Machine Intelligence Systems

Research Project

Analyzing Survivors of the Titanic

Ethan Stanks
9-1-2023

Introduction

For this research project, a data collection containing information on the survivors of the titanic was given. Each row represented a passenger on board. The columns were the data collected from each passenger. The data collected was if the passenger survived, their ticket class, their sex, their age in years, number of siblings and spouses, number of parents and children, their ticket number, the cost of their fare, their cabin number, and what port they embarked from.

Objectives

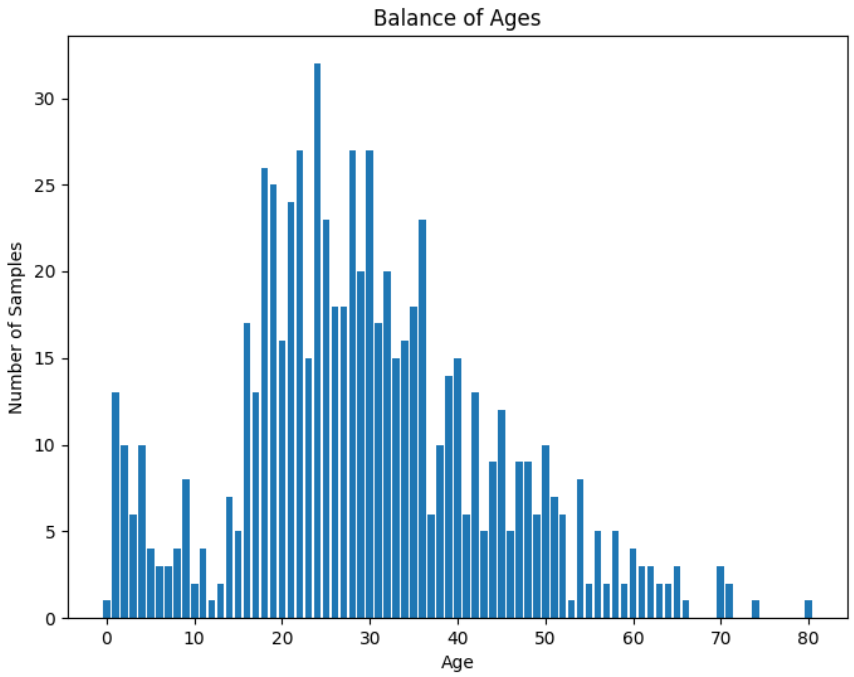
The purpose for this project is to gain a better understanding of applying machine learning algorithms to a real-world application of data. The instructions for this project are to load the data and clean any sample or features that need it, perform exploratory data analysis, find a model that makes good predictions, produce visualizations, do validations on the model, and write about the findings.

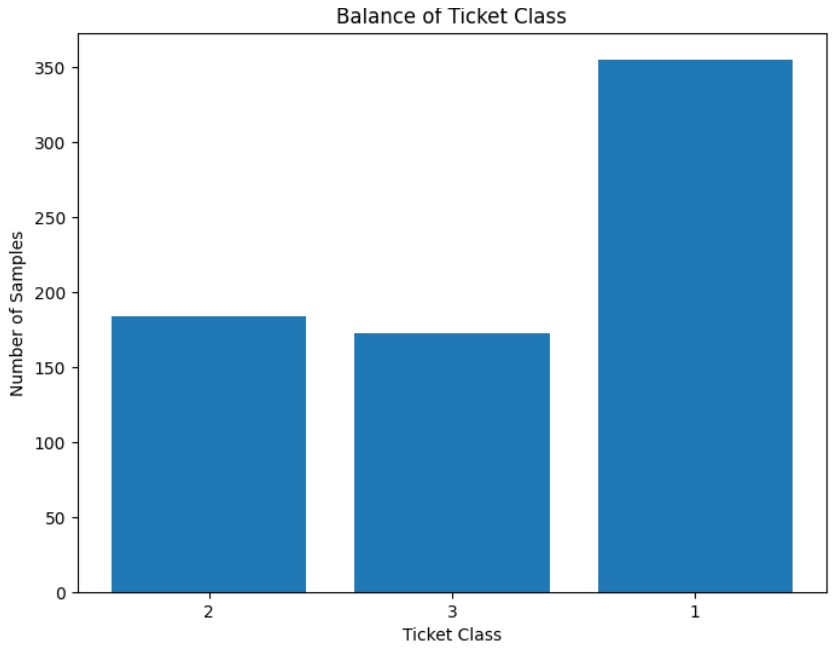
My personal objective is to predict if a passenger would survive the sinking. I've seen the titanic movie so from my perspective I know that mostly woman and children were saved in the movies. I'd like to see if this matches up with the data. Also, the movie showed lower class passengers locked out from getting to the emergency boats. I'd like to know if a passenger's class affects their survival to see if it matches the movie.

