Exercise 7.2: Titanic Case Study Part 2

DSC 550

Taniya Adhikari 1/31/2021

Part 1 Analysis

Case Study: Analyze data to predict who will Survive the Titanic

In [3]:
```
import pandas as pd
import numpy as np
import string
import re
import matplotlib.pyplot as plt
from collections import Counter
import yellowbrick
```

1. Load the data from the “train.csv” file into a DataFrame.

In [84]:
```
# Loading the data
data = pd.read_csv("train.csv")
```

2. Display the dimensions of the file (so you’ll have a good idea the amount of data you are working with.data

In [85]:
```
print("The dimension of the table is: ", data.shape)
```

```
The dimension of the table is:  (891, 12)
```

It has 891 records and 12 variables

3. Display the first 5 rows of data so you can see the column headings and the type of data for each column.

In [86]:
```
# Display the data
data.head(10)
```

Out[86]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Survived variable is Boolean variable with 0 or 1, Missing values have NaN. Survived will be the target variable and all other columns will be features.

5. Look at summary information about your data (total, mean, min, max, freq, unique, etc.) Does this present any more questions for you? Does it lead you to a conclusion yet?

Describing the Statistics of all numerical data

In [87]:
```
print("Describe Data")
data.describe()
```

```
Describe Data
```

Out[87]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Summary for categorical data

In [88]:
```
print("Summarized Data")
data.describe(include=['O'])
```

```
Summarized Data
```

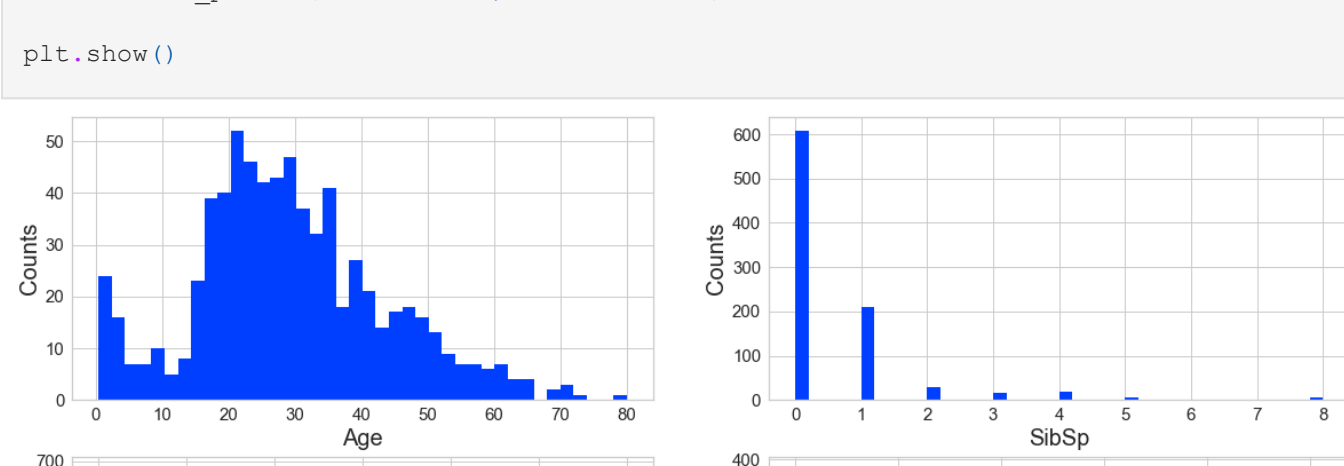
Out[88]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Sjostedt, Mr. Ernst Adolf	male	347082	C23 C25 C27	S
freq	1	577	7	4	644

The minimum and maximum age is .42 and 80. That means it had passengers as young as infants and older people. So Age can be an important feature. Fare also had a big spread, with max value of 512.329. There were higher number of male passengers on the ship.

