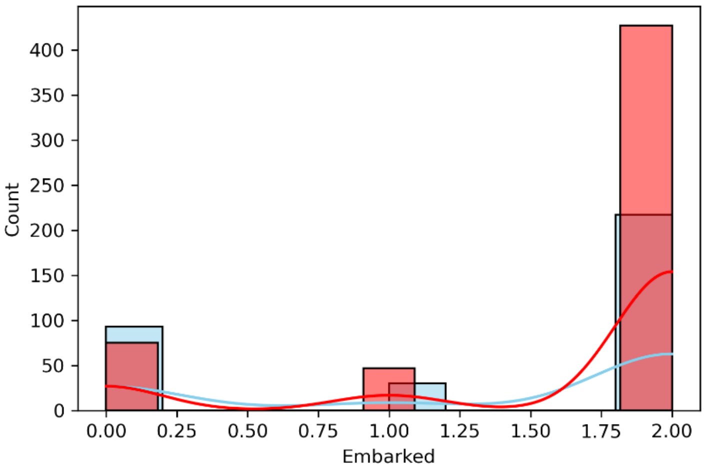
Machine Learning Project 2022-D. Theofilopoulos

Here I will demonstrate the various steps of the project along with comments made by me regarding these steps and my understanding of the problem.

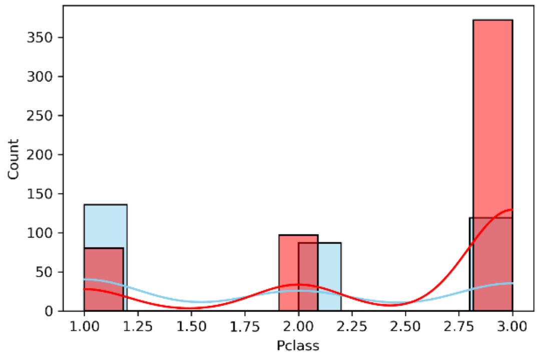
We will begin with visualization. I created a dataset from the original by selecting the features you were suggesting since I also agree on the selection. I don't see a reason why the name of the passenger plays any role. Even if it was a famous person this should also be demonstrated in the fare which as we will see plays a role. Moreover, the ticket number and the cabin don't play any role since they provide more information that proves to be compactified in the ticket fare.

Visualization

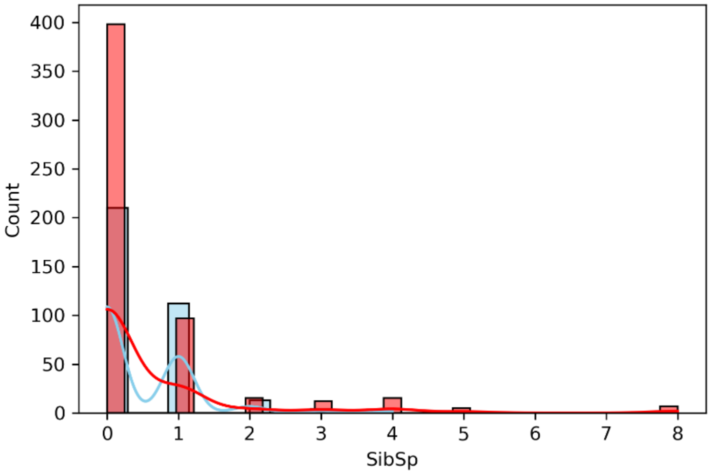
I will present the various plots I made along with some comments. Light blue will refer to the survivors, while red to the one that didn't make it.



