**Hunger Games Analysis**

**German E. Baltazar Reyes**

This week’s project evaluated the accuracy and behavior of simple logistic regression models and shallow Artificial Neural Networks (ANN) in classification and regression tasks. We were given different datasets to select and analyze with these algorithms. The chosen dataset consisted of 891 observations regarding the information of the passengers aboard the Titanic. Each statement had 69 features that described their status in the ship and survival.

The use case of this dataset will be based on the novel by Suzanne Collins, *The Hunger Games*. In the first novel of the series, they describe how several people are put against each other in an arena to survive, being the last of them the victor. During these events, the people outside could watch the transmission and bet on the participant they thought might win. So, in this case, we will use the Titanic dataset as one version of The Hunger Games, and our primary purpose will be to determine if a person, given particular characteristics, might survive or not. Also, the fare of their tickets will be interpreted as the amount of money they bid on them during the event so that we will train a second model to predict how much money their bet should be.

The objective values for both tasks will be to know if each person survived and how much their fare was. The first step was removing every repeated row on the dataset, 96 rows. After that, only the following features were contemplated since they were more related to the given use case:

* *Sex (binary)*. Indicates if the person was female (0) or male (1).
* *Age (int)*. Indicates how old each person was.
* *Small family (binary).* Determines if the person had a small family or not.
* *Large family (binary).* Determines if the person had a large family or not.
* *Pclass\_1 (binary).* Determines if the person was in first class.
* *Pclass\_2 (binary).* Determines if the person was in second class.
* *Pclass\_3 (binary).* Determines if the person was in third class.
* *Survived (boolean).* Determines whether the person survived the accident.
* *Fare (float).* The amount of money each person paid for their ticket on the ship.

One advantage of this dataset is that their boolean features were already clean, so the behavior between them is coherent. For example, if a person were already in first class, the dataset would not indicate that they belonged in second or third class either.

The classification of the family size of each passenger was modified to reduce both columns into one. Each observation was labeled into three new categories, as shown in Table I. This classification allowed the creation of a new column, named *fam\_size*, that will indicate the family size of each person with one single class rather than two boolean values. For the case of the class, a similar procedure was followed, classifying each person according to the information shown in Table II and resulting in a new column called *classes*.

After these modifications, the dataset was reduced into a table with 795 observations and 11 features. For most of the analysis, the main goal will be to determine each person's survival. In general, as shown in Figure 1, only 41% of the passengers survived. A deeper analysis of the survival of each passenger was made, comparing that output with the sex, age, class, and family size.

Figure 2 shows how even though there were more men on the ship, most died during the accident. In addition, the second plot of the image indicates that people between 20 and 30 years old had a slightly superior chance of survival; nevertheless, only the children below ten had the best chances of survival.

Table I. Classification of family size

|  | **Small Family** | **Large Family** | **Class** |
| --- | --- | --- | --- |
| *Single* | No | No | 0 |
| *Small Family* | Yes | No | 1 |
| *Large Family* | No | Yes | 2 |

Table II. Classification of person’s class

| **Ship Class** | **Dataset Class** |
| --- | --- |
| First | 1 |
| Second | 2 |
| Third | 3 |