6. Make some histograms of your data

In [89]:
```
# set figure size
plt.rcParams['figure.figsize'] = (20, 10)

# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)

# Specify the features of interest
num_features = ['Age', 'SibSp', 'Parch', 'Fare']
xaxes = num_features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts']

# draw histograms in for loop
axes = axes.ravel()
for idx, ax in enumerate(axes):
    # drops NaN values
    ax.hist(data[num_features[idx]].dropna(), bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)

plt.show()
```



1. Age Median falls somewhere between 20 to 30, meaning there well more passengers between 20 to 30.
2. Most (around 600 or more) people did not have siblings with them.
3. Most (around 700 or more) people had no parent/children (Parch) on board.
4. Fare data is highly skewd. Majority of fare were below 50 and few tickets were expensive upto 500.

7. Make some bar charts for variables with only a few options.

In [90]:
```
# set figure size
plt.rcParams['figure.figsize'] = (20, 10)

# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)

# make the data read to feed into the visulizer
# Survived Boolean Variable
X_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Survived').size
Y_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Survived').size
# make the bar plot
axes[0, 0].bar(X_Survived, Y_Survived)
axes[0, 0].set_title('Survived', fontsize=25)
axes[0, 0].set_ylabel('Counts', fontsize=20)
axes[0, 0].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
# Embarked categorical variable
X_Embarked = data.groupby('Embarked').size().reset_index(name='Counts')['Embarked']
Y_Embarked = data.groupby('Embarked').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 1].bar(X_Embarked, Y_Embarked)
axes[1, 1].set_title('Embarked', fontsize=25)
axes[1, 1].set_ylabel('Counts', fontsize=20)
axes[1, 1].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
# Pclass categorical variable
X_Pclass = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).groupby('Pclass')
Y_Pclass = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).groupby('Pclass')
# make the bar plot
axes[0, 1].bar(X_Pclass, Y_Pclass)
axes[0, 1].set_title('Pclass', fontsize=25)
axes[0, 1].set_ylabel('Counts', fontsize=20)
axes[0, 1].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
# Sex Boolean variable
X_Sex = data.groupby('Sex').size().reset_index(name='Counts')['Sex']
Y_Sex = data.groupby('Sex').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 0].bar(X_Sex, Y_Sex)
axes[1, 0].set_title('Sex', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)

plt.show()
```

1. There were more passenger in 3rd class compared to 1st and 2nd
2. There were more male passengers than female
3. There were more people mounted from Southampton (S). (Embark — Southampton, Cherbourg, and Queenstown)

8. To see if the data is correlated, make some Pearson Ranking charts

In [91]:
```
#set up the figure size
plt.rcParams['figure.figsize'] = (15, 7)

# import the package for visulization of the correlation
from yellowbrick.features import Rank2D

# extract the numpy arrays from the data frame
X = data[num_features].values

# instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=num_features, algorithm='pearson')

# Fit the data to the visualizer
visualizer.fit(X)

# Transform the data
visualizer.transform(X)

visualizer.show()
```


```
Out[91]: <AxesSubplot:title={'center':'Pearson Ranking of 4 Features'}>
```

9. Use Parallel Coordinates visualization to compare the distributions of numerical variables between passengers that survived and those that did not survive.

In [92]:
```
# setup the color for yellowbrick visulizer
from yellowbrick.style import set_palette
set_palette('sns_bright')

# import packages
from yellowbrick.features import ParallelCoordinates
```

In [93]:
```
# Specify the features of interest and the classes of the target
classes = ['Not-survived', 'Survived']
num_features = ['Age', 'SibSp', 'Parch', 'Fare']
```

