Survival Rates of the Titanic Passengers

The sinking of the titanic is the worst and possibly the most famous shipwreck of modern history, with the dead reaching 1503 souls. The goal of the research project into the survival rates of the titanic is to see if we could train a supervised learning model to predict future survival rates based on the provided dataset.

# Data Processing

We're working with a dataset, graciously provided by our professor, consisting of 13 features. For the initial phase, I handpicked 5 features that intuitively felt relevant in determining survival outcomes;

All provided features in dataset: ['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']

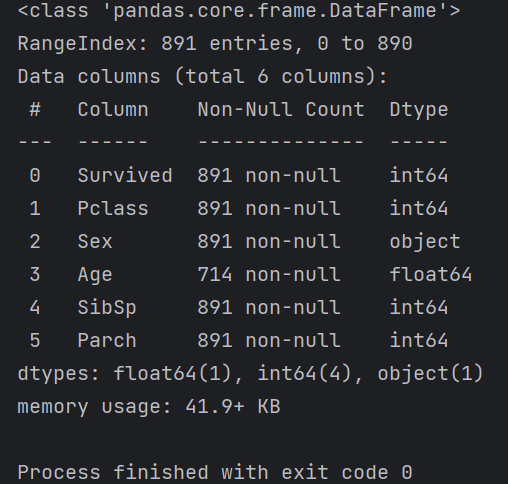
I choose to work with 5 features for my first survival rate model to predict the ['Survived'] column with 890 samples.

First supervised learning model features: ['Pclass', 'Age', 'Parch', 'SibSp', 'Sex', 'Survived'(output)]

Using the load\_data() function, I streamlined the data acquisition process. This function is tailored to fetch only the columns I've earmarked for this phase.

def load\_data():  
 *"""Load data from Titanic CSV files."""*

filename = 'titanic\_passengers.csv'data\_dir = os.path.join(os.path.dirname(\_\_file\_\_), '..', 'data')  
 path = os.path.join(data\_dir, filename)  
 data = pd.read\_csv(path, usecols=['Pclass', 'Age', 'Parch', 'SibSp', 'Sex', 'Survived'])  
 df = pd.DataFrame(data)  
 return df

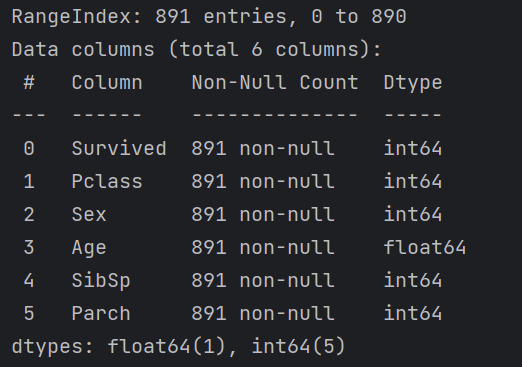
**** An initial df.info() run highlighted some cleaning tasks: filling missing age data and converting the gender column from a text to a binary format for computational efficiency. For the age gaps, I leaned on the KNNImputer, which, in a nutshell, guesstimates missing values based on the proximity of other data points.

In order to do this I would need to create a function convert\_sex(df) that would take in the dataframe, target the sex column and replace all ‘male’ with 0 and ‘female’ with 1 which would allow the supervised model to use that feature in its predictions.

def convert\_sex(df):  
 *"""Sex change from obj to binary values"""* df = df[df['Sex'] != '']  
 df['Sex'] = df['Sex'].replace({'male': 0, 'female': 1})  
 return df

To fill in the missing data for the 177 missing samples in the age I created a function to handle the KNNImputer(); fill\_age\_knn(df). To do that I had to separate the age column into 2 groups, missing\_age\_data and available\_age\_data. It then uses the age column of the available set along with the Parch(number of parents or children) and SibSp (the number of siblings or spouses) that the passenger has on board to guesstimate an appropriate number for the missing value.

