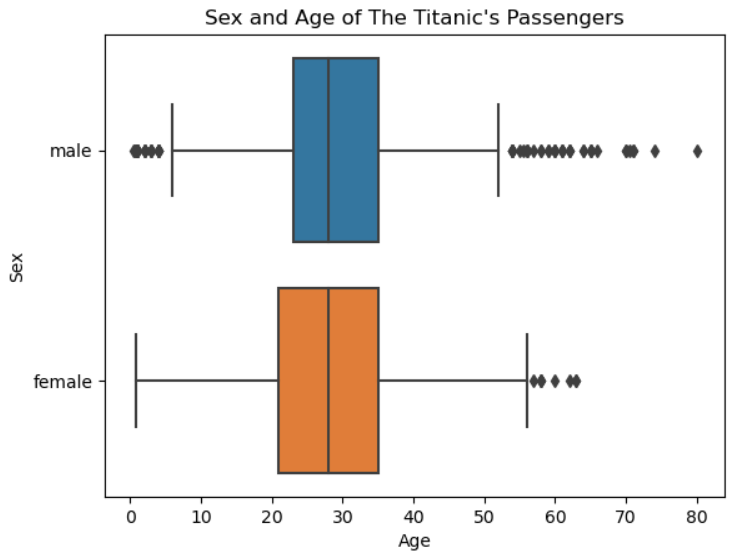
#Strip plot of Age and Sex columns.

sns.stripplot(data = trainingData, x = "Sex", y = "Age", jitter = 0.4, hue = "Sex", legend = False).set(title = 'Sex and Age of The Titanic\'s Passengers')

#Box plot of Age and Sex columns.

sns.boxplot(data = trainingData, x = "Age", y = "Sex").set(title = 'Sex and Age of The Titanic\'s Passengers')





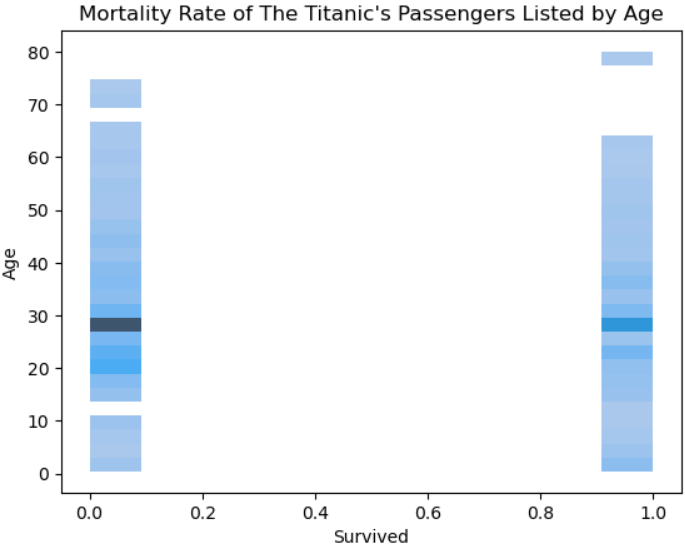
A histogram and a box plot of ***Age* and *Survived*** reveal more info:

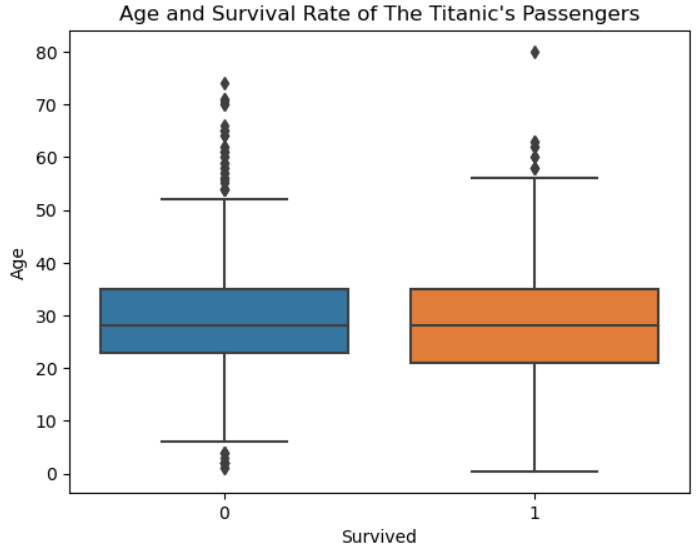
#Histogram of Survived and Age columns.