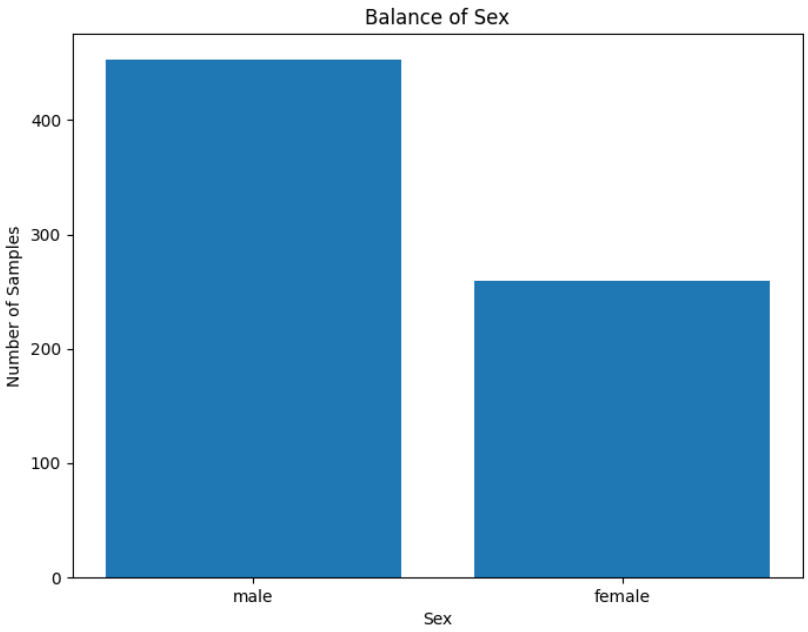
Exploratory Data Analysis

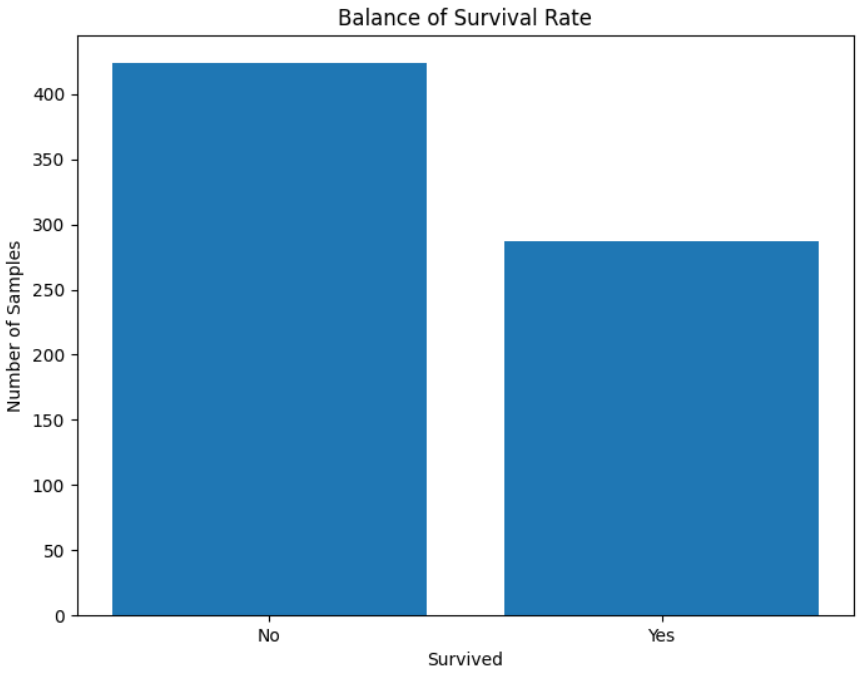
To start making predictions using the data, I need to know what this data holds. I am using Pandas, a data structure and data analysis Python library, to read the (.csv) data file. Loading the data, I can see that there are 891 rows, but most columns are missing data. 'PassengerId', 'Survived', 'Pclass', 'SibSp', and 'Parch' columns contain int64 data. 'Age' and 'Fare' columns contain float64 data. 'Name', 'Sex', 'Ticket', 'Cabin', and 'Embarked' columns contain object data, most likely Strings. Out of all the columns, I'd like to use 'Sex', 'Pclass', and 'Age' to predict if a passenger will survive. I will need to round 'Age' to a whole number as the data contains decimals for estimated ages. 'Pclass' is an int64 so I won't need to do any type conversions for it. 'Sex' is in String form which is either 'male' or 'female', but I can convert that to dummy code later.