def fill\_age\_knn(df):  
 *"""fill in missing values for Age using KNNImputer"""* df\_imputed = df.copy()  
 missing\_age\_data = df\_imputed[df\_imputed['Age'].isnull()]  
 available\_age\_data = df\_imputed[df\_imputed['Age'].notnull()]  
  
 imputer = KNNImputer(n\_neighbors=3)  
 imputer.fit(available\_age\_data[['Age', 'Parch', 'SibSp']])  
 imputed\_data = imputer.transform(missing\_age\_data[['Age', 'Parch', 'SibSp']])  
 imputed\_data = np.round(imputed\_data[:, 0], 2)  
 df\_imputed.loc[missing\_age\_data.index, 'Age'] = imputed\_data  
  
 return df\_imputed

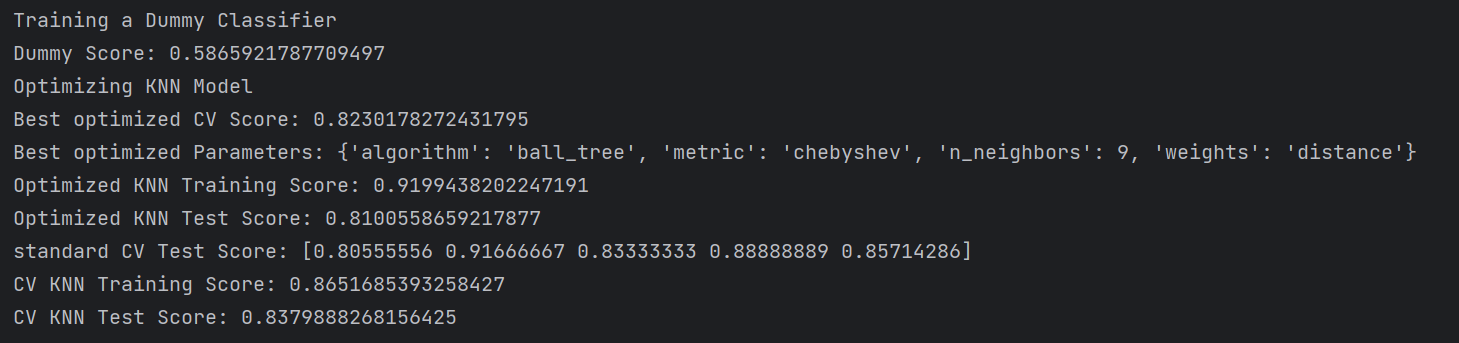
After calling df.info() we are presented with a workable data frame

# Model and Data split

I choose to go with the k-nearest neighbor (KNN) model because we were taught this model in class, and it performs well for the binary classification that I am looking to perform. There are other more advanced models I am sure; however, I do want to stay within the confines of what is taught within the class environment.

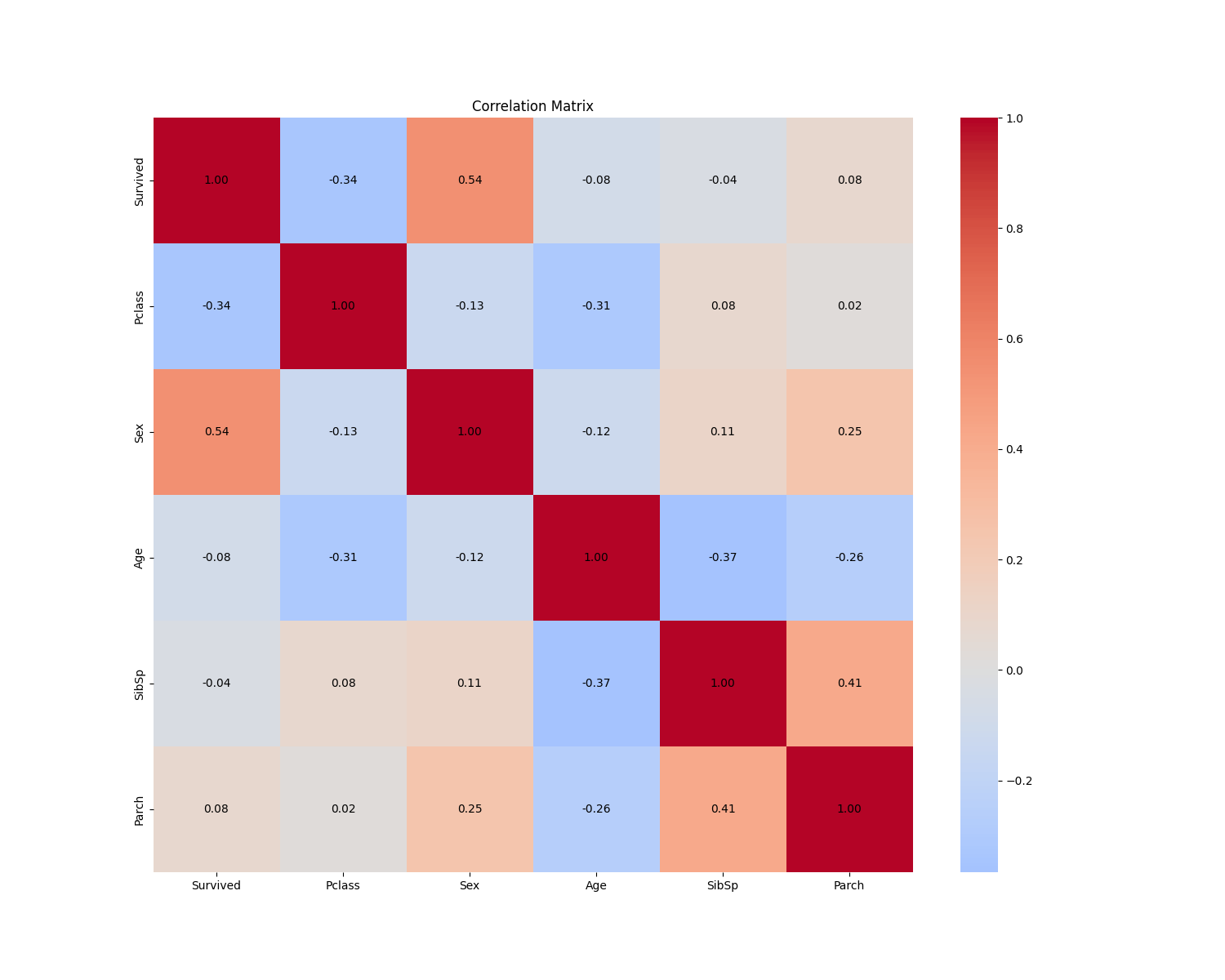
To split the data at first I just did the regular 2 split, training and test. But after week 4 of the MIS class, we were taught about kfolds and advanced validation, so I adjusted my models to take that into account, which improved the variation by 4 points in both training and test scores.