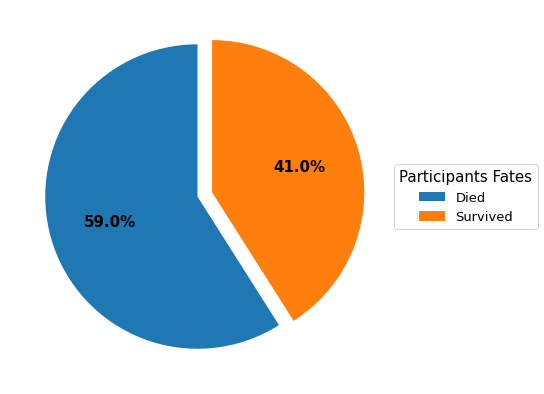
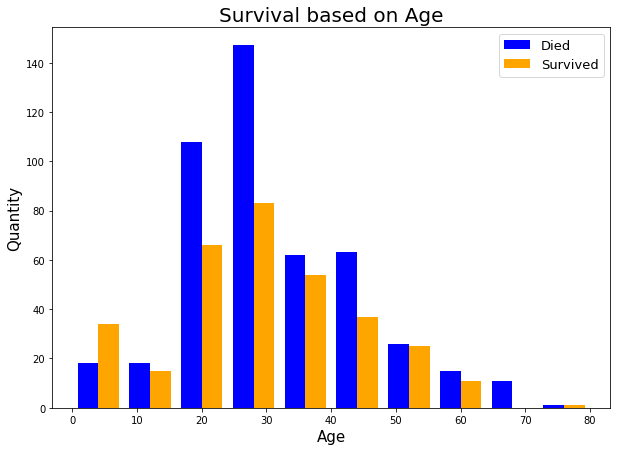
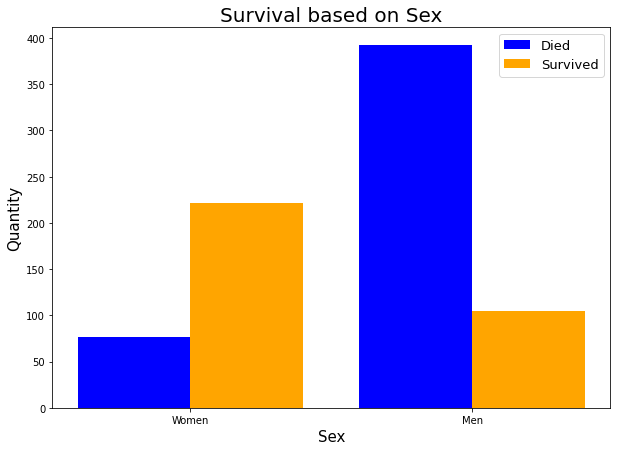
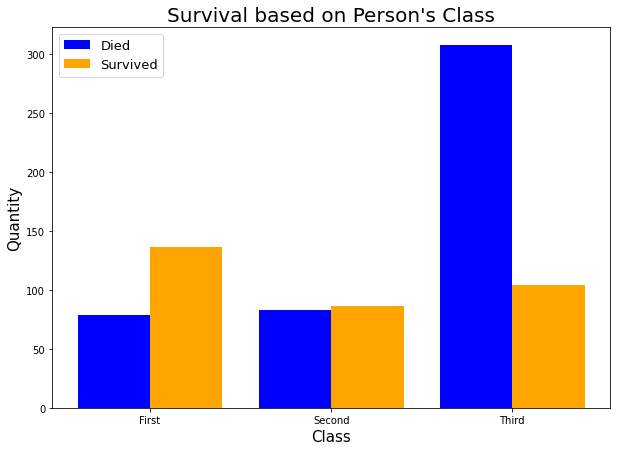
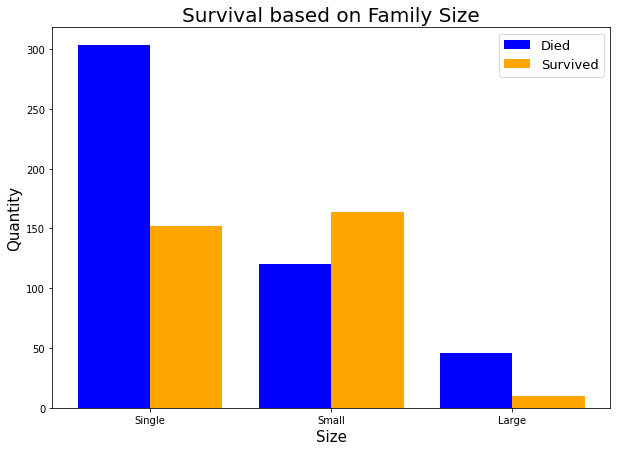


Figure 1. Survival percentages



a) b)

Figure 2. Survival analysis based on a) sex and b) age



a) b)

Figure 3. Survival analysis based on a) family size and b) class

After creating the new features of *fam\_size* and *classes*, it was also possible to observe that the people with a relatively small family had better chances of survival than the ones that were single or had a huge family. Finally, it was proved that the people with a better position in the ship (first class) had more significant chances of survival than those in the lower class (third). The chances of survival were not directly relative to those belonging to the second class. All of these observations are better explained in Figure 3.