sns.histplot(data = trainingData, x = "Survived", y = "Age").set(title = 'Mortality Rate of The Titanic\'s Passengers Listed by Age')

#Box plot of Age and Survived columns.

sns.boxplot(data = trainingData, x = "Survived", y = "Age").set(title = 'Age and Survival Rate of The Titanic\'s Passengers')



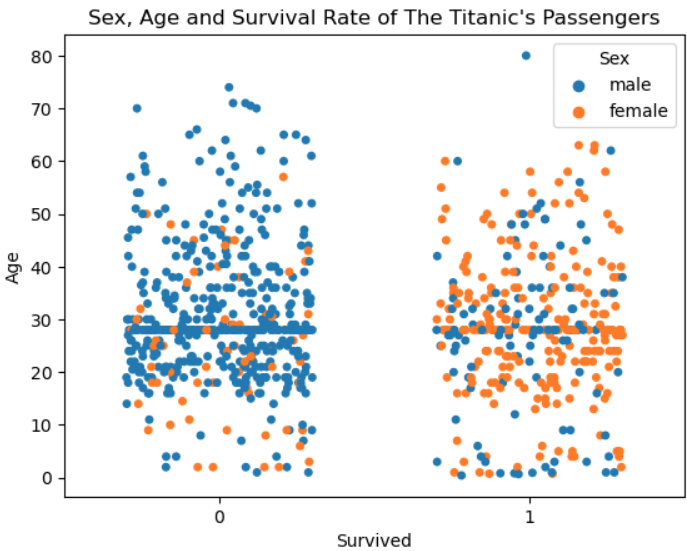


Passengers of all ages exist in both columns, eliminating the possibility of only younger people surviving the sinking. Though there is a gap in survivors aged between 60 and 70, one 80-year-old passenger managed to be saved. Most of both the survivors and the dead were between 20 and 35 years old, conforming to the general *Age* average.

A strip plot of ***Age*, *Sex* and *Survived*** is further proof of everything stated until now:

#Strip plot of Sex, Age and Survived columns.

sns.stripplot(data = trainingData, x = "Survived", y = "Age", jitter = 0.3, hue = "Sex").set(title = 'Sex, Age and Survival Rate of The Titanic\'s Passengers')

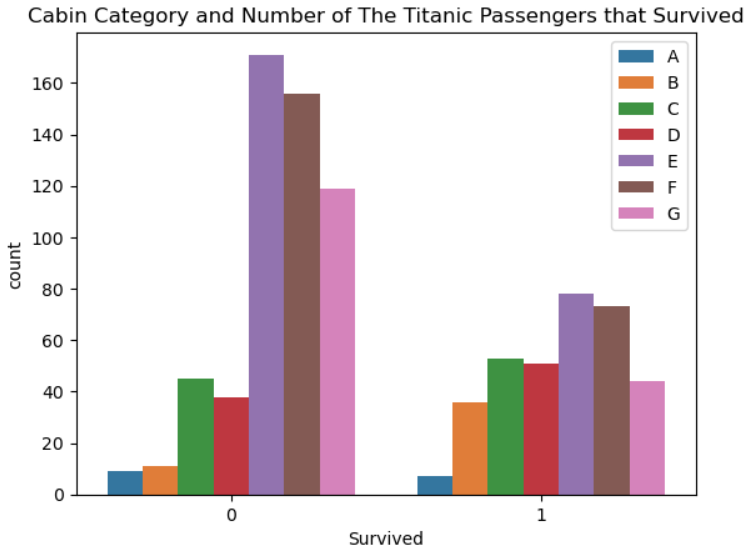


We continue our EDA by creating a count plot of ***CabinOnlyLetter* and *Survived***:

#Count plot of the Survived and CabinOnlyLetter columns.

sns.countplot(data = trainingData, x = "Survived", hue = "CabinOnlyLetter", hue\_order = ["A", "B", "C", "D", "E", "F", "G"]).set(title = 'Cabin Category and Number of The Titanic Passengers that Survived')

plt.legend(loc = 0)

