Miguel Sayao
John Lorenz
Aaron Herrera
Tanner Sundwall
CS 445/545

Overview

For our group project, we compared the results of a multilayer perceptron to the results of a support-vector machine utilizing Titanic-related data found here: https://www.kaggle.com/c/titanic.

Of the estimated 2,224 passengers and crew aboard, more than 1,500 died. It only carried enough lifeboats for 1,178 people.

Capacity:

- Passengers: 2,435
- Crew: 892
- Total: 3,327 (or 3,547 according to other sources)

The Data

The data set is made up of the following attributes of each passenger:

- PassengerId: Unique ID of the passenger
- Survived: Survived (1) or died (0)
- Pclass: Passenger's class (1st, 2nd, or 3rd)
- Name: Passenger's name
- Sex: Male or Female
- Age: Passenger's age
- SibSp: Number of siblings/spouses aboard the Titanic
- Parch: Number of parents/children aboard the Titanic
- Ticket: Ticket number
- Fare: Fare paid for ticket
- Cabin: Cabin number
- Embarked: Where the passenger got on the ship (C = Cherbourg, S = Southampton, Q = Queenstown)

This data set uses the information of 891 passengers for training, and 418 for testing.

Although it was a concern whether or not we would have labels for the testing data when we originally proposed the project, it turns out that this particular Kaggle competition started back in 2012, so despite it being against the rules, if someone looks hard enough, they can find the test

data with all the correct labels. Originally, we were going to submit it in the format Kaggle wanted, forego the confusion matrix, and just utilize whatever accuracy number they report to us, but this proper testing data allowed us to test our algorithms exactly how we did for the previous programming assignments.

Something we also didn't really anticipate was exactly how useful some of the data would be as is. With the previous programming assignments, the farthest we've gone in terms of manipulating data was with programming assignment 2 when we had to split the data between training and testing. With this Titanic passenger data set, we have attributes that don't seem all that useful like the passenger's name, ticket number, and cabin number. The thing about these attributes is that some of them contain a bit of useful information that must first be extracted, such as titles in a person's name (Mr., Mrs., Master., Rev., Lady., etc…), a non-unique set of letters preceding the ticket number (STON, PC, C, A, SC AH, etc…), or a letter indicating the location of the cabin on the ship itself which precedes the cabin number (A, B, C, D, etc…).

After extraction, the ticket number and cabin number data are easy enough to feed into an MLP, but the passenger's title extracted from their name is still personal, so we can group similar titles together. For example, Capt., Col., and Major. are all similar enough, so we group those together.

We utilize a feature of pandas.Dataframe called get_dummies that allows us to automatically create new columns for each of these non-numerical attributes, meaning we transform a table that looks like this:

into a table that looks like this:

Ticket_?	Ticket_A4	Ticket_A5	Ticket_AS	Ticket_C	Ticket_CA	Ticket_CASOTON
0	0	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	0	0	0	0
...	...	...	...	...	...	...
1	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	0	0	0	0

All it's doing is taking each possible attribute, creating a new column for each of them, and placing a 0 if false, or a 1 if true. We need this type of manipulation in order to turn non-numerical attributes into numerical attributes, allowing them to be fed into whatever model we want to use them as input for.
What we've also done is made sure that both the training and testing sets produce the same columns regardless of whether or not any of the passengers actually have that attribute. Without this, we end up with 56 columns of data for the training set, and 52 columns for the testing set. This happens because the two sets aren't guaranteed to have at least one passenger for each title, ticket, cabin, etc…

Some passengers are also missing information. For attributes that are numerical like age and fare, we can simply replace these NaN values with the mean of the entire data set. For attributes that aren't numerical, we add a '?' column to indicate that attribute is missing for that particular passenger, as can be seen in the diagram above.