In [94]:
```
# copy data to a new dataframe
data_norm = data.copy()
```

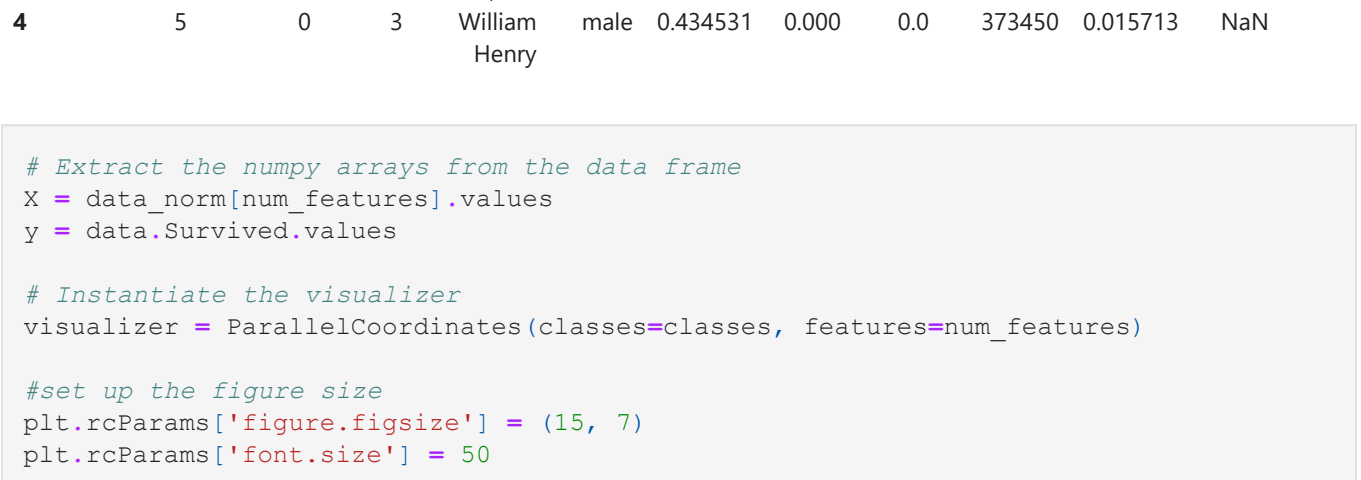
In [95]:
```
# normalize data to 0-1 range
for feature in num_features:
    data_norm[feature] = (data[feature] - data[feature].min(skipna=True)) / (data[feat
```

In [96]:
```
data_norm.head(5)
```

Out[96]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	1	0	3	Braund, Mr. Owen Harris	male	0.271174	0.125	0.0	A/5 21171	0.014151	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	0.472229	0.125	0.0	PC 17599	0.139136	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	0.321438	0.000	0.0	STON/O2. 3101282	0.015469	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	0.434531	0.125	0.0	113803	0.103644	C123	
4	5	0	3	Allen, Mr. William Henry	male	0.434531	0.000	0.0	373450	0.015713	NaN	

In [97]:
```
# Extract the numpy arrays from the data frame
X = data_norm[num_features].values
y = data.Survived.values

# Instantiate the visualizer
visualizer = ParallelCoordinates(classes=classes, features=num_features)

#set up the figure size
plt.rcParams['figure.figsize'] = (15, 7)
plt.rcParams['font.size'] = 50

visualizer.fit(X, y)      # Fit the data to the visualizer
visualizer.transform(X)   # Transform the data
visualizer.show()
```



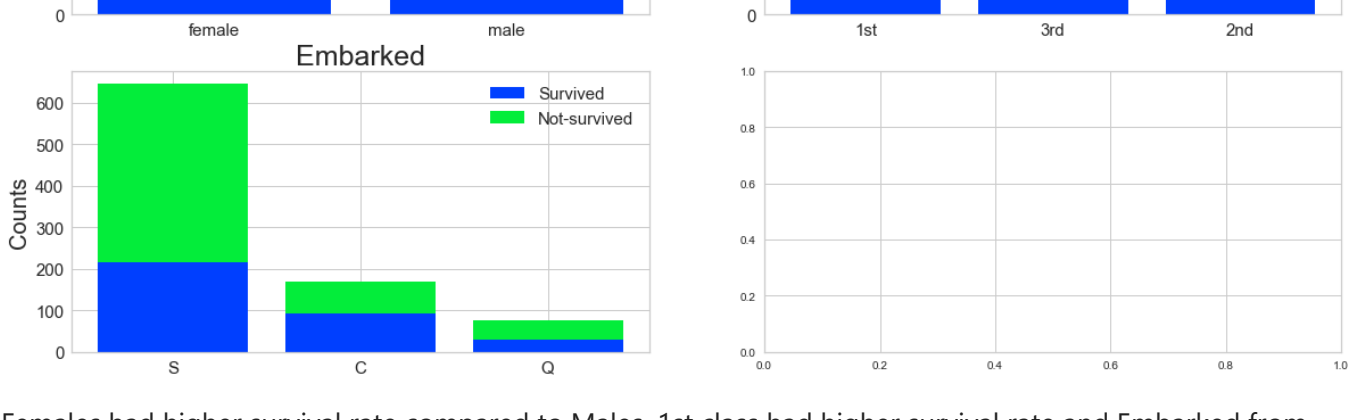

```
Out[97]: <AxesSubplot:title={'center':'Parallel Coordinates for 4 Features'}>
```

Passengers traveling with siblings on the boat have a higher death rate (Blue) and passengers who paid a higher fare had a higher survival rate (Green).