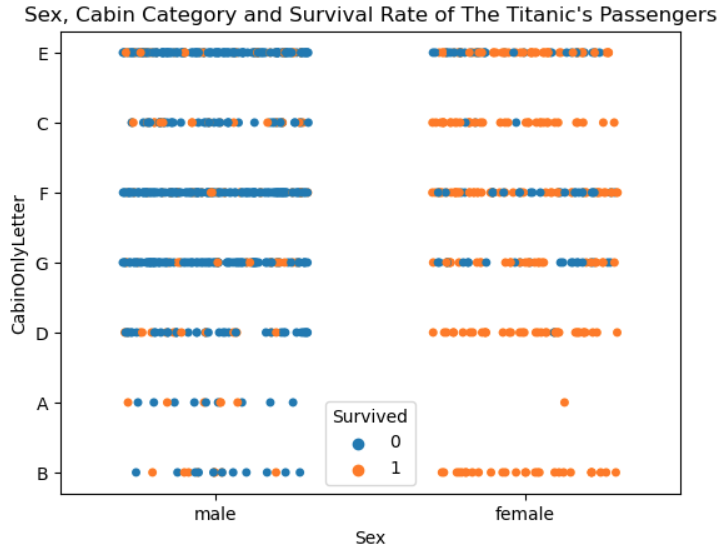


Deck A, C and D occupants have a survival ratio of approximately 50-50. Deck A was the closest to the lifeboats and Deck C was in the upper half of the ship, so the fact that only half of their passengers survived is definitely strange. Deck D passengers also have a 50-50 survival ratio, which is unexpected, given that they were in the middle of the ship and far from the lifeboats. Naturally, all lower-part decks (E, F and G) have more dead than survived people and Deck B the opposite. Apparently, cabin category did not affect passengers’ survival odds exactly the way we expected.

Following this, we create a strip plot of ***Sex*, *CabinOnlyLetter* and *Survived***:

#Strip plot of Sex, Cabin and Survived columns.

sns.stripplot(data = trainingData, x = "Sex", y = "CabinOnlyLetter", jitter = 0.3, hue = "Survived").set(title = 'Sex, Cabin Category and Survival Rate of The Titanic\'s Passengers')

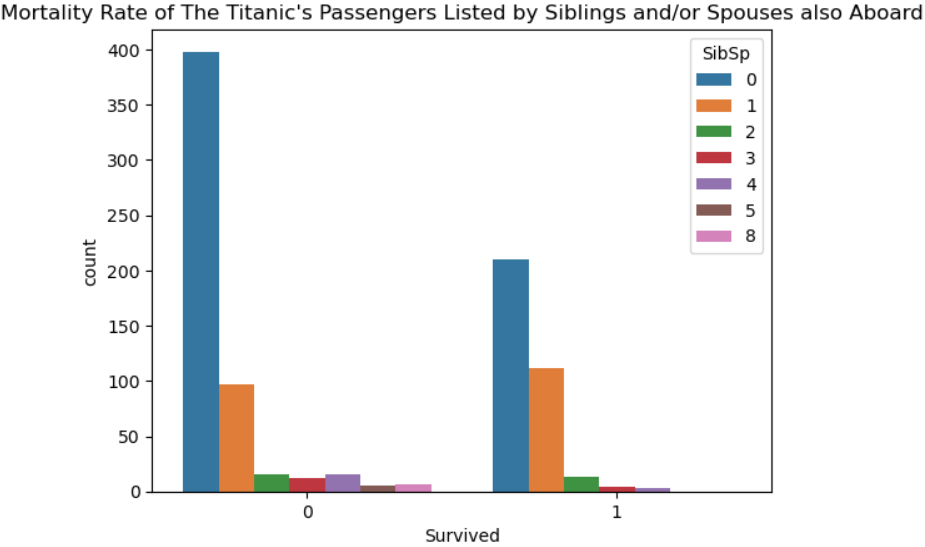


Again, it is evident that nearly all survivors were women and nearly all decedents men. What is more, cabin category did not affect women’s survival chances.

