**Logistic regression to determine titanic survival**

Alan Antonio Macías López

**Introduction**

The Titanic accident occurred on April 15, 1912. The iceberg collision made the Titanic sink in approximately two and a half hours which was relatively fast for a ship of such size.

The ship was carrying 2224 people of all ages, genders and classes that night, 1514 passengers died in the icy waters and only 710 escaped in lifeboats. It is said that the chances of surviving of a passenger where influenced by several factors such as age, gender, class, etc…

The purpose of this report is to use a statistical model called logistic regression to determine if a passenger survived given some information about him or her.

**Logistic Regression**

Logistic regression is a statistical model used for classification problems. The most common type of logistic regression is binary, which is used to solve problems where the dependent variable (label) can land in two different categories, like checking if a tumor is malignant or not, determining if an email is spam or not, or in this particular case, predicting the survival of a titanic passenger.

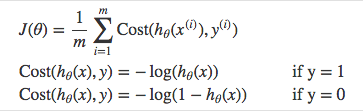
**Predictions**

For each sample (passenger) we will make a prediction to determine if he or she survives or not. In order to do this, we first make the hypothesis using the linear function. Where each x is an attribute of the passenger, each m is the parameter that affects that attribute and b is the bias.

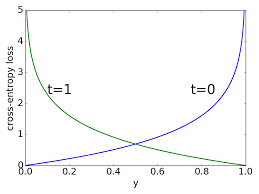
Once the hypothesis is calculated we apply the sigmoid function , which will allow us to get a value that strictly ranges from 0 to 1, so for example if we get 0.65 as a result of the sigmoid function for one passenger, we can say that that passenger has a 65% probability of surviving.

**Cost Function**

The cost function allows us to know how badly the model is performing, and therefore allowing it to improve over time. We will use cross-entropy as the cost function for this model. This function can be divided in two parts depending on the value of.



The advantage of doing this is clearer when you look at both functions graphed.



This combined function makes it easier to calculate the gradient and minimize the cost. Since there is only one minimum, it also solves the problem of landing at a local minimum.

**Gradient descent**

It is the process of following the gradient of the cost function to land at its minimum, in order to have the smallest possible error. For this implementation we will use stochastic gradient descent, which updates the parameters every iteration to minimize the error. To update the parameters we will use the following formula.

Where: m = parameter to update

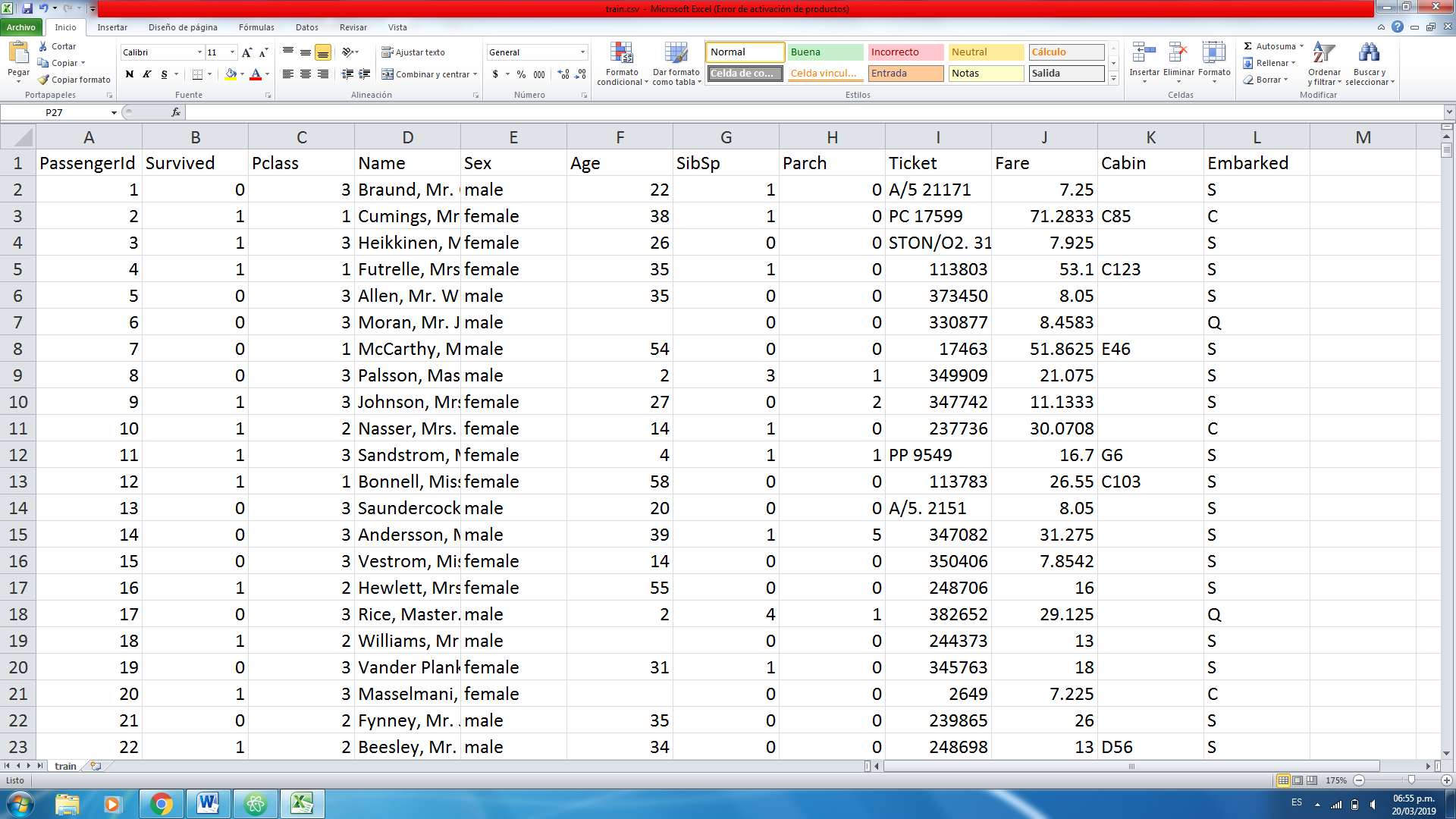
= learning rate, which limits how much the model can learn per iteration.