To make good predictions for whether a passenger will survive, the data can't have a huge class balance issue. If one class significantly outweighs the others, a model might become overly biased toward the majority class. I created three separate bar graphs to investigate the class balances for each 'Age', 'Pclass', and 'Sex'. Also, I created a bar graph to see the balance between passengers who survived to see any issues with bias for the output.


Balance of Ages
Number of Samples
Age
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
80

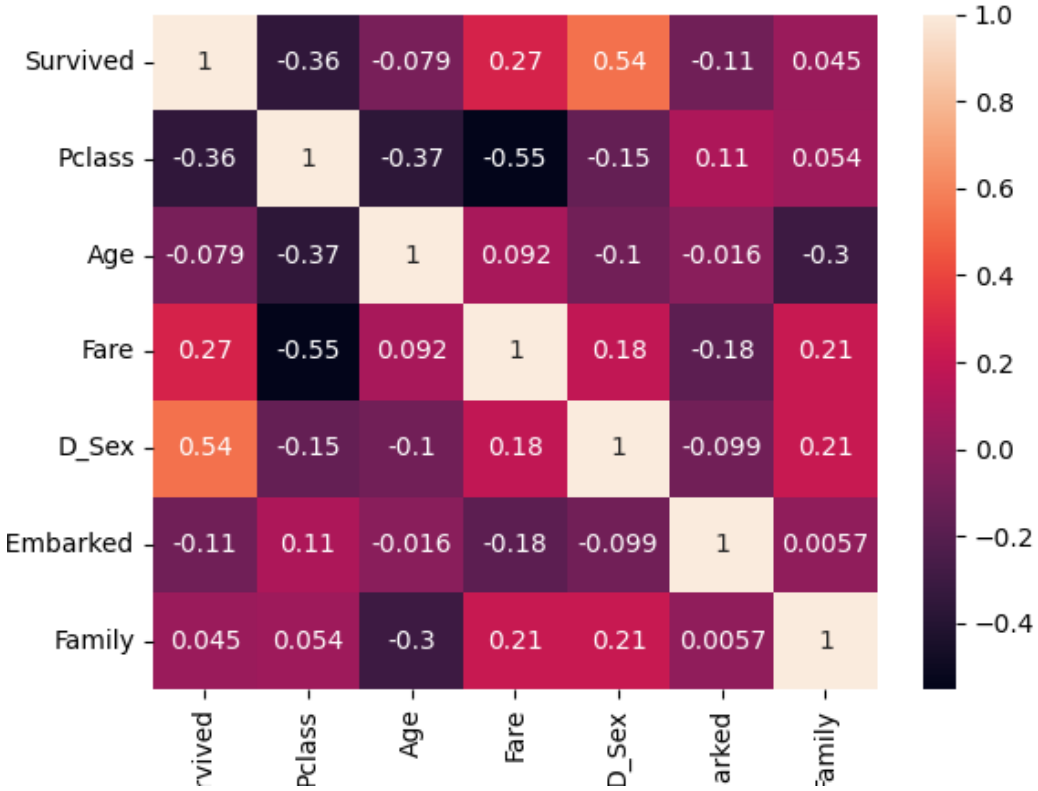

Balance of Ticket Class
Number of Samples
Ticket Class
0
50
100
150
200
250
300
350
2
3
1


Balance of Sex
Number of Samples
Sex
0
100
200
300
400
male
female



Analyzing these graphs there may be some issues with class balances. ‘Age’ looks healthy and well spread out. There are roughly 150 more males than females which shouldn’t cause any bias. Ticket class two and three have roughly the same number of passengers, but ticket class 1 has 150 more passengers, so the model might have bias for passengers with a ‘Pclass’ of 1. Most of the passengers didn’t survive, but there is only a 100-passenger difference so class balance shouldn’t affect the output feature.