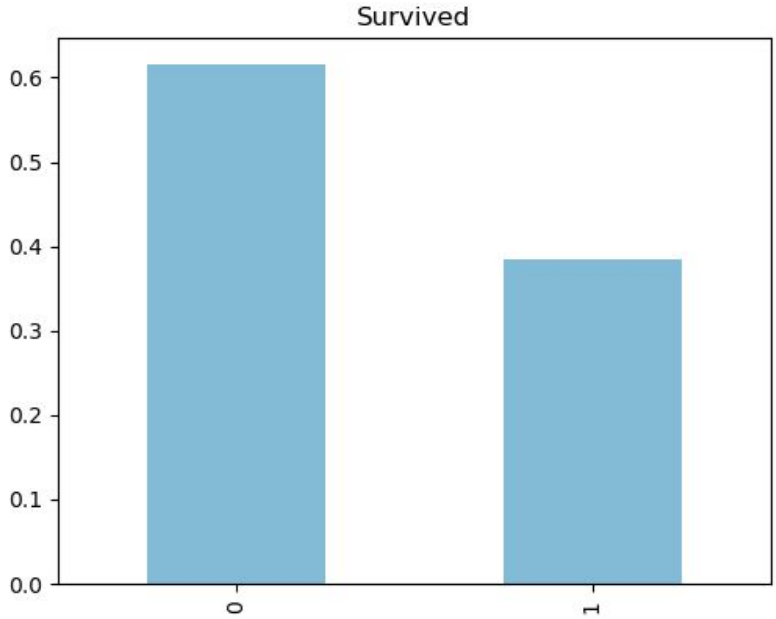
Determining What Input To Use

As was mentioned, there are 56 columns of data for the training set, and 52 columns for the testing set. When the columns that are missing from both are added, we get a total of 62 columns broken down like so:

Attribute	**# of columns**
Pclass	3
Title	5
Sex	1
Age	1
SibSp	1
Parch	1
Ticket	37
Fare	1
Cabin	9
Embarked	3
Total	62

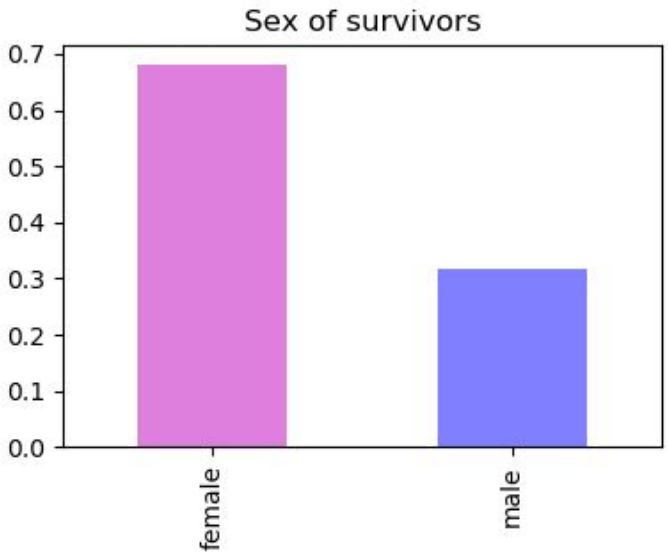
This is a lot of data per passenger, which means we needed to sift through it in order to figure out which of it was actually useful in determining whether or not a passenger survived. We tried throwing everything we had at various training algorithms, and nothing really produced satisfying results, so we started looking into what the input's relationship with the output was before dealing with any algorithms.

One of the first thing's we found was the overall percentage of those that didn't survive compared to the amount that did:

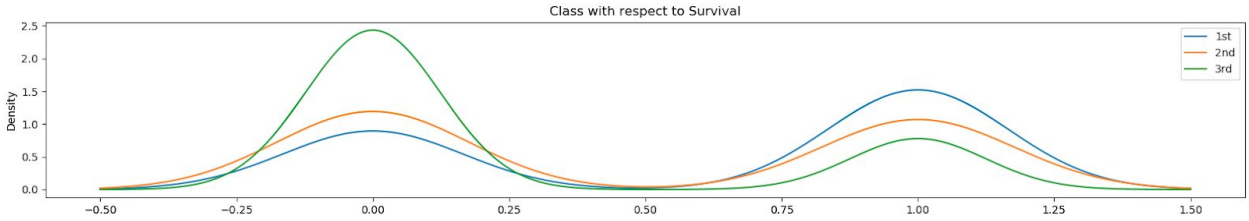


If roughly 60% of the passengers in the training data didn't survive, one could obtain an accuracy of 60% simply by guessing that every single one of the passengers didn't survive. Ideally, one would then hope to achieve an accuracy higher than 60% after going through all the effort of extracting this data and training a model against a machine learning algorithm.

We also took a look at the role the sex of the passengers might have played in their survival. With the survival of women and children being the priority during the evacuation of the Titanic, 74% of the women, 52% of the children, and 20% of the men survived.