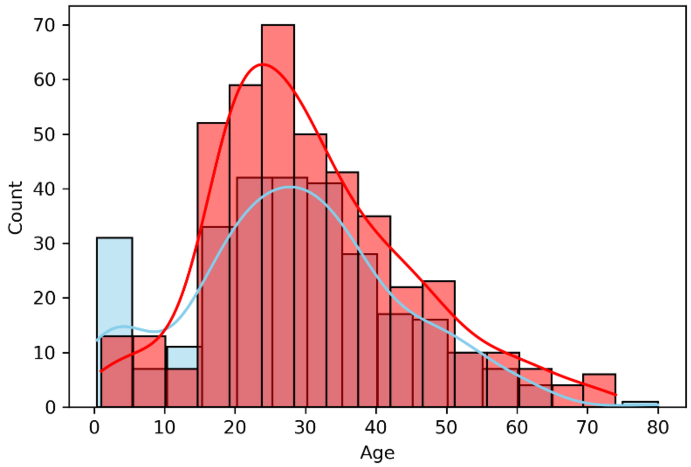
It is clear that the one that embarked on Cherbourg (value 0.0 in the x-axis) had better chances to survive.



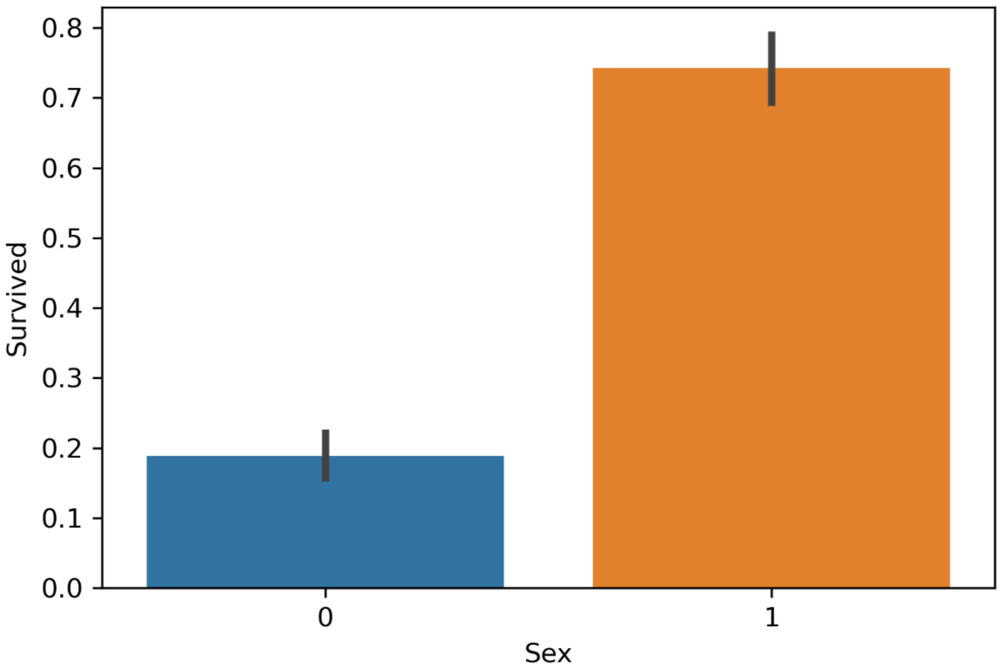
Here we see that the ones that we in the 1st class (value 1.0 in the x-axis) had better chances to survive.



In the above picture, we see that the higher were the family members of a person (spouse or children) the lower were the chances. While a couple of people had some probability to survive (maybe through cooperation) then then probabilities dropped significantly.