The following step is to display the mortality rate of the passengers that were accompanied by their spouse or siblings. We create a count plot of ***Survived* and *SibSp***:

#Count plot of Survived and SibSp columns (siblings and spouses)

sns.countplot(data = trainingData, x = "Survived", hue = "SibSp").set(title = 'Mortality Rate of The Titanic\'s Passengers Listed by Siblings and/or Spouses also Aboard')

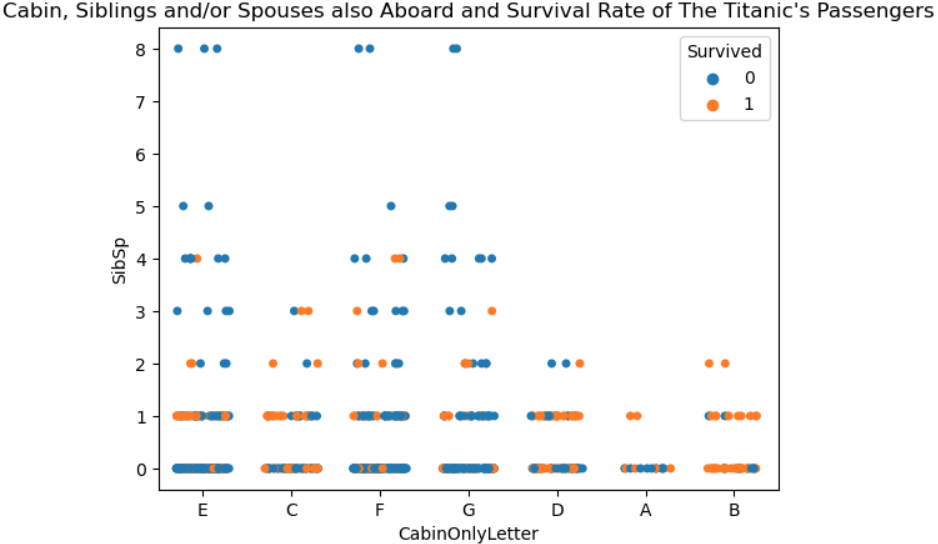


Travelling alone did not raise one’s chances of surviving, as the unmarried persons that died were more than those that survived, which complies with the general survival rate (most people died). All other values are at approximately 50-50 chances, meaning that being only with your spouse or with one or two siblings did not affect your survival odds, but groups of four or more people were less likely to live.

Showing a strip plot of ***Survived*, *CabinOnlyLetter* and *SibSp*** proves that the number of siblings/spouses was a stronger survival factor than cabin category:

#Strip plot of CabinOnlyLetter, SibSp and Survived columns.

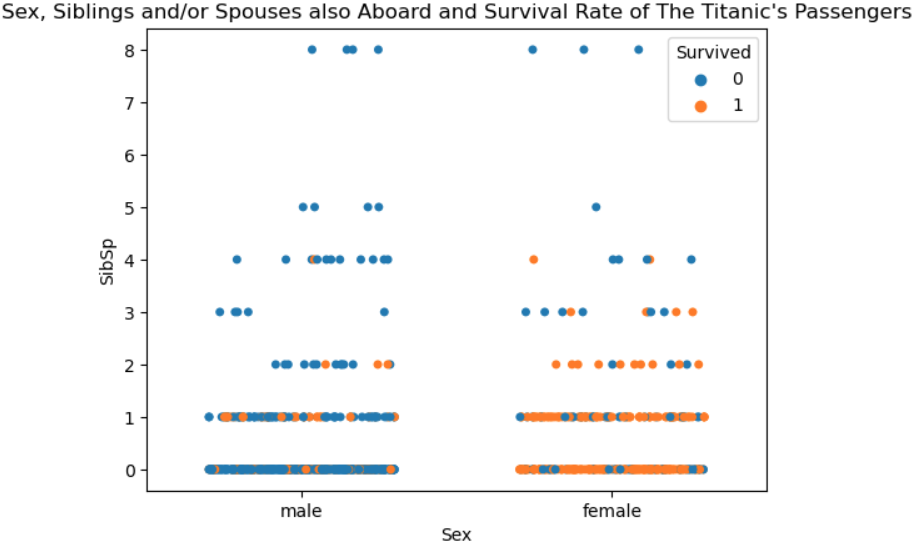
sns.stripplot(data = trainingData, x = "CabinOnlyLetter", y = "SibSp", jitter = 0.3, hue = "Survived").set(title = 'Cabin, Siblings and/or Spouses also Aboard and Survival Rate of The Titanic\'s Passengers')



We return to studying the reasons for the women’s higher survival rate and create another strip plot of ***Sex*, *SibSp* and *Survived***:

#Strip plot of Sex, SibSp and Survived columns.

sns.stripplot(data = trainingData, x = "Sex", y = "SibSp", jitter = 0.3, hue = "Survived").set(title = 'Sex, Siblings and/or Spouses also Aboard and Survival Rate of The Titanic\'s Passengers')