Moving forward, we need to see if ‘Age’, ‘Pclass’, and ‘Sex’ correlate to ‘Survived’. Since ‘Survived’ will be the output for the model knowing what features correlate will help build a great model. This will help with identifying relationships between features as well as making sure we are picking the correct ones to use as the input.



Analyzing the correlation plot, there are some correlations between features with the model's output as well as correlations between possible inputs. I want to use 'Age', 'Pclass', and 'Sex' so I'm going to focus on their correlations. 'Pclass' has a correlation of -0.36 to 'Survived'. This can indicate that having a lower 'Pclass' can lead to a low survival rate. 'Age' has a very low correlation to 'Survived' with a score of -0.079, so it might not be a good fit for this model. 'Sex' has a high correlation of 0.54, so it'll be a perfect fit for this model. The input features also can't have high correlation to one another, or else multicollinearity could occur, leading to unstable model coefficients. 'Pclass' doesn't have any correlation to 'Sex' but has a -0.37 correlation to 'Age' which might throw the model off. 'Sex' doesn't have any correlation to 'Pclass' or 'Age'. 'Fare' has a correlation to 'Survived' by 0.27, but this may be caused from 'Fare' having a correlation of -0.55 with 'Pclass', so 'Fare' might not be a good fit. After analyzing the correlation plot, dropping 'Age' from the input features may strengthen this model. Using 'Pclass' and 'Sex' to predict 'Survived' should produce a strong model.

Data Preparation

For the data to be clean and useable I needed to prepare it. The first thing I did was remove features. I removed the 'Name' and 'Ticket' feature from passengers because it can't be used for any classification or regression models. I removed the 'Cabin' feature due to the data quality being poor. I removed the 'PassengerId' feature because it had no correlation to anything. I then dropped any n/a rows using Pandas dropna() function. Doing so brought the total row count from 891 to 711 rows. I created dummy code mappings for 'Sex' and 'Embarked'. For 'Sex', male was mapped as 0 and female was mapped as 1. For 'Embarked', ports 'Q' was mapped as 0, 'C' was mapped as 1, and 'S' was mapped as 2. I combined 'SibSp' and 'Parch' features into a 'Family' feature and dropped them to keep track of family count in one feature only. I rounded 'Age', so it was no longer a float value. I also dropped any row that had an age below 1 because if the age is less than 1 it is fractional. I then rounded the 'Fare' feature, so it was no longer a float value. With the data prepared, a model can start to take form from the cleaned input features.

Train Model

To train my model, I created an input and output variable. The input variable contained only the mapped 'Sex' feature and the 'Pclass' feature. The output variable contained only the 'Survived' feature. Input is what my model will take in so that it can predict the output. Since 'Survived' is either True or False I went with a classification model. The model that I am most confident with is SKLearn's KNeighborsClassifier model. I used SKLearn's train_test_split() function to split the input and output features into training and testing variables. I then trained my KNeighborsClassifier model by calling the fit() function, passing in the input and output train variables from train_test_split.

Model Validation

To see how well the KNeighborsClassifier model performs at predicting if a passenger will survive based off their 'Pclass' and 'Sex' then validation must be performed. The test score for the model is 0.747 and the train score is 0.767. A good score is considered 75% for a model. If we round the test and train scores, we will get 75% and 77%. This tells us the model is working well. 75% is equivalent to a student getting a perfect C on a test, so it's not the best but it passes the test.