I built a total of 3 models to pit against each other for this first iteration; a KNN with Cross Validation (CV) model, a Optimized KNN with CV model (one that I had a GridSearchCV() find the best params and run the model again with them) and a ElasticNet Model.

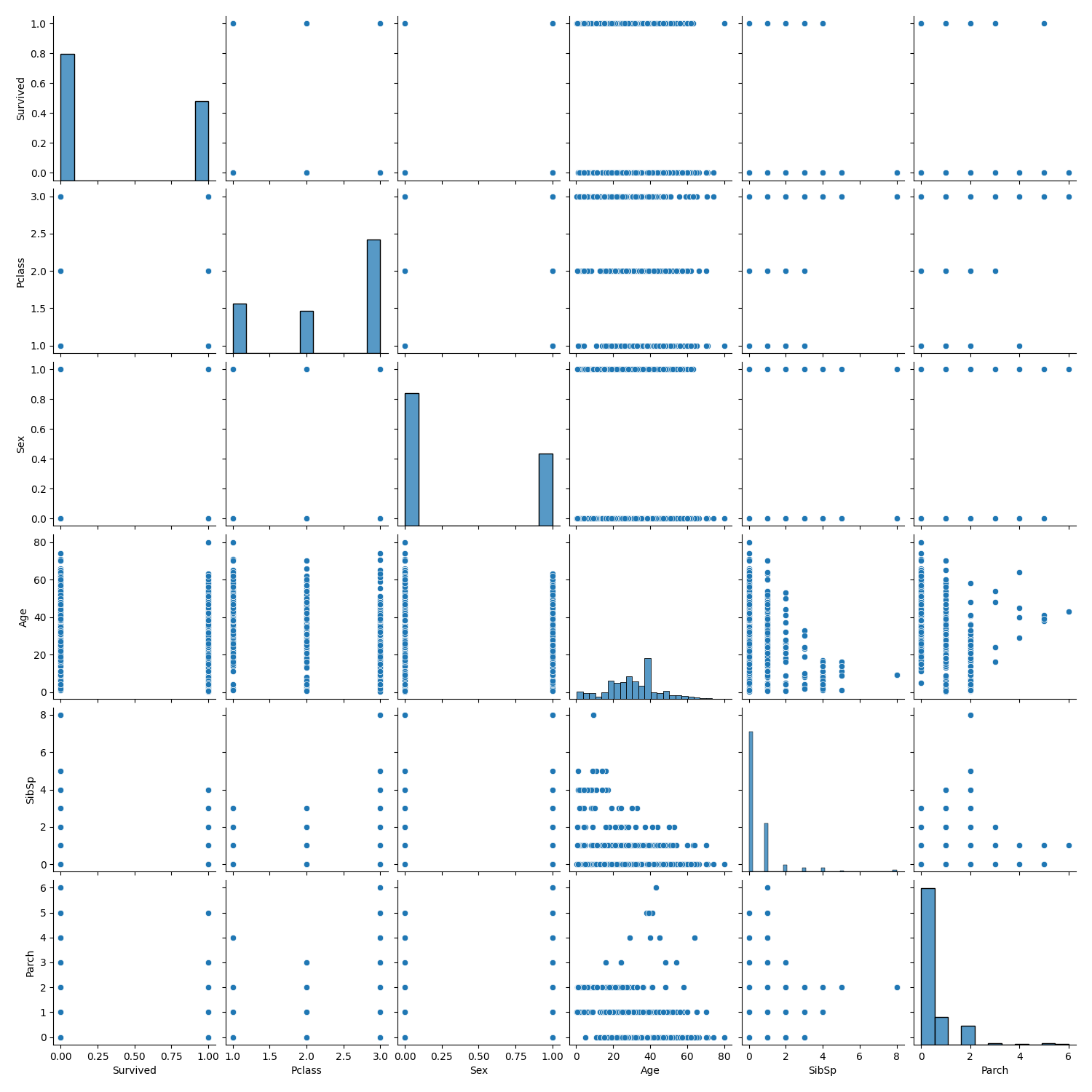


Due to my inability to reduce the overfit on the optimized model I choose to go with the Standard KNN\_CV Model I created that had a more appropriate training and test score ratio.

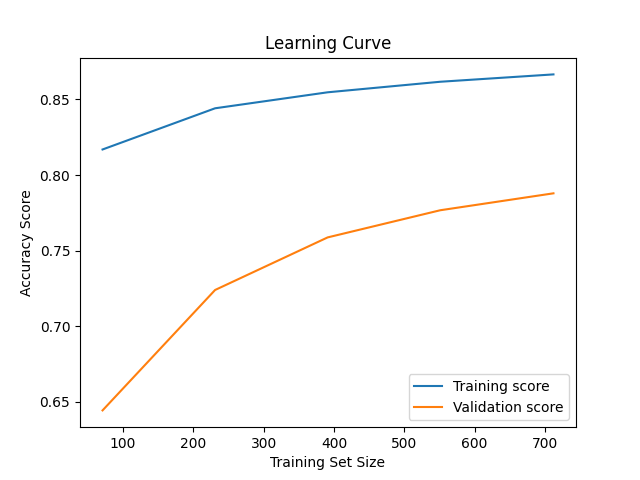
# Exploratory Data Analysis (EDA)

* For the Exploratory Data Analysis process, I created a collection of plots, first one was a heatmap to show if there was any obvious connections between the data. 

