Machine Learning Engineer Nanodegree
Capstone Project

Haitham Alhad Hyder

April 1, 2020

1 Definition

Note:*(approx. 1-2 pages)*

Project Overview

The project involves the infamous Titanic sinkage of 1912. The data set and inspiration to solve the problem from the Titanic challenge Kaggle competition (Kaggle n.d.).

Only 1502 of the 2224 on board the ship survived and was a big tragedy since it was labelled the "unsinkable" ship. We will be building a model that will identify the chance of survival a person has.

1.1 Problem Statement

We will be building a model that inputs the characteristics of a person and outputs a value between 0 and 1 giving us the chance of someone surviving.

The model that we come up with should be capable of identifying patterns that affect the chance of survival and therefore, by rounding the probability it outputs we can tell if that person's prediction is survival or not.

The steps to solving this problem are as follows:

1. Loading up the train data and split it into a train, validation and test data sets.
2. Build an XGBoost binary linear classifier as well as a PyTorch neural network.
3. At the same time we build an SKLearn decision tree which will be our Statistical base model that will help us compare it to our ML(Machine Learning) models.
4. After evaluating all the models and picking the best one, we will deploy to an API endpoint.

5. Finally we will create a web page that communicates with our model; sending in a user's chosen passenger characteristics and outputting the probability of survival.

1.2 Metrics

The main criteria that would help us evaluate the models we create is accuracy.

$$\text{accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{dataset size}}$$

Since we are not extremely concerned with neither false negatives nor false positives, using accuracy as the chosen metric is much better than recall or precision. We only want to know out of the total inputs what percentage of them did we label correctly.

2 Analysis

Note: *(approx. 2-4 pages)*

2.1 Data Exploration

The train csv file which is the only one we will use since the test csv file, doesn't contain labels making it useless in evaluating our models. It contains data on 891 passengers.

Data columns		
Variable	**Definition**	**Key**
Survived	*Label* if a person survived or not	0 = No; 1 = Yes
Pclass	Ticket class *Proxy for socioeconomic status (SES)*	1 = 1st (upper), 2 = 2nd (middle), 3 = 3rd (lower)
Sex	The gender of the person	
Age	The age in years	
SibSp	# of siblings and spouses aboard *Note:* Sibling = (brother, sister, stepbrother, stepsister) Spouse = (husband, wife)	
Parch	# Parents and children aboard *Note:* Parent = (mother, father) Child = (daughter, son, stepdaughter, stepson)	
Fare	Passenger fare	
Embarked	Port of embarkation	C = Cherbourg, Q = Queenstown, S = Southampton
PassengerId	A unique number for each person	
Name	The Name of the passenger	
Ticket	The unique ticket number	
Cabin	Cabin number	

Table 1: The variables within the data set

The columns that are not bolded have various problems such as having many null values (e.g. Cabin) or not being relevant in solving the problem (e.g. PassengerId) and will not be used to build the features of the models.

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

Table 2: Statistics of the data before cleaning and processing

Table 2 gives us interesting insights into the data. The mean of survived tells us that most people among the 891 didn't survive. Since the mean is much closer to 0 while the median is at 0.

The average age of the passengers we have is around 30 years old.

2.2 Exploratory Visualization

To visualize the data we first have to clean the data set as discussed in 3.1.

