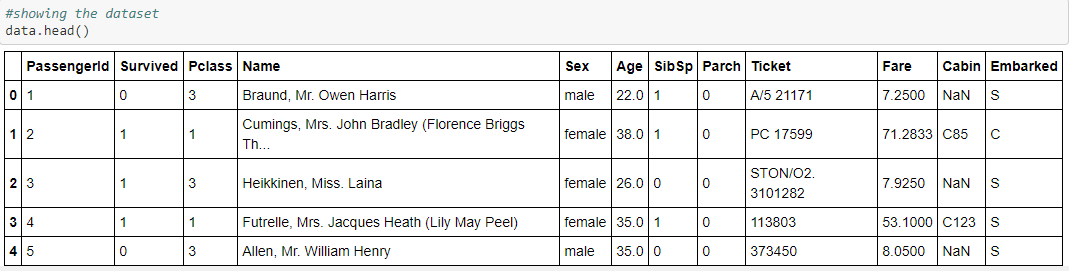
**Final Report**

**Introduction**

The Titanic is one of the most historic terrible accidents that every human alive knows. We wanted to figure out if there is a connection between the people who died and the people who survived this accident based on the dataset. The dataset describes some of the features about the people who were on this ship, it does not include information about that crew, and some information is lost and describes as null.

**Related work and Required background**

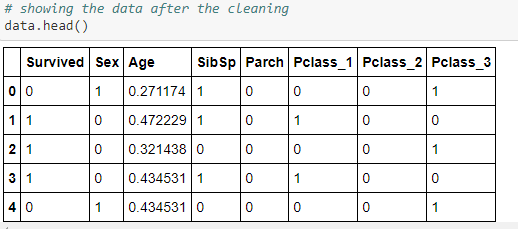
The variables on our dataset are passengerid, Survived, Pclass, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked. Pclass refers to passenger class (1st, 2nd, 3rd), and is a proxy for socio-economic class. Age is in years, and some infants had fractional values. Survived refers to if the passenger survived (0=no, 1=yes). Sex refers to gender (male/female). SibSp is the number of siblings aboard. Parch is the number of parents aboard. Embarked refers to Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton). Ticket refers to the ticket number. Fare refers to passenger fare. Cabin refers to the passenger‘s cabin. The relevant variables are Pclass, Age, Sex, Survived, SibSp, and Parch.



We took our dataset from “Kaggle” website.

**Project description**

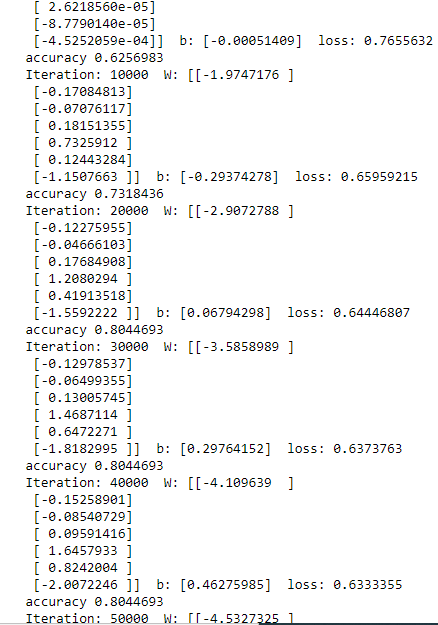
In this project, we want to predict who survived and who did not. We must mention that our dataset is small, and it affected our results. We started by cleaning the data, we dropped the rows with NAN values, and we dropped columns that cannot help us predict if the person survived or not like: ‘Fare’, ‘Cabin’ and ‘Ticket’, which depends on ‘Pclass’ column and ‘PassengerId’, ‘Name’ and ‘Embarked’ which were not helpful for our model. We split the ‘Pclass’ column into three different columns and we named them ‘Pclass\_1’, ‘Pclass\_2’, ‘Pclass\_3’. Then we changed the 'sex' column into 0, 1 (0=female, 1=male) and last we normalized the ‘age’ column.



After we finished cleaning the data, we set up the x and y labels, and then we split them into x\_train, x\_test, y\_train, y\_test (test = 0.25, train = 0.75). In this project, there are 3 parts, in each part, there is 1 model. The models are:” logistic regression”, “using hidden layers”, “long short term memory (LSTM)”. Each model is more complicated than it’s previous, so in theory, we should get higher accuracy.