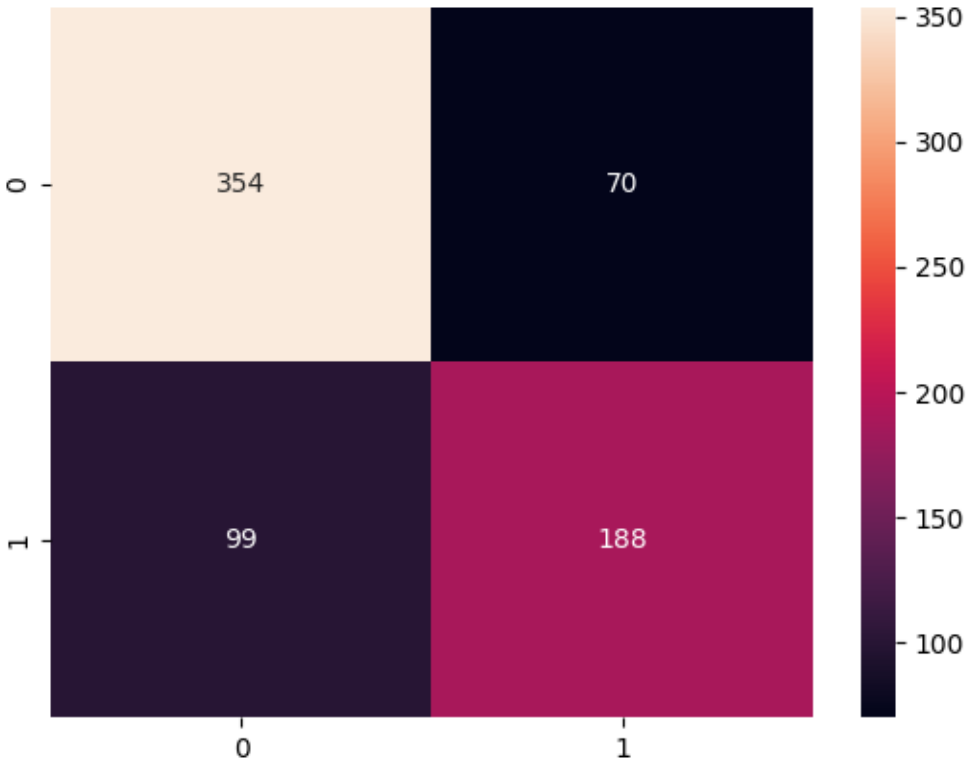
To dive deeper into the model's validation, I printed the classification report. Precision for dead was 79% and 65% for survived. Recall for dead was 85% and 56% for survived. F1-Score for dead was 81% and 60% for survived. The support for the dead was 117 and 61 for survived. Precision shows how often the model was right at predicting. For this model, it was 79% correct at predicting if a passenger would not survive and 65% for surviving. Recall shows the total correct guesses over the total possible correct guesses. Recall had a very strong score of 85% for not surviving guesses but a failing score of 56% for surviving guesses. F1-Score is the average of precision and recall. F1-Score of 81% for not surviving is a B- on a test, showing our model is great at predicting if a passenger will die. F1-Score of 60% for surviving is a D- on a test, showing our model is poor at predicting if a passenger will survive. Support is the number of labels in the data it represents. The poor score for the 'surviving' might be a cause of class balance, since 'not surviving' had 56 more labels than 'surviving'.

Another great validation technique I learned was cross-validation. Cross-validation is used to show robust model evaluation. I decided to do a cross fold of 10. Cross-validation is the process of repeatedly splitting the data into multiple folds. Each fold represents a training set and a test set. The number of folds determines how many times the model is divided into trained and test sets. The scores for using 10 folds are the following:

- 1-fold: 76%
- 2-fold: 67%
- 3-fold: 76%
- 4-fold: 83%
- 5-fold: 74%
- 6-fold: 71%
- 7-fold: 74%
- 8-fold: 73%
- 9-fold: 81%
- 10-fold: 78%

These scores show how well the model performs on different variations of the data. Taking the average of these scores will give an estimate of how well the model performs for unseen data. The average of doing the cross-validation with 10 folds gives the score of 73% resulting in a C-, slightly below a good score.

Lastly, I created a confusion matrix. A confusion matrix shows the quality of a prediction by showing what the model picked when it was incorrect. If the diagonal is large, then the model is good at predicting.



Analyzing this confusion matrix, it is good at predicting if a passenger will not survive. The number 354 says that when it predicted a passenger as not surviving, that passenger was not surviving. The number 70 says that when it predicted a passenger as not surviving, that passenger was surviving. The number 99 says that when it predicted a passenger as surviving, that passenger was not surviving. The number 188 says that when it predicted a passenger as surviving, that passenger was surviving. This is another great form to validate how well the model performs at making the predictions. Clearly this model lacks the ability to predict if a passenger survives but is good at predicting if a passenger does not survive.

Conclusion

Using validation techniques, this model is okay but not good at predicting if a passenger will survive based off their passenger class and sex. This could be from an issue with class balance or the data not having a strong relationship with one another. Another factor could be from where the data came from; this data is from a real-world tragedy. The luck of a passenger could have affected if they would have lived or survived that day, no matter their sex or class on the ship. Overall, this model somewhat matches the movie. This model predicts that females in classes one and two will survive, but class three will die. It predicts that males in class one will survive, but classes two and three will die. These predictions line up with how they let only female passengers onto the emergency boats, but those passengers in class 3, male or female, were locked out from ever getting a chance at survival.