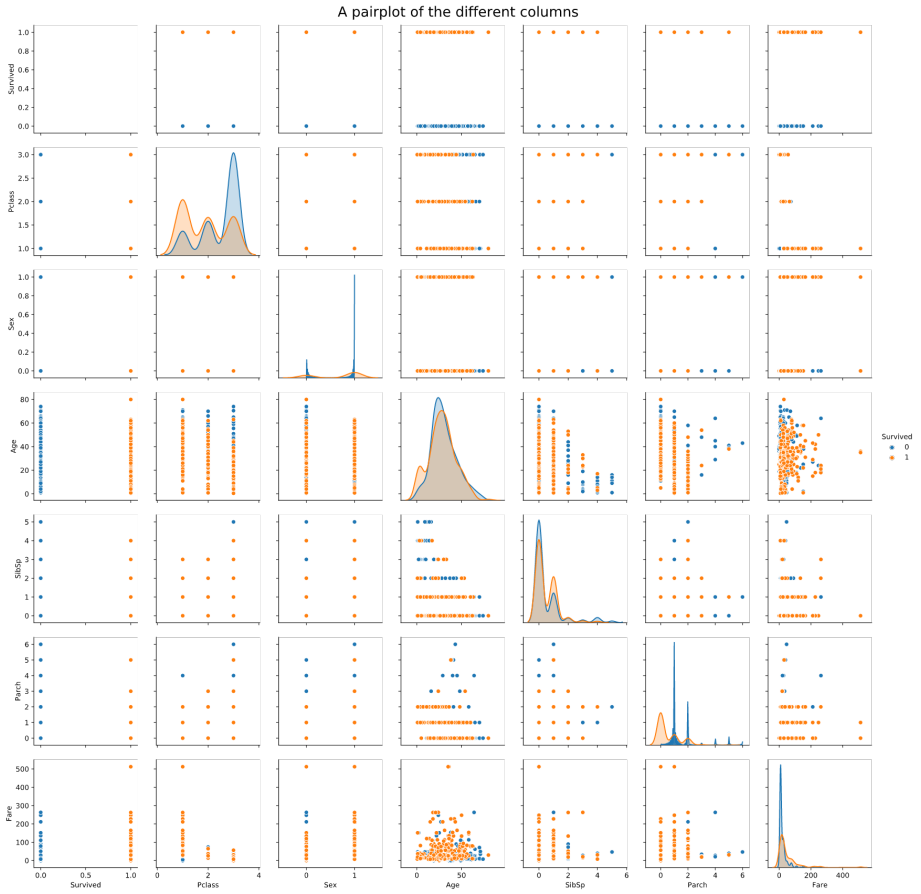

Figure 1: A pair plot of the cleaned columns that helps us to start identifying patterns of features that might increase chances of survival

The figure 1 plot suggests that the lower *Pchar* value a person has, the higher their chances of survival.

2.3 Algorithms and Techniques

The modelling algorithms that we use are: sci-kit learn decision tree, XGBoost binary classifier, and a PyTorch Neural Network.

All the following models take in the 6 inputs we identified in table 1.

The decision tree identifies how the data is segmented leading to a specific label (0 or 1 [Survived]). We will use the default values parameters except max_depth that we set to 10.

The XGBoost algorithm builds a logistic function model that gives us the probability between 0 and 1. All the hyper-parameters were left at their default values except the following:

- max_depth=10 [matching it to that of the decision tree]
- eta=0.2
- gamma=4
- min_child_weight=6
- subsample=0.8
- silent=0
- objective='binary:logistic'
- early_stopping_rounds=10
- num_round=500

While the PyTorch model produces a single output that is also a value 0 or 1 and the following hyper-parameters:

- "input_features": 6
- "hidden_dim": 30
- "output_dim": 1
- "epochs": 250

2.4 Benchmark

The manner that we evaluate the different values is using the accuracy of the multiple models. The Sci-kit learn which is our base model, has an 80% accuracy. Therefore, the goal was to get an accuracy higher than 80%.

3 Methodology

Note: *(approx. 3-5 pages)*

3.1 Data Preprocessing

For processing we begin by cleaning the data then splitting it into training, validation and test data sets. To clean the data we:

1. first, map the gender values to the following: male = 0, female = 1.
2. then, only pick the columns (the bolded variables in table 1) we require
3. and finally drop all the rows with null values.

During inference using the web app, we prevent users from submitting data for inference, unless it is correct and contains only the variables we require.

3.2 Implementation

In this section, we first had to train the models, then creating the web app and API endpoint.

The sci-kit learn Decision Tree is just a statistical model that doesn't require training.

To Train the XGBoost model, we fit training data and optimize on reducing the validation error rate.

For the PyTorch model, which has the following architecture:

- Fully connected layer taking in the 6 inputs
- Passing it to the 30 hidden Fully connected Layers
- And outputting a single a value

The loss function that we use is the BCELoss and we use the Adam optimizer.

An important update that occurred after implementing the solutions, was that only the XGBoost model produces a probability while the other two models produce a binary value representing survival.

During the application development cycle, we first had to deploy the chosen model, then create a lambda function that invokes the model and finally create an API gateway that will allow us to send and receive data from our lambda function.

A complication that occurred was that the lambda function has to carry out extra computations to save the user's input into a .npy file then send that to our model for inference.

3.3 Refinement

Firstly, we used the sci-kit learn model that had an accuracy of 80%, next, we made a new model, this time being an XgBoost model that yielded and 81% accuracy which is not a great leap forward.

Therefore, going the extra mile and using a neural network in the hopes of it learning the different patterns involved within the data set might yield a higher accuracy and it did. The PyTorch model test set accuracy was 85%.

To improve our already great model, one option was reducing the model complexity and using 20 instead of 30 hidden layers. The accuracy drops to 84% however, recall increases from 70% to 72%. Since we mentioned that recall and precision do not matter as much as accuracy, we stick with our first PyTorch model as the best model so far.

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

3.4 Results

Note: *(approx. 2-3 pages)*

3.5 Model Evaluation and Validation

The final model is a PyTorch Neural Network trained with the hyper-parameters described in section 2.3.