x = sample of the dataset

h(x) = Predicted label of the sample x

x[i, j] = attribute that is affected by the parameter

**Titanic dataset**

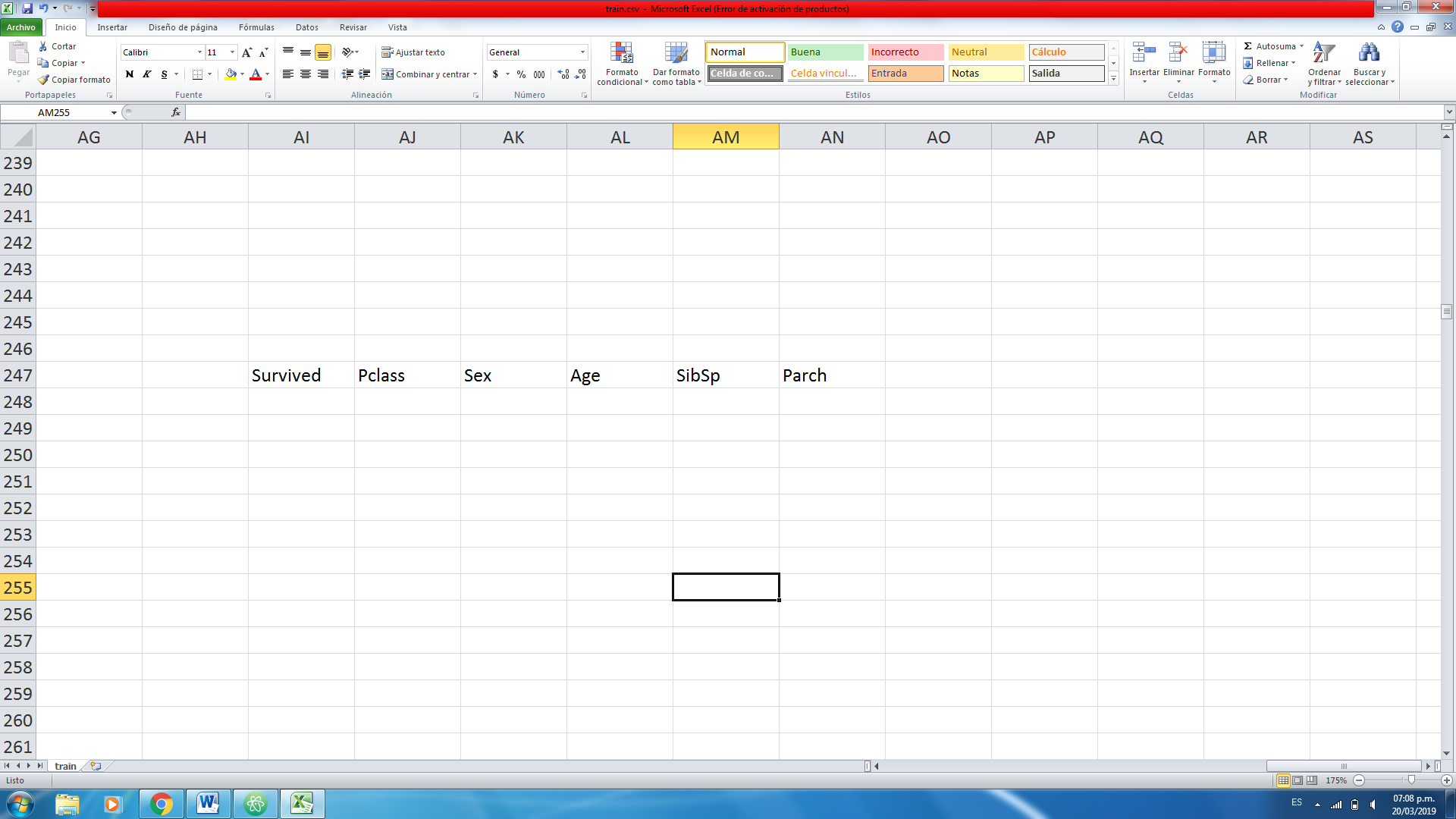


These are the first samples of the titanic “train.csv” dataset. We will preprocess the whole dataset to solve some problems so that our model works properly.