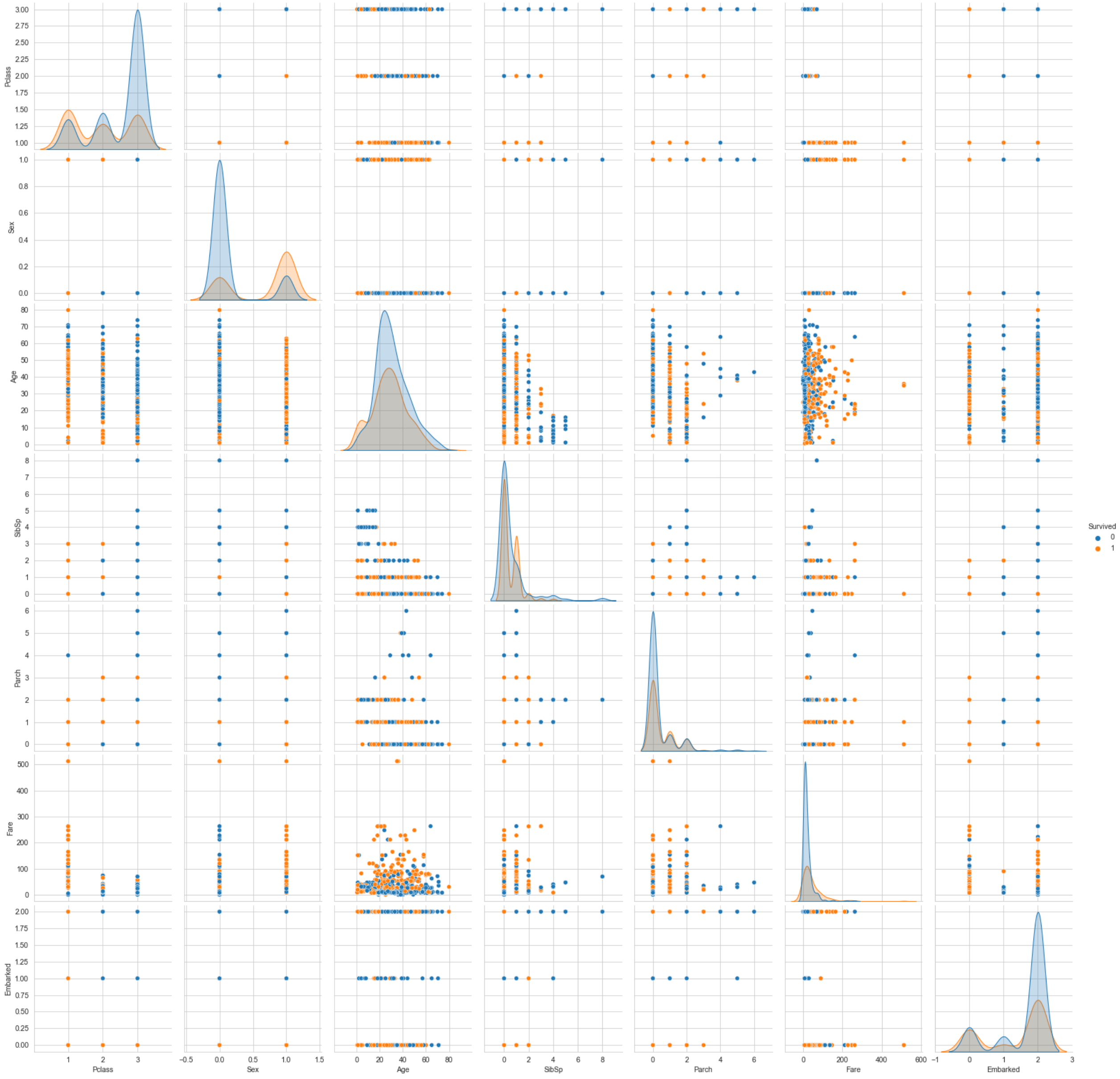
Here we see that most of the passengers were from 18 to 40 years old. The chances of survival for older ages dropped significantly but for ages greater than 75 years old there were only survivors. I guess that the elder people were helped by the other passenger and the ship crew. Also, we see that for kids up to 5 years old the same thing happens. This should imply a sense of help towards the helpless.



Here we see that the male passengers had much better chances to survived than the females. Probably due to physique etc.

I decided to plot some of the feature since these were the ones, I found more appropriate.

Now I comment on the pair of features to see if there is one pair that will give good classification



Since the problem is more complex than the flower classification, I couldn't find a suitable pair candidate. The ones I found that would give better results in the classification are the pairs sex-age, Pclass-Embarked(1st row-last column).

Since we see that the data are more complex (and more in number) I believe that we won’t see an AUC of 1 since it will be very difficult for the Classifier to get a perfect score. The columns that have NaN values are the ones that depict Age and Embarked. These are the ones that we will remove in the final step.