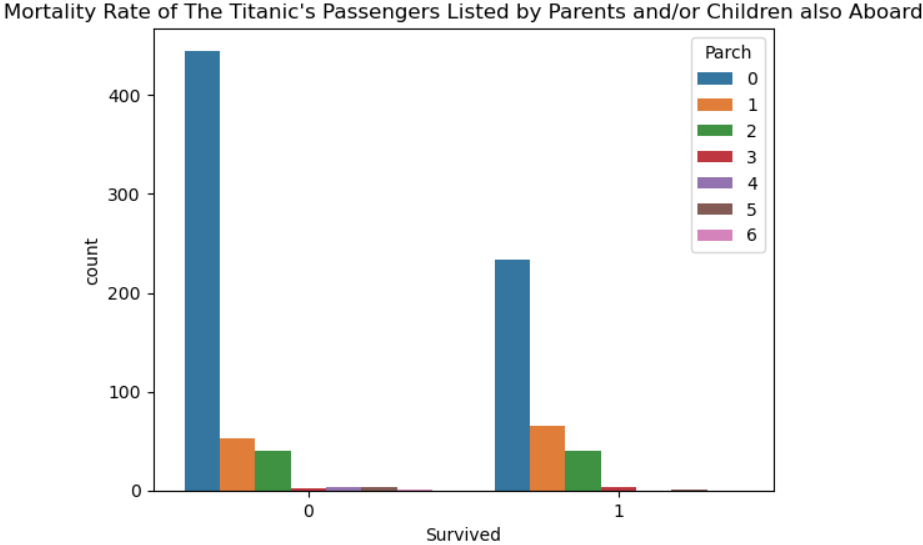


Excluding the last two (5 and 8 spouses/siblings), women survivors surpass men in every other category. Women with 3 or more spouses/siblings had less chances of survival than those with 0 up to 2.

***Parch*** is the only attribute we have not yet combined **with *Survived***:

#Count plot of Survived and Parch columns (parents and children)

sns.countplot(data = trainingData, x = "Survived", hue = "Parch").set(title = 'Mortality Rate of The Titanic\'s Passengers Listed by Parents and/or Children also Aboard')

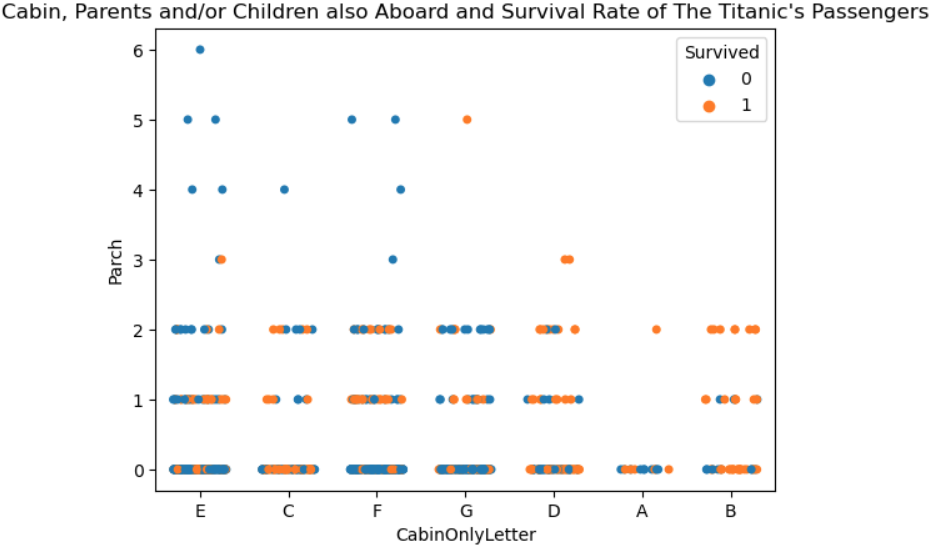


As with *Survived* and *SibSp*, persons travelling alone did not have a higher survival rate and all other categories share 50-50 chances.

The graph of ***Cabin*, *Parch* and *Survived*** is more complicated. People with one parent or child that survived did so regardless of the cabin category they were staying in. The decks with the most survivors are B and C; decks E, F and G, the lowest ones, have the fewest and decks A and D a 50 – 50 ratio. Cabin location might have played a bigger role in this passenger category’s survival:

#Strip plot of CabinOnlyLetter, Parch and Survived columns.