10. Use Stack Bar Charts to compare passengers who survived to passengers who didn’t survive based on the other variables.

Based on Categorical variables Sex, Embarked, Pclass

In [98]:
```
#set up the figure size
plt.rcParams['figure.figsize'] = (20, 10)

# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)

# make the data read to feed into the visulizer
# Categorical variable Sex
Sex_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data['Su
Sex_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data
Sex_not_survived = Sex_not_survived.reindex(index = Sex_survived.index)

# make the bar plot - Sex
p1 = axes[0, 0].bar(Sex_survived.index, Sex_survived.values)
p2 = axes[0, 0].bar(Sex_not_survived.index, Sex_not_survived.values, bottom=Sex_surviv
axes[0, 0].set_title('Sex', fontsize=25)
axes[0, 0].set_ylabel('Counts', fontsize=20)
axes[0, 0].tick_params(axis='both', labelsize=15)
axes[0, 0].legend((p1[0], p2[0]), ('Survived', 'Not-survived'), fontsize = 15)

# make the data read to feed into the visualizer
# Categorical data Pclass
Pclass_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}}).repla
Pclass_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}}).r
Pclass_not_survived = Pclass_not_survived.reindex(index = Pclass_survived.index)

# make the bar plot - Pclass
p3 = axes[0, 1].bar(Pclass_survived.index, Pclass_survived.values)
p4 = axes[0, 1].bar(Pclass_not_survived.index, Pclass_not_survived.values, bottom=Pcla
axes[0, 1].set_title('Pclass', fontsize=25)
axes[0, 1].set_ylabel('Counts', fontsize=20)
axes[0, 1].tick_params(axis='both', labelsize=15)
axes[0, 1].legend((p3[0], p4[0]), ('Survived', 'Not-survived'), fontsize = 15)

# make the data read to feed into the visualizer
# Categorical data Embarked
Embarked_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[dat
Embarked_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})
Embarked_not_survived = Embarked_not_survived.reindex(index = Embarked_survived.index)

# make the bar plot -  Embarked
p5 = axes[1, 0].bar(Embarked_survived.index, Embarked_survived.values)
p6 = axes[1, 0].bar(Embarked_not_survived.index, Embarked_not_survived.values, bottom=
axes[1, 0].set_title('Embarked', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)
axes[1, 0].legend((p5[0], p6[0]), ('Survived', 'Not-survived'), fontsize = 15)

plt.show()
```



Females had higher survival rate compared to Males, 1st class had higher survival rate and Embarked from Southampton has lower survival rate

Part 2 Analysis

11. Some of the questions have been answered by seeing the charts but in some ways, looking at this much data has created even more questions. We will fill in the missing values and eliminate features

In [99]:
```
#fill the missing age data with median value
def fill_na_median(data, inplace=True):
    return data.fillna(data.median(), inplace=inplace)

fill_na_median(data['Age'])
```

In [100...
```
# check the result
print(data['Age'].describe())
```

```
count    891.000000
mean      29.361582
std       13.019697
min        0.420000
25%       22.000000
50%       28.000000
75%       35.000000
max       80.000000
Name: Age, dtype: float64
```

In [101...
```
# fill with the most represented value
def fill_na_most(data, inplace=True):
    return data.fillna('S', inplace=inplace)
```

In [102...
```
fill_na_most(data['Embarked'])
```

In [103...
```
# check the result
print(data['Embarked'].describe())
```

```
count      891
unique       3
top          S
freq       646
Name: Embarked, dtype: object
```

In [104...
```
# subsetting the data and removing the features that are unnecessary
# removing "PassengerId", "Name", "Ticket" and "Cabin".
# (ID is not useful in anyway, and so the name, ticket and cabin also has too many va
data = data[['Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex', 'Embarked']
```