Holdout method

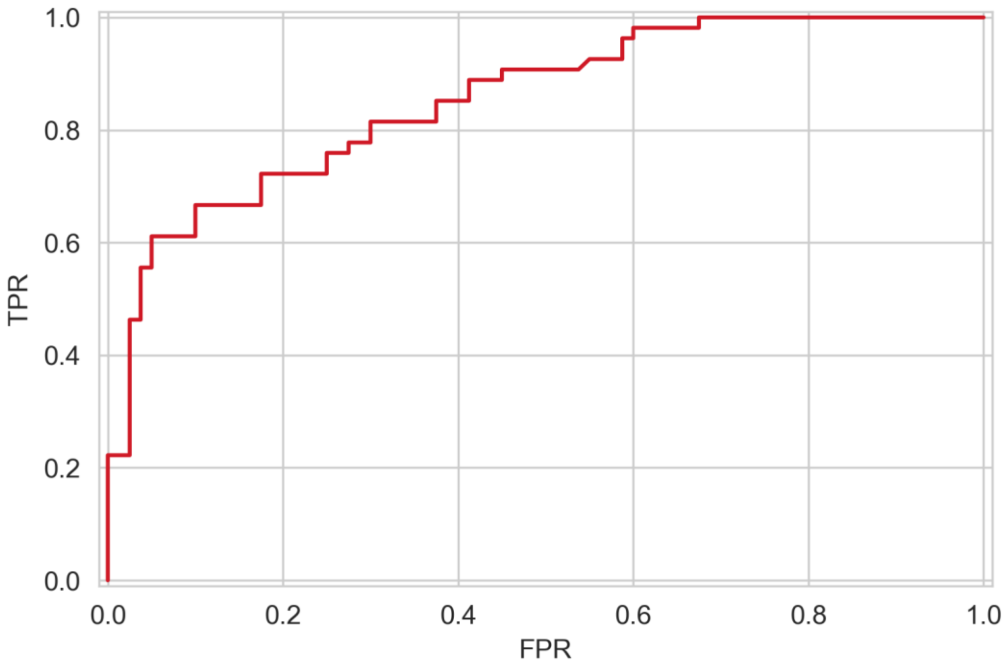
Here by following your example I divided the training set to be 80% of the data and the rest the validation. Of course, our target variable is the column “Survived”. Before proceeding with the XGBoost Classifier I applied the MinMax scaling as feature normalization. By this scaling the training and the validation data will take values between 0 and 1, according to the formula

$$x' = \frac{x - x\_\{min\}}{x\_\{max\} - x\_\{min\}}$$

I will make a table with the test size, the accuracy and the classifier

Test size	Accuracy	ROC-AUC
0.10	0.788	0.847
0.15	0.80	0.854
0.20	0.77	0.835
0.25	0.77	0.834
0.30	0.75	0.825

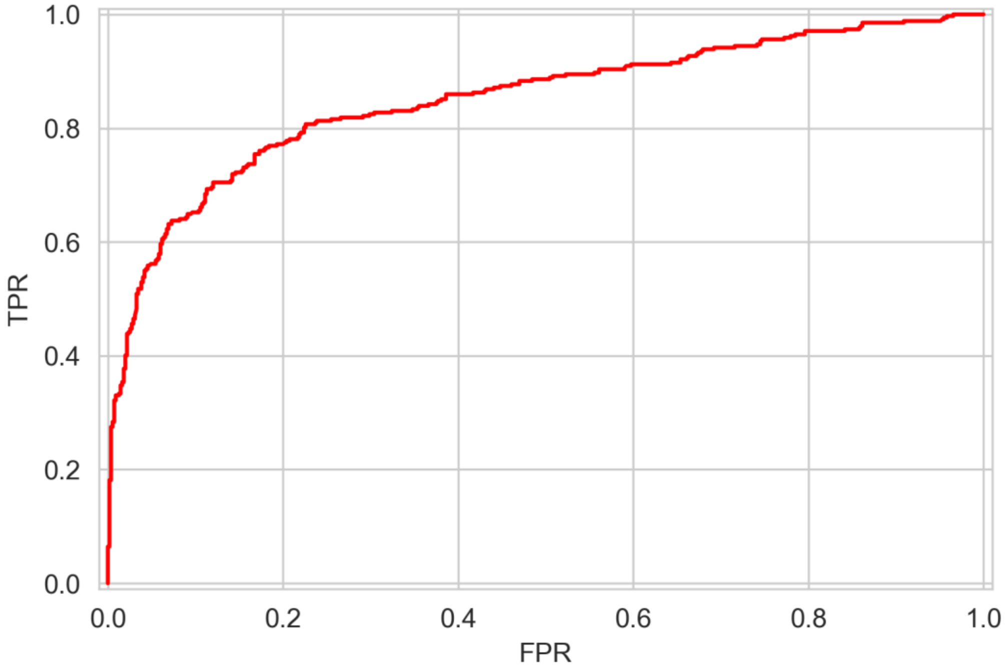
We see that the optimal performance come from when the test is 15% of the dataset.
As expected, we didn’t get an AUC of 1 but an AUC of 0.85. Since we don’t have the same number of each class, accuracy is not the best metric how well our classifier works.
Below is the ROC-AUC for the case where the test data set is 15% of the data.