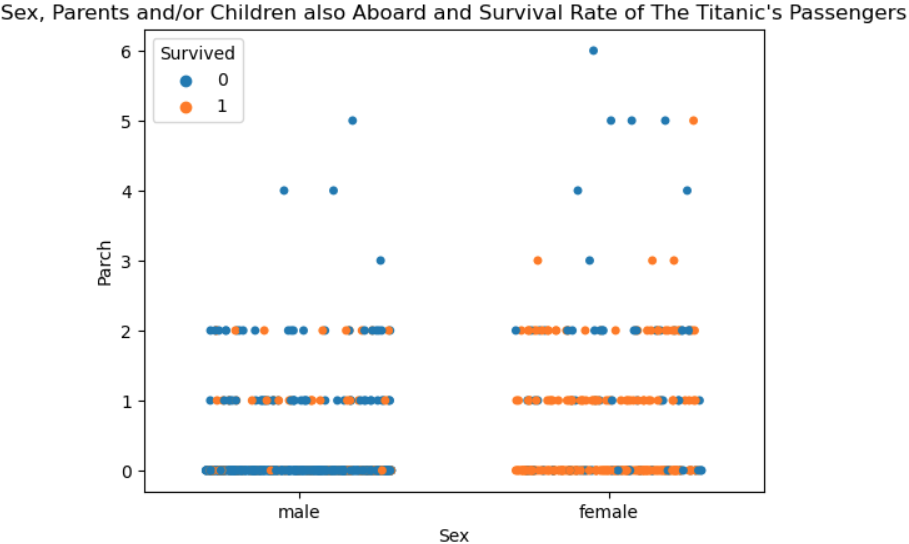
sns.stripplot(data = trainingData, x = "CabinOnlyLetter", y = "Parch", jitter = 0.3, hue = "Survived").set(title = 'Cabin, Parents and/or Children also Aboard and Survival Rate of The Titanic\'s Passengers')



As for women, once more they survived in greater numbers than men (strip plot of ***Sex*, *Parch* and *Survived*** below) and had better chances, the fewer parents/children were travelling with them:

#Strip plot of Survived, Sex and Parch columns (parents and children).

sns.stripplot(data = trainingData, x = "Sex", y = "Parch", jitter = 0.3, hue = "Survived").set(title = 'Sex, Parents and/or Children also Aboard and Survival Rate of The Titanic\'s Passengers')



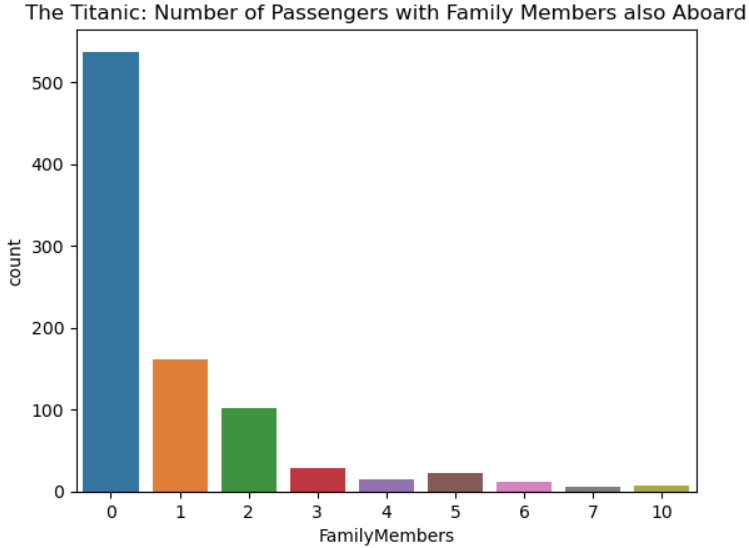
Lastly, we transform *SibSp* and *Parch* by adding their values. The new column, “***FamilyMembers***”, represents the sum of a passenger’s relatives also aboard the ship (count plot):

#Creates column "FamilyMembers" that is the sum of SibSp and Parch.

trainingData['FamilyMembers'] = trainingData['SibSp'] + trainingData['Parch']

#Count plot of FamilyMembers column.