From our training data set, we can see a similarly large gap between the sex of those who survived, with nearly 70% of the survivors being women. If we simply told the model to look at the sex of the passenger and predict that they would survive if female, and not survive if male, we would find that we'd have a rather high accuracy.



With the x-axis of the above graph being survival (0 = dead, 1 = alive), we can also see a predictable correlation between passenger class and survival. A passenger's chance at survival is higher if they're of a higher class, and dramatically lower if they're of a lower class. Who would've thought that the rich would be prioritized over the poor during an evacuation? Certainly not Leonardo DiCaprio.

Getting Results

Due to time constraints and the unforeseen amount of effort it took to deal with all the data in this data set, we've decided to fall back on using the Scikit-learn library in order to compare various machine learning algorithms and their effectiveness.

Utilizing all 62 columns we have for input provides us with the following results:

62 Columns		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	68.69%	66.51%
Gaussian NB	46.24%	41.63%
Multi-Layer Perceptron	88.55%	75.12%
Perceptron Learning	77.44%	70.1%
Decision Tree	99.21%	72%

Simplifying this to PCA provides us with the following results:

62 Columns (PCA)		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	68.01%	66.51%
Gaussian NB	66.78%	65.31%
Multi-Layer Perceptron	69.02%	66.03%
Perceptron Learning	66.55%	65.55%
Decision Tree	99.21%	66.27%

Utilizing just the 3 ‘Pclass’ columns and the 5 ‘Title’ columns, we received the following results:

3 ‘Pclass’ columns and 5 ‘Title’ columns		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	78.9%	77.27%
Gaussian NB	78.23%	75.12%
Bernoulli NB	78.9%	77.27%
Multinomial NB	78.9%	77.27%
Multi-Layer Perceptron	78.9%	77.27%
Perceptron Learning	75.31%	74.88%
Decision Tree	78.9%	74.88%

Simplifying this to PCA provides us with the following results:

3 ‘Pclass’ columns and 5 ‘Title’ columns (PCA)		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	78.9%	78.23%
Gaussian NB	78.9%	76.56%
Bernoulli NB	72.95%	66.27%
Multi-Layer Perceptron	78.9%	76.56%
Perceptron Learning	66.78%	59.33%
Decision Tree	78.9%	78.23%

Utilizing every column but the 9 ‘Cabin’ columns, we received the following results:

All columns but 9 ‘Cabin’ columns		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	68.69%	66.51%
Gaussian NB	46.24%	41.63%
Multi-Layer Perceptron	88.86%	77.99%
Perceptron Learning	76.54%	71.05%
Decision Tree	98.88%	73.68%

Simplifying this to PCA provides us with the following results:

All columns but 9 ‘Cabin’ columns (PCA)		
Algorithm	**Train Acc.**	**Test Acc.**
Support Vector	68.01%	66.51%
Gaussian NB	66.78%	65.31%
Multi-Layer Perceptron	68.69%	65.55%
Perceptron Learning	68.01%	65.31%
Decision Tree	98.88%	66.75%

Decision Trees

The decision trees produced from this data are far too large to fit on this paper, but if you're really interested, here's an Imgur link:

- All 62 columns, no depth limit: https://i.imgur.com/lTHGyRi.png
    - Train = 99.21%
    - Test = 72%

Essentially, what we can gather from this decision tree is the importance of 'Title_Mr' is that it immediately takes the 891 passengers, splits them between 374 who do, and 517 who don't have this attribute. It then almost immediately cares about 'Pclass' and 'Fare' due to the correlation between being upper class and being more likely to survive. The rest of the data there is very interesting to look at if you have the time.

Another decision tree can be found in the Imgur link below:

- No 'Title', 'Ticket', or 'Cabin' columns, depth limit of 7: https://i.imgur.com/3v6KbFp.png
    - Train = 87.88%
    - Test = 75.6%

What we can gather from this decision tree is that we once again prioritize 'Sex', even when we don't have passenger titles to fall back on. Due to the high chance of surviving as a female, we can see heading down the right side of the tree that we immediately begin caring about 'Pclass', 'Age', and 'Fare'. For males, we care about 'Age', 'Pclass', and 'SibSp'.

What does all of this mean?

All this means that we were very privileged to be given the data sets we were given for the programming assignments in this class, because it's a scary world out there when you're given data that by itself, cannot be used to determine much of anything, and even after it's modified, you don't really know if it's useful at all.