In [105...
```
# check the data
print(data.describe())
```

```
         Survived      Pclass         Age       SibSp       Parch        Fare
count  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.361582    0.523008    0.381594   32.204208
std      0.486592    0.836071   13.019697    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   22.000000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   35.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

In [106...
```
print(data.describe(include=['O']))
```

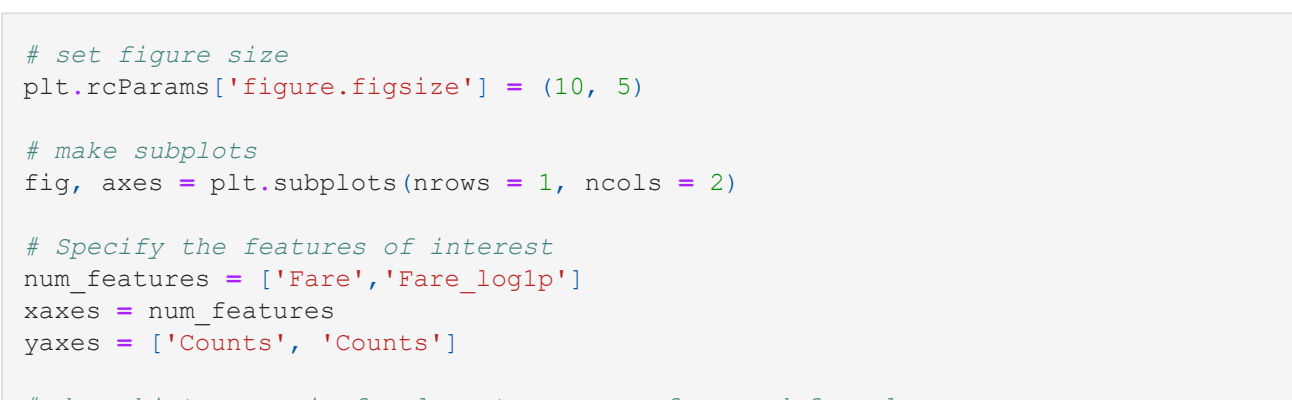
```
         Sex Embarked
count    891      891
unique     2        3
top     male        S
freq     577      646
```

12. If you go back and look at the histograms of Fare, you’ll see that it is very skewed… many low cost fares, not very many high cost fares. Log Transformation is a good method to use on highly skewed data.

In [108...
```
# log-transformation of fare variable because it is skewed variable with log-normal d
def log_transformation(data):
    return data.apply(np.log1p)

data['Fare_log1p'] = log_transformation(data['Fare'])
```

I decided to show the comparision of two histogram to see the difference in log-normal and log-transformed distribution of the fare

In [115...
```
# set figure size
plt.rcParams['figure.figsize'] = (10, 5)

# make subplots
fig, axes = plt.subplots(nrows = 1, ncols = 2)

# Specify the features of interest
num_features = ['Fare','Fare_log1p']
xaxes = num_features
yaxes = ['Counts', 'Counts']

# draw histograms in for loop to compare fare and fare_log
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(data[num_features[idx]], bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)

plt.show()
```



13. Convert your categorical data into numbers (Sex, PClass, Embark)

In [116...
```
# get the categorical data
cat_features = ['Pclass', 'Sex', "Embarked"]
data_cat = data[cat_features]
```

In [117...
```
# replace the variable for Pclass
data_cat = data_cat.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})
```

In [118...
```
# One Hot Encoding
data_cat_dummies = pd.get_dummies(data_cat)
# check the data
print(data_cat_dummies.head(8))
```

```
   Pclass_1st  Pclass_2nd  Pclass_3rd  Sex_female  Sex_male  Embarked_C  \
0           0           0           1           0         1           0
1           1           0           0           1         0           1
2           0           0           1           1         0           0
3           1           0           0           1         0           0
4           0           0           1           0         1           0
5           0           0           1           0         1           0
6           1           0           0           0         1           0
7           0           0           1           0         1           0

   Embarked_Q  Embarked_S
0           0           1
1           0           0
2           0           1
3           0           1
4           0           1
5           1           0
6           0           1
7           0           1
```