The plot shows that there was no strong connection between any of the features in the set. To see the collinearity between the various features, I created a pair plot to quickly see if I missed anything on the heatmap.



As you can see its much the same as the heatmap, not showing much useful information. Next I moved on to see if adding more data would be useful by creating a learning curve.

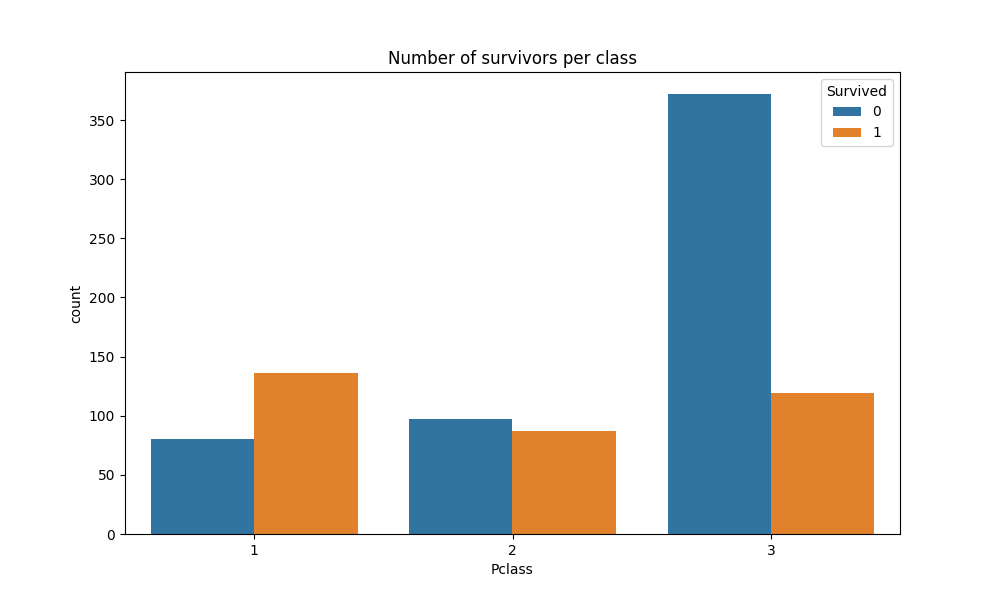
 With the upward trend starting to plateau, we can be confident that more data would not help much, but since its not plateaued yet, more data could be helpful