It does a much better job in labelling a person as survived or not more than the other models that we tested.

	SKLearn	XGBoost	PyTorch (30)	PyTorch (20)
precision	0.712	0.926	0.854	0.818
recall	0.740	0.500	0.700	0.720
Accuracy	0.804	0.811	**0.853**	0.846

Table 3: The evaluation metrics of the different models *The 30 and 20 represent the number of hidden layers*

To test the model, we sent in user data through the web app as shown in the section 4.1.

3.6 Justification

The results we found with the 30 hidden layer PyTorch model are much stronger than the benchmark result and therefore, is a much better model to use.

We can definitely send in data that is unreasonable to the model for inference such as negative ages or something else that is weird. To solve this problem, we use form controls in the web app to help the user submit clean and correct input that is reasonable for the model to infer. It is not a perfect solution, since someone might alter the data before it reaches the server, although "better late than never".

4 Conclusion

Note: *(approx. 1-2 pages)*

4.1 Free-Form Visualization

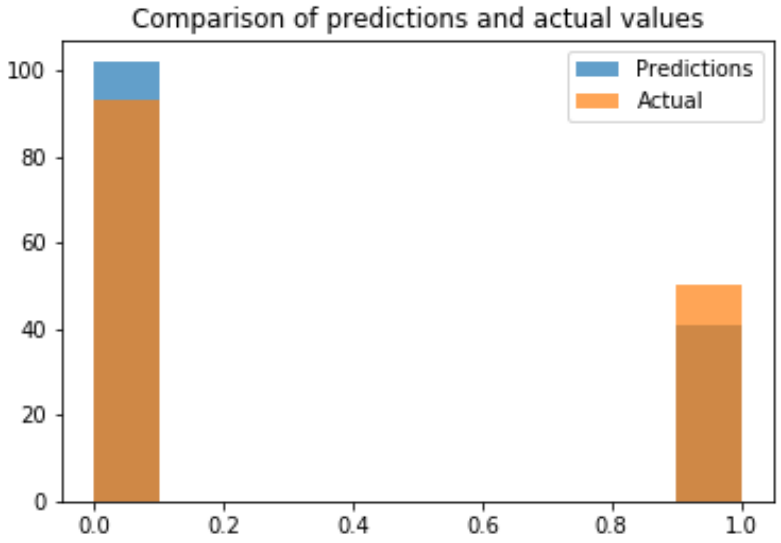

Figure 2: A comparison of the histograms of labels produced as predictions compared to the actual labels of the test data set. It shows that the predicted values exaggerates the chances of someone not surviving and fails to label those who survive as survivors

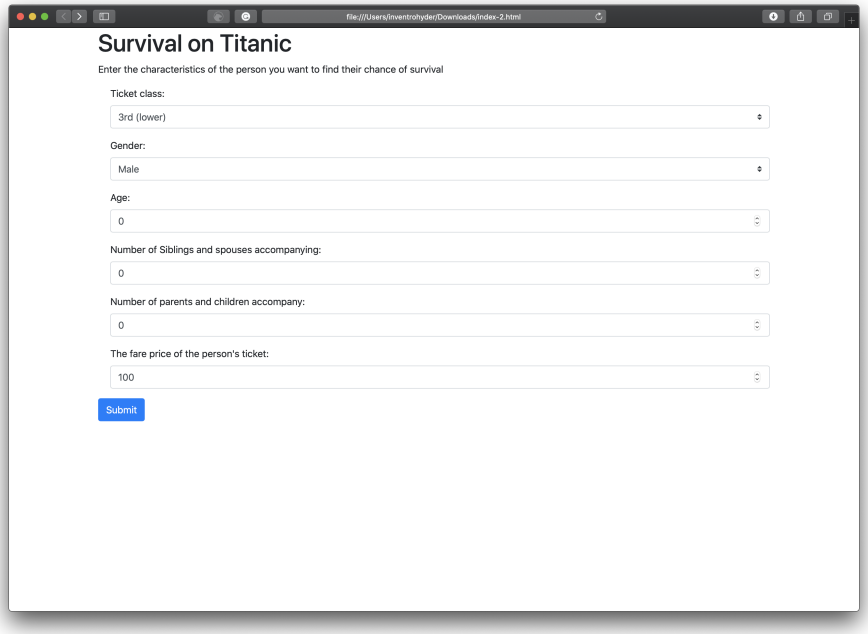

Figure 3: The web app that I made in-order to connect users with our deployed model.