k-fold validation

Here we present the ROC-AUC for the case of k=5. We haven't applied any scaling to our feature data. We get a AUC of 0.8502513874242376 which is similar to the holdout case with a test set of 15%.

The plot is show below



I tried also for k=10 and I got an AUC of 0.86.

Then I ran the **holdout** classifier. The result is

```
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259258
AUC std is  1.1102230246251565e-16
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.854050925925926
AUC std is  1.1102230246251565e-16
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.854050925925926
AUC std is  1.1102230246251565e-16
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
```

```
AUC std is  0.0
Accuracy of Classifier with threshold set to 0.5:  0.8059701492537313
Mean AUC is  0.8540509259259259
AUC std is  0.0
```

We see that the mean is almost the same after 10 processes and the std is 0.
For the kfold after 10 runs we have

```
ROC-AUC of XBGClassifier:  0.8502513874242376
Mean AUC is  0.8540509259259259
AUC std is  0.0
ROC-AUC of XBGClassifier:  0.8502513874242376
Mean AUC is  0.8540509259259259
AUC std is  0.0
ROC-AUC of XBGClassifier:  0.8502513874242376
Mean AUC is  0.8540509259259259
AUC std is  0.0
ROC-AUC of XBGClassifier:  0.8502513874242376
Mean AUC is  0.8540509259259259
AUC std is  0.0
ROC-AUC of XBGClassifier:  0.8502513874242376
Mean AUC is  0.8540509259259259
AUC std is  0.0
```

We see that the performance in the kfold classification is more stable than the holdout method since the ROC-AUC is every time the same.

The final step is to remove the columns that have NaN values (Age and Embarked) and create the new dataset. The apply the MLP classifier for different amount of hidden layers.

For 1 hidden layer: ConvergenceWarning:

```
number of iterations of the solver:  300
num of layers:  3
Num of o/p:  1
Accuracy of MLPClassifier with threshold set to 0.5:  0.7238805970149254
ROC-AUC of MLPClassifier:  0.8386574074074075
```

(Worse that the other 2 methods)
For 3 hidden layers: ConvergenceWarning

```
number of iterations of the solver:  300
num of layers:  5
Num of o/p:  1
Accuracy of MLPClassifier with threshold set to 0.5:  0.7835820895522388
ROC-AUC of MLPClassifier:  0.8212962962962963
```

For 4 hidden layers

```
number of iterations of the solver:  397
```

```
num of layers:  6
Num of o/p:  1
Accuracy of MLPClassifier with threshold set to 0.5:  0.753731343283582
ROC-AUC of MLPClassifier:  0.8418981481481481
```

We see that the MLP classification produces close results but worse than the other two methods. A reason for this maybe that the fact due to NaN values in columns of Age and Embarked, we had to remove them. Thus, we have removed data that play a role in the classification. When we used the XGBooster this was not necessary since this classifier can deal with NaN values.