Regarding the fares purchased on the ship, Figure 4 shows those between 20 and 40 years old were the ones that spent more money. This could also mean that based on age, the higher one pay, the higher his chances to survive. Finally, it can be observed in Figure 5 that women were the ones who spent the most among the surviving passengers, being the opposite of the ones that died. Of the $11,173.85 paid by those who did not survive, only $1,674.94 was paid by women. However, from the $16,227.38 paid by the survivors, a total of $11,962.57 was paid by women.

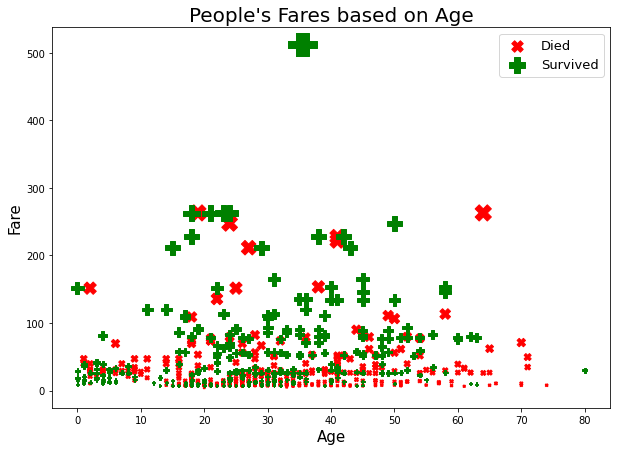


Figure 4. Fare comparison based on the passenger’s age

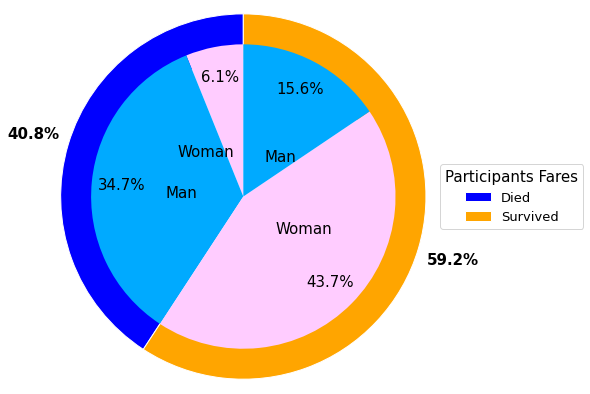


Figure 5. Fare comparison based on sex

Since there are two different objectives: one for classifying survivors and another for predicting the total fare of a person, it is necessary to create two separate models for each task. For both cases, the dataset was distributed into a training, validation, and testing set, following distribution of 70/10/15%, respectively. Each dataset was randomly shuffled, leaving 556 observations for training the models, 120 to validate the models and 119 for testing.

For the classification task, two different models were trained: a Simple Logistic Regression (SLR) model, and a shallow ANN. For both cases, since the classification was of a binary class, the sigmoid activation function was implemented for predicting. In this particular case, the fare’s amount was not considered, since the intention was to determine that quantity based on the survival chances. For this reason, only the age, sex, family size, and class were considered.

The SLR model was finetuned as follows: first, the best number of epochs was determined, varying from 30 different values, between 1000 and 30000 epochs. All of these values were trained using a default learning rate of 0.001. After finetuning, a total of 17000 epochs were selected, resulting in a training accuracy of 79.67%. Table III shows the overall accuracies and hyperparameters of this model, while Figure 6 shows the behavior of the training accuracy.

With the number of epochs defined, the validation set was used to finetune the size of the learning rate. A total of 100 values were tested, from 0.0001 to 0.1, being the best value 0.002118 with a validation accuracy of 79.16%. Figure 7 shows the changes in accuracy based on the size of the learning rate. Once each parameter was defined, the resulting test accuracy was 76.47%. Its loss function is shown in Figure 8.

For the training of the ANN, the training set was used to determine the best architecture for the classification task. The selected architectures were the following:

* [6]
* [4]
* [4,6,4]
* [4,6,6,4]
* [6,4]

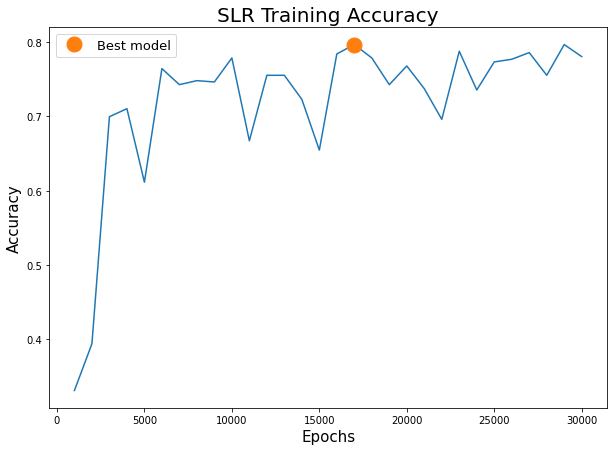


Figure 6. SLR model training accuracy

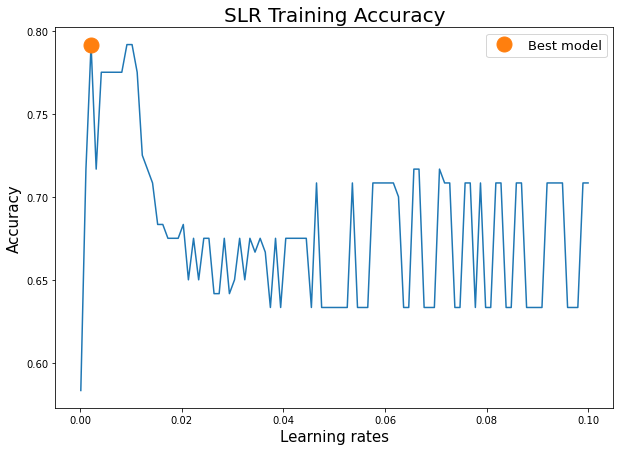


Figure 7. SLR model validation accuracy

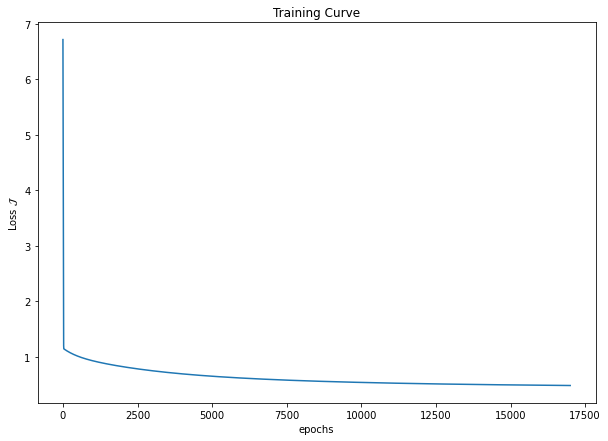


Figure 8. SLR model test loss function

Since the dataset was a small binary classification problem, only sigmoid activation functions were used for every layer. According to Figure 9, the best model was the architecture of one single hidden layer with six neurons, with a training accuracy of 60.25%. The validation set was used to finetune both the learning rate and the number of epochs of the model. For the first hyperparameter, thirty values were selected between 0.0001 and 0.1, while the second feature had ten different values, from 100 to 2000 with a step of 200.

The best model resulted from a learning rate of 0.0001 and 100 epochs, resulting in a 54.16% on the validation set. Figure 10 shows the behavior of this finetuning, while Figure 11 shows the loss function of the final model, with 57.98% on the test set. The general architecture of the final ANN model is shown in Table III. Comparing these two models, it was determined that the best model would be the SLR model since it got a bigger accuracy with its test set.

Table III. Model comparison for the classification task

|  | **Architecture** | **Epochs** | **Learning rate** | **Training accuracy** | **Validation accuracy** | **Test accuracy** |
| --- | --- | --- | --- | --- | --- | --- |
| *SLR* | - | 17,000 | 0.002118 | 79.67% | 79.16% | 76.47% |
| *ANN* | [6] | 100 | 0.0001 | 60.25% | 54.16% | 57.98% |

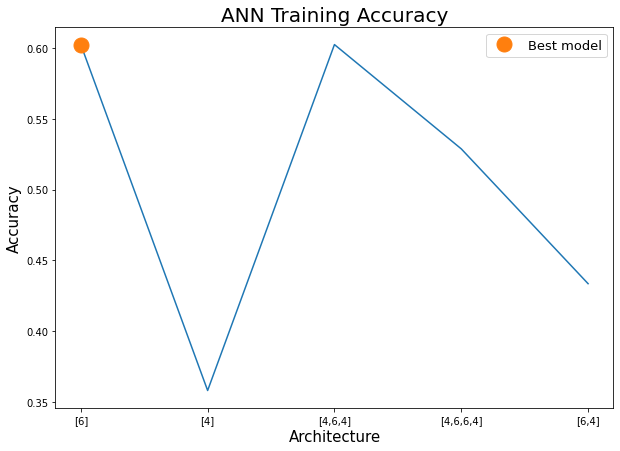


Figure 9. ANN training accuracy

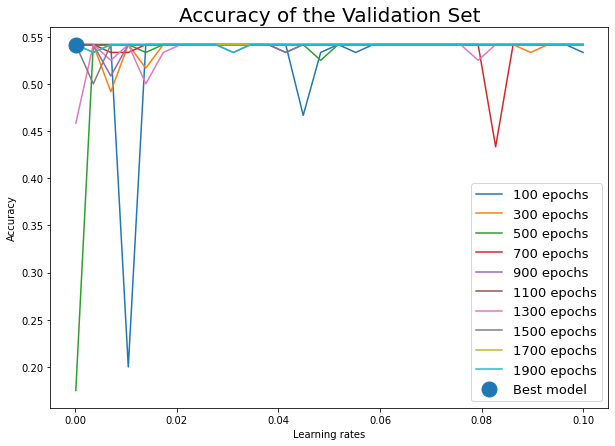


Figure 10. ANN validation accuracy

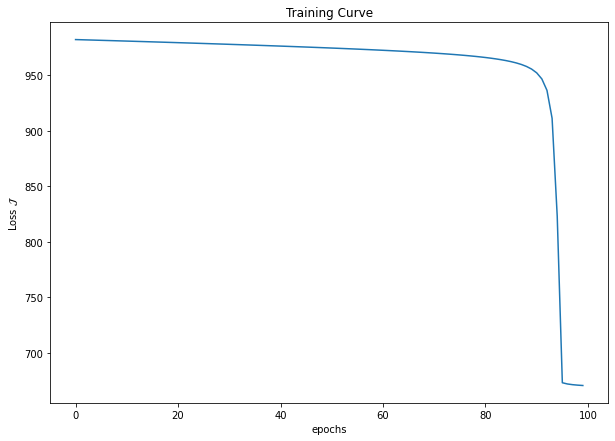


Figure 11. ANN loss function

Once the model for selecting the classification of each person's survival was made, the next step was to create the model that would predict the fare’s amount through a regression model. Since the preliminary analysis of the data showed that there is no direct linear relationship between the fare and the rest of the features of the dataset, it was decided to train only an ANN model to find the best approximation in a multidimensional space (considering a dimensional analysis per hidden layer).

The output label would be the fare column of the dataset, while the rest of the columns were considered for the input. The tuning of the model followed the same structure as the classification model, using the same ranges and architectures. The only difference was for selecting the activation functions, ReLU functions for the first and last hidden layers, while hyperbolic tangent for the middle ones.

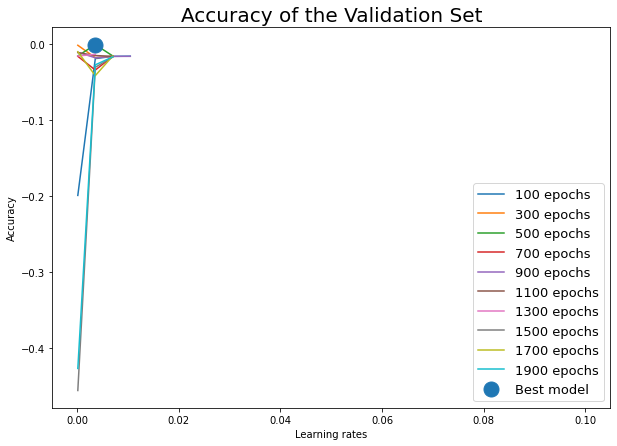
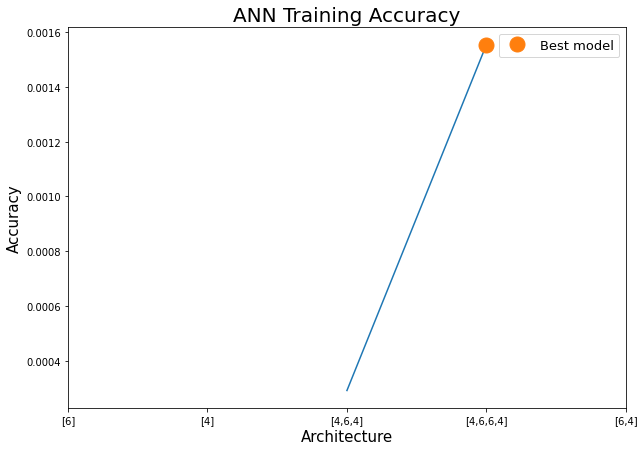
The model's training resulted in selecting an architecture of [4,6,6,4] with its activation functions as [ReLU,tanh,tanh,ReLU]. This training of the model resulted in an R2 of 0.001554. Next, the model’s validation resulted in an R2 of -0.00082 with a total of 500 epochs and a learning rate of 0.0035. Figure 12 shows the behavior on both sets, while Figure 13 shows the loss function of the test set, which resulted in an R2 of -0.00386. Table IV shows the overall structure of the regression model.

In this particular case, it can be observed that the non-linearity of the fares affects the model, as well as the lack of sufficient data. Even though ANNs are one of the most used algorithms for machine learning tasks nowadays, this case demonstrates that, at least, the basic architecture is not capable enough of finding a proper relationship between the fares and the rest of the information of the passengers.

Although the resulting models generated positive results on the classification task, their accuracy is relatively small. This behavior is expected since the dataset has not had much data, making it difficult for the models to generalize well enough with new information. On the other hand, and considering a more realistic case of use, this application could be very beneficial for insurance companies that evaluate if a person is prone to require a higher amount of protection (economically speaking). This application could not only benefit these companies by saving money but also the clients will have a better understanding of how probable it would be that they may be in more danger during a particular event: in this case, boarding a ship.

Table IV. Regression R2 comparison

|  | **Architecture** | **Epochs** | **Learning rate** | **Training R2** | **Validation R2** | **Test R2** |
| --- | --- | --- | --- | --- | --- | --- |
| *Regression ANN* | [4,6,6,4] | 500 | 0.0035 | 0.001554 | -0.00082 | -0.00386 |



a) b)

Figure 12. Regression ANN accuracy for a) training and b) validating

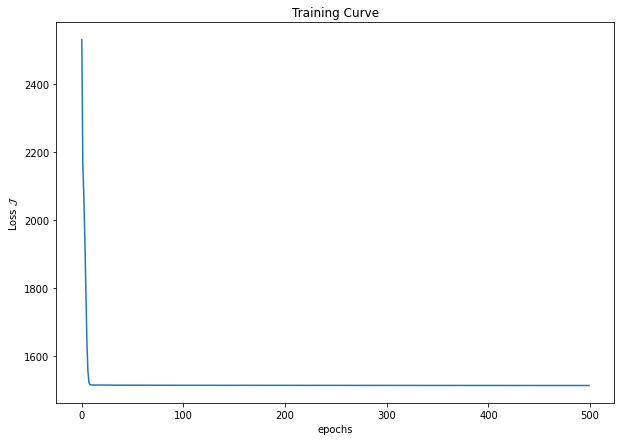


Figure 13. Regression ANN loss function