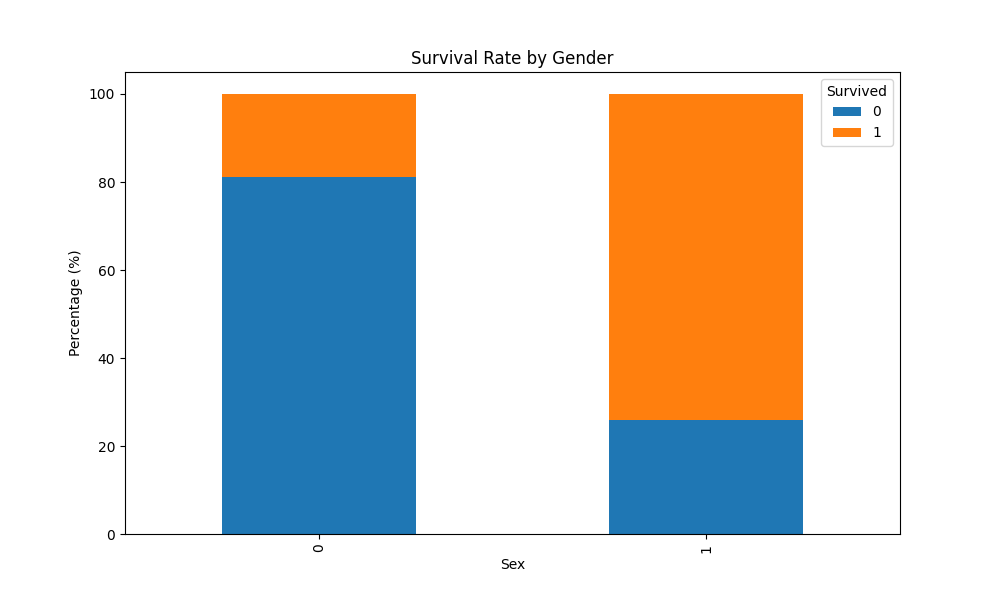
A graph of a number of neighbors

Description automatically generated

To better visualize the performance decrease with the increase in the n\_neighbors property of the KNN model, I created a validation curve that tracks the models performance throught the range of n\_neighbor numbers.

The most important feature for the variance of the model appears to be the gender of the sample, followed by the Age and how big the family was that was on board the titanic at the time of the sinking.



The lease interesting findings I found was that there was a higher survival rate in the 1st class areas and surprisingly the 3rd class area.  
 So then I wanted to check out if Sex had any effect on survival rates.