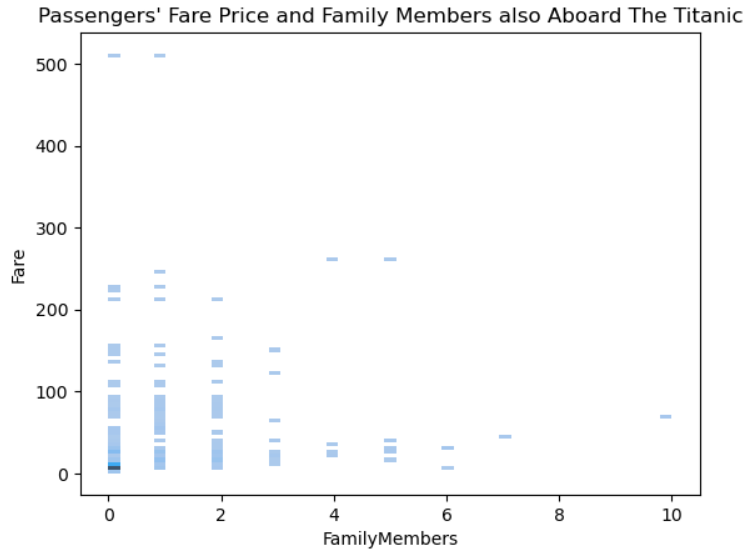
sns.countplot(data = trainingData, x = "FamilyMembers").set(title = 'The Titanic: Number of Passengers with Family Members also Aboard')



A histogram of ***FamilyMembers* and *Fare*** shows us that the two attributes are not directly proportional:

#Histogram of FamilyMembers and Fare columns.

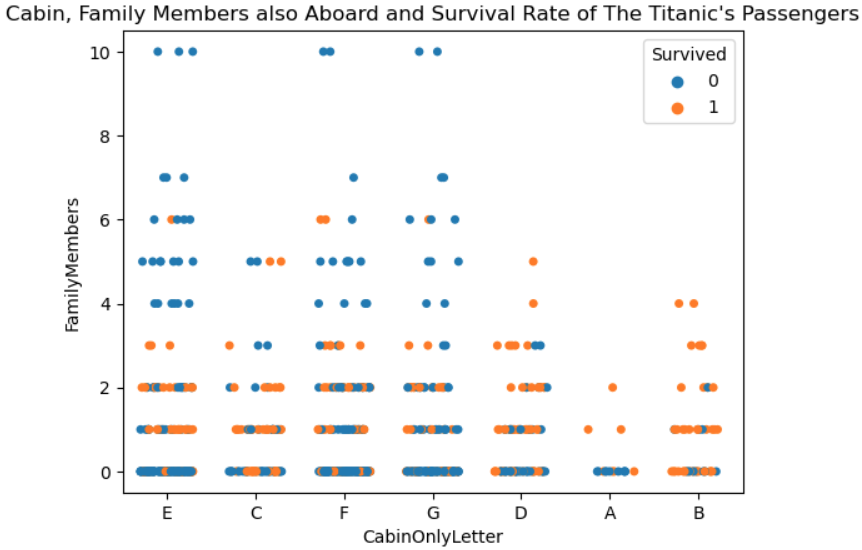
sns.histplot(data = trainingData, x = "FamilyMembers", y = "Fare").set(title = 'Passengers\' Fare Price and Family Members also Aboard The Titanic')



*FamilyMembers* allows us to form a more accurate opinion on our data. The following graph is a strip plot of ***FamilyMembers*, *CabinOnlyLetter* and *Survived***:

#Strip plot of Survived, FamilyMembers and CabinOnlyLetter columns.

sns.stripplot(data = trainingData, x = "CabinOnlyLetter", y = "FamilyMembers", jitter = 0.3, hue = "Survived").set(title = 'Cabin, Family Members also Aboard and Survival Rate of The Titanic\'s Passengers')