The first model is the “logistic regression”. We did 100000 iterations and 0.001 learning rate. We can see that after 30,000 iterations the accuracy doesn’t change, so it will be best to stop after 30,000 iterations. In the end, we could see the weight, the bias, the loss, and the accuracy of our prediction.

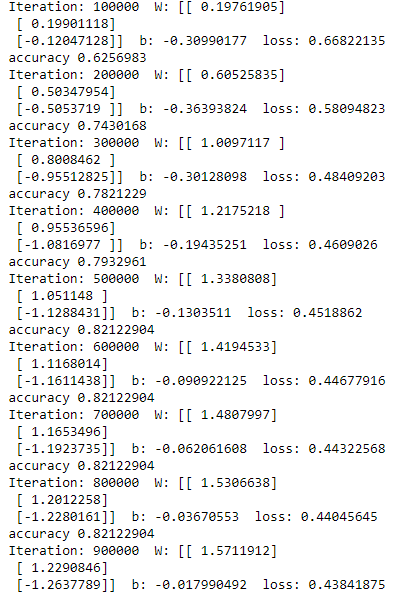
The model had an accuracy of 80% on the testing set after optimization and training the data.



Then we wanted to improve the accuracy of the first model, so we added one hidden layer with 3 nodes and 1,000,000 iterations and 0.001 learning rate after we tried a few times with different numbers we understood those numbers will give us the highest accuracy. We chose to train this model with more iterations since it is a more complicated model.

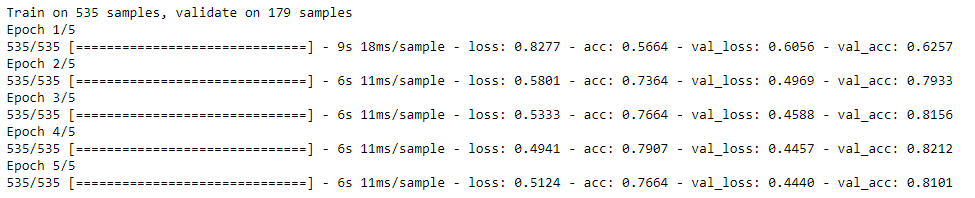
The model had an accuracy of 82% on the testing set after optimization of the network and training the data using GradientDescentOptimizer.

As we can see using a hidden layer after 600000 there is no change the accuracy, so it will be best to stop before. As we can see, compare to the logistic regression model there are improvements in our predictions.



In our last model the LSTM, which the most complicated between the models, should bring us to the highest accuracy. But, surprisingly, after different attempts, the model had an accuracy of 79%.

We used Keras to build our model and set our Dense layer to 5. We used the activation layer “relu” because an activation layer improves the accuracy. Then we compiled the model and train it on the train data with 5 epochs and “adam” optimizer. We can see that after epoch number 4 there is overfitting. As we can see, after the fourth epoch there is overfitting so it will be best to stop at this point.

****

**Experiments**

Normalize – in the begging, we ran all our models without normalizing our data, and after we normalized the age (the only column that is not binary) and split the ‘Pclass’ column, we got better results.

Irrelevant columns – in the begging, we ran all our models with all the columns, and it results in low accuracy, and we chose to drop the irrelevant columns, some columns depended on other columns, so we took only some of them and not all of them. After a unique pick that we did carefully, we got better results.

The number of iteration and learning rate – first we started with a small number of iterations which resulted in low accuracy. Then we tried lots of iterations which caused overfitting. In the end, we found the numbers that will cause the highest accuracy that we can get without cause overfitting.

**Conclusion**

* Because the dataset is very small without too much information and after we cleaned the data it became much smaller. So it is hard to predict in high accuracy who died and who didn’t base on the data.
* Although long short term memory (lstm) is a more complicated model, it didn’t give better accuracy than the simple logistic regression. We suspect that the main reason for this is the fact that our dataset is small.
* We found the numbers that will cause the highest accuracy that we can get without cause overfitting.
* For a future project, we will search for a larger dataset.

For the full Jupyter notebook code, visit - <https://github.com/almogun9963/deep-learning-.git>