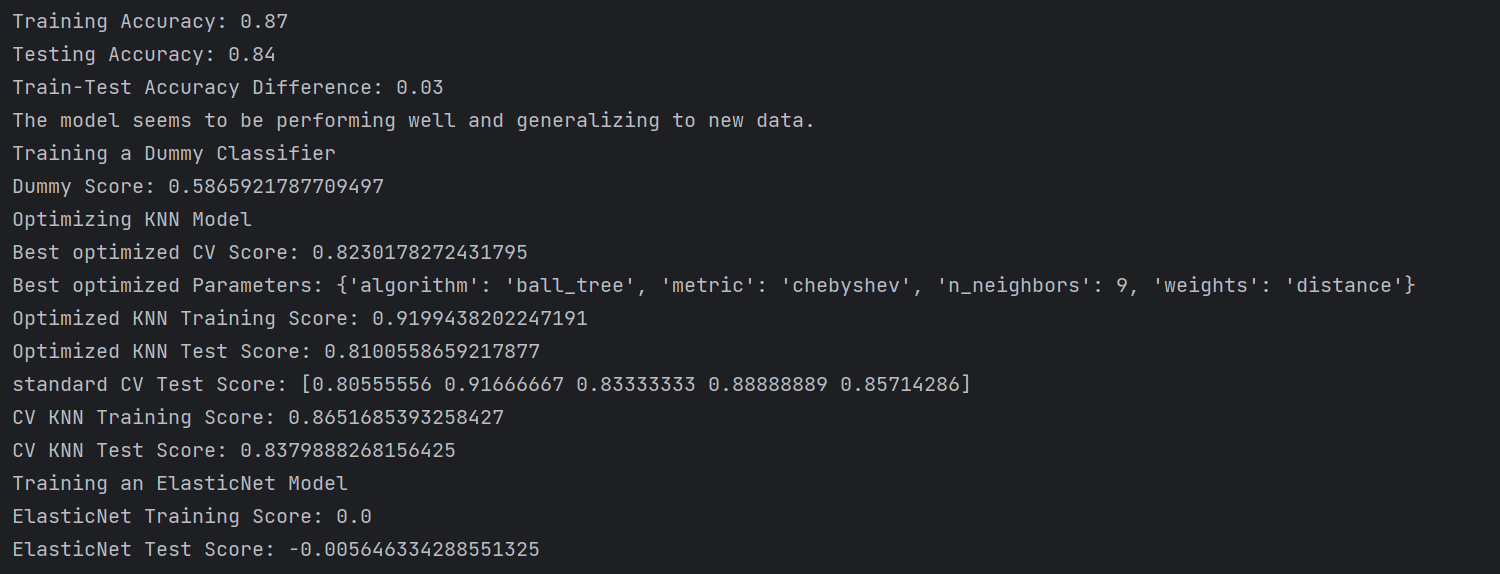
As you would imagine the females (1) had a much higher survival rate than the males (0) which is to be expected with the saying of “Women and children first”

# The model code and breakdown

def knn\_model\_cv(X\_train, y\_train, X\_test, y\_test, k\_folds=5, n\_neighbors=3):  
  
 #training KNN  
 model = KNeighborsClassifier(n\_neighbors=n\_neighbors)  
 cv\_scores = cross\_val\_score(model, X\_train, y\_train, cv=k\_folds)  
 print(f"Mean CV Accuracy: {np.mean(cv\_scores):.2f}, Std: {np.std(cv\_scores):.2f}")  
 model = model.fit(X\_train, y\_train)  
  
  
 #prediction and testing  
 y\_pred = model.predict(X\_test)  
 test\_accuracy = accuracy\_score(y\_test, y\_pred)  
  
 train\_accuracy = model.score(X\_train, y\_train)  
  
 difference = train\_accuracy - test\_accuracy  
 print(f"Training Accuracy: {train\_accuracy:.2f}")  
 print(f"Testing Accuracy: {test\_accuracy:.2f}")  
 print(f"Train-Test Accuracy Difference: {difference:.2f}")  
  
 if train\_accuracy > test\_accuracy and difference > 0.10:  
 print("The model might be overfitting because the training accuracy is significantly higher than the testing accuracy.")  
 elif train\_accuracy < 0.75 and test\_accuracy < 0.75:  
 print("The model might be underfitting as both training and testing accuracies are low.")  
 else:  
 print("The model seems to be performing well and generalizing to new data.")  
  
 return model

The model I decided to go with for my predictions is a KNN model I made by creating a function which I can call from main and change the attributes like k\_fold, n\_neighbors to quickly see the change in performance by the numbers and difference is then relayed back to the console with a text line depending on how the data scores.

The reason I choose the CV KNN model because it was the most accurate model I created. The Optimized model had an issue being major overfit which I could not decrease, and the EleasticNet neural network I tried just to see the results was an astounding failure.



Conclusion

Through the project I was able to successfully train a Supervised Learning Model ‘k-nearest neighbor’ (KNN) to predict who survived and who did not at a training rate of .87 and a test score of .84 with a difference in variance of .03. There is still room for improvement for the model through possible additional feature engineering that I didn’t think of, or by using a different ML models. The outcomes were interesting and enlightening about the events that happened that night. All in all this was a very interesting and challenging exercise in applying Machine Learning to a real life event and interpreting the outcome.