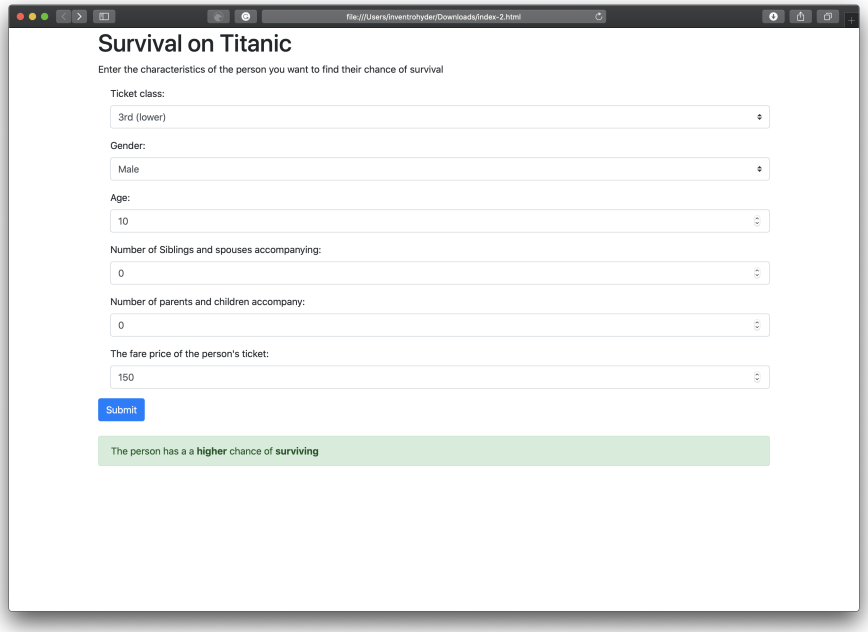

Figure 4: An example of a predicted survival

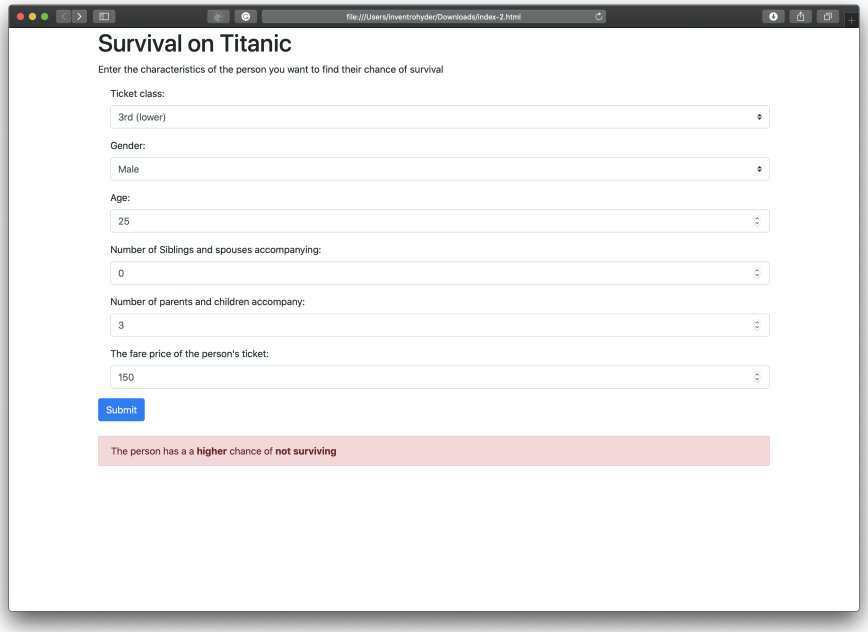

Figure 5: An example of a predicted non-survival

4.2 Reflection

We can summarize the project as follows:

1. Obtaining the data set and understanding the problem
2. Cleaning the data and processing to pick out the features to use
3. Choosing a modelling algorithm and fitting it (training)
4. Comparing the different models using a metric to pick a better one
5. Using the chosen model to make inferences for a web app

One the key challenges that I got was trying to solve the errors that occurred during the entire process and getting much more familiar with cloud watch logs.

I also learnt that I can save Numpy arrays into .npy files which I didn't realize till now.

The final model and web app were able to work and based on the accuracy we got it seems it is a suitable model for this application.

4.3 Improvement

I think there are further improvements that I could have made. One of which would be the use of Hyperparameter tuning training jobs so that I can optimize my models even further.

I haven't studied Deep Learning yet and the concepts that I used in this project were from the tip of the iceberg of an introduction given within the coursework. Therefore, I would love to understand them more deeply and have total insight of them.

If I used the best model we got so far as the new benchmark, I hypothesis that a much better solution does exist. First, through the hyper-parameter tuning jobs mentioned before, as well as through feature engineering which I didn't make the best use of.

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?

References

Kaggle (n.d.). *Titanic: Machine Learning from Disaster*. URL: https://www.kaggle.com/c/titanic/overview.