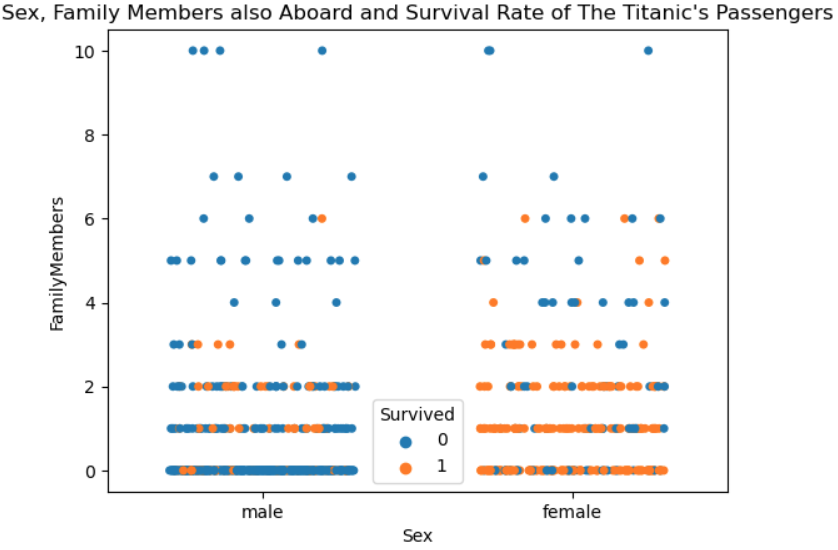


The plot shows that the only categories of passengers that had higher chances of survival were those with 1 to 3 family members travelling along. This coincides with the *Survived*/*SibSp* and *Survived*/*Parch* visualizations, where we could see that most of the passengers that travelled alone perished in the accident. Families of 4 and above had only slight chances of surviving.

As for women, a strip plot of ***FamilyMembers*, *Sex* and *Survived*** once again shows us that women with few (0 to 3) family members survived the sinking in greater numbers than men and had better survival odds than their counterparts travelling with many relatives:

#Strip plot of Survived, FamilyMembers and Sex columns.

sns.stripplot(data = trainingData, x = "Sex", y = "FamilyMembers", jitter = 0.3, hue = "Survived").set(title = 'Sex, Family Members also Aboard and Survival Rate of The Titanic\'s Passengers')



Summarizing our EDA’s results, we believe that, overall, **women travelling alone with up to three relatives had the most chances of surviving, no matter their age and cabin category, and single men had the lowest**.

References

KAGGLE (2019) *Titanic – Machine Learning from Disaster*. [Online] Available from: https://www.kaggle.com/competitions/titanic/data. [Accessed: 20 March 2023].

MYATT, G.J. and JOHNSON, W.P. (2014) *Making Sense of Data I*. [Online] Second Edition. Wiley. Available from: https://archive.org/details/makingsenseofdat0000myat/mode/2up. [Accessed: 15 March 2023].

NADIA (2021) *Titanic EDA Tutorial*. [Online] Available from: https://www.kaggle.com/code/demidova/titanic-eda-tutorial. [Accessed: 22 March 2023].

NAIR, A. (2019) *How to Get Started with Exploratory Data Analysis & Data Preprocessing*. [Online] Available from: https://analyticsindiamag.com/tutorial-get-started-with-exploratory-data-analysis-and-data-preprocessing/. [Accessed: 12 March 2023].

PRAJWAL, C.N. (2022) *EDA - Exploratory Data Analysis: Using Python Functions*. [Online] Available from: https://www.digitalocean.com/community/tutorials/exploratory-data-analysis-python. [Accessed: 25 March 2023].

SHIN, T. (2020) *An Extensive Step by Step Guide to Exploratory Data Analysis*. [Online] Available from: https://towardsdatascience.com/an-extensive-guide-to-exploratory-data-analysis-ddd99a03199e. [Accessed: 15 March 2023].

SMITHSONIAN (2017) *The Titanic*. [Online] Available from: https://www.si.edu/spotlight/titanic. [Accessed: 10 April 2023].

WIKIPEDIA (2023a) *Sinking of the Titanic*. [Online] Available from: https://en.wikipedia.org/wiki/Sinking\_of\_the\_Titanic. [Accessed: 10 April 2023].

WIKIPEDIA (2023b) *Titanic*. [Online] Available from: https://en.wikipedia.org/wiki/Titanic. [Accessed: 10 April 2023].

ZIMMERMAN, W. (2023) Describing Data, Part 1, STAT 200. [Lecture notes] *Skewness & Central Tendency*. [Online] Elementary Statistics. The Pennsylvania State University, Eberly College of Science, online courses. Available from: https://online.stat.psu.edu/stat200/lesson/2/2.2/2.2.4/2.2.4.1. [Accessed: 26 March 2023].