**Data preprocessing**

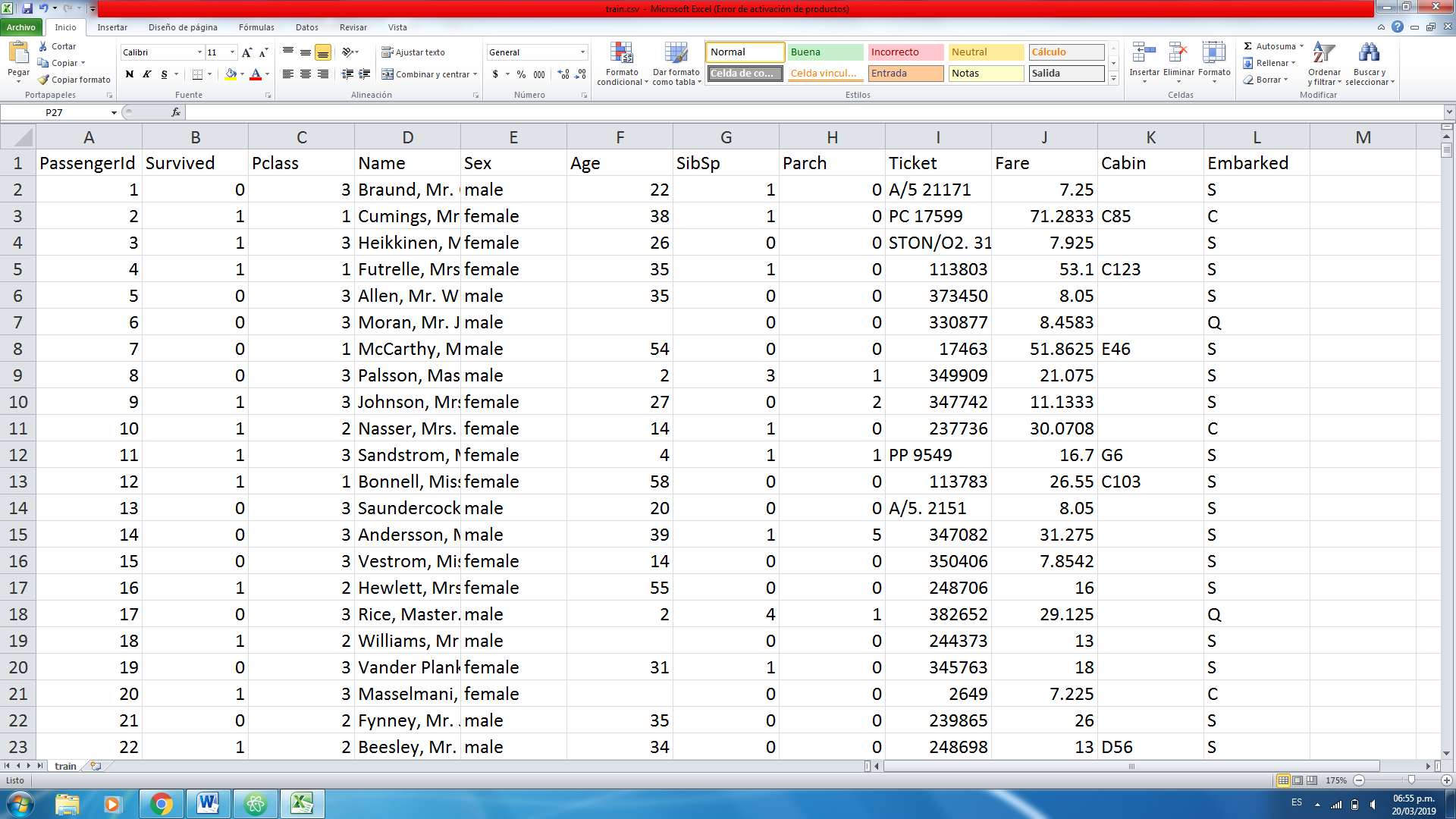
Problem 1: Unnecessary fields

Before anything else, we must ask ourselves: ¿What passenger attributes may influence the chances of surviving? There are many ways of answering this question, but for this implementation these are the fields that will be considered as relevant, the rest will be removed.



It is important to mention at this point that the “Survived” column does not contain an attribute, but the real label of each sample, so 1 = survived and 0 = did not survive. We will use this value to get the error of each prediction and help the model learn when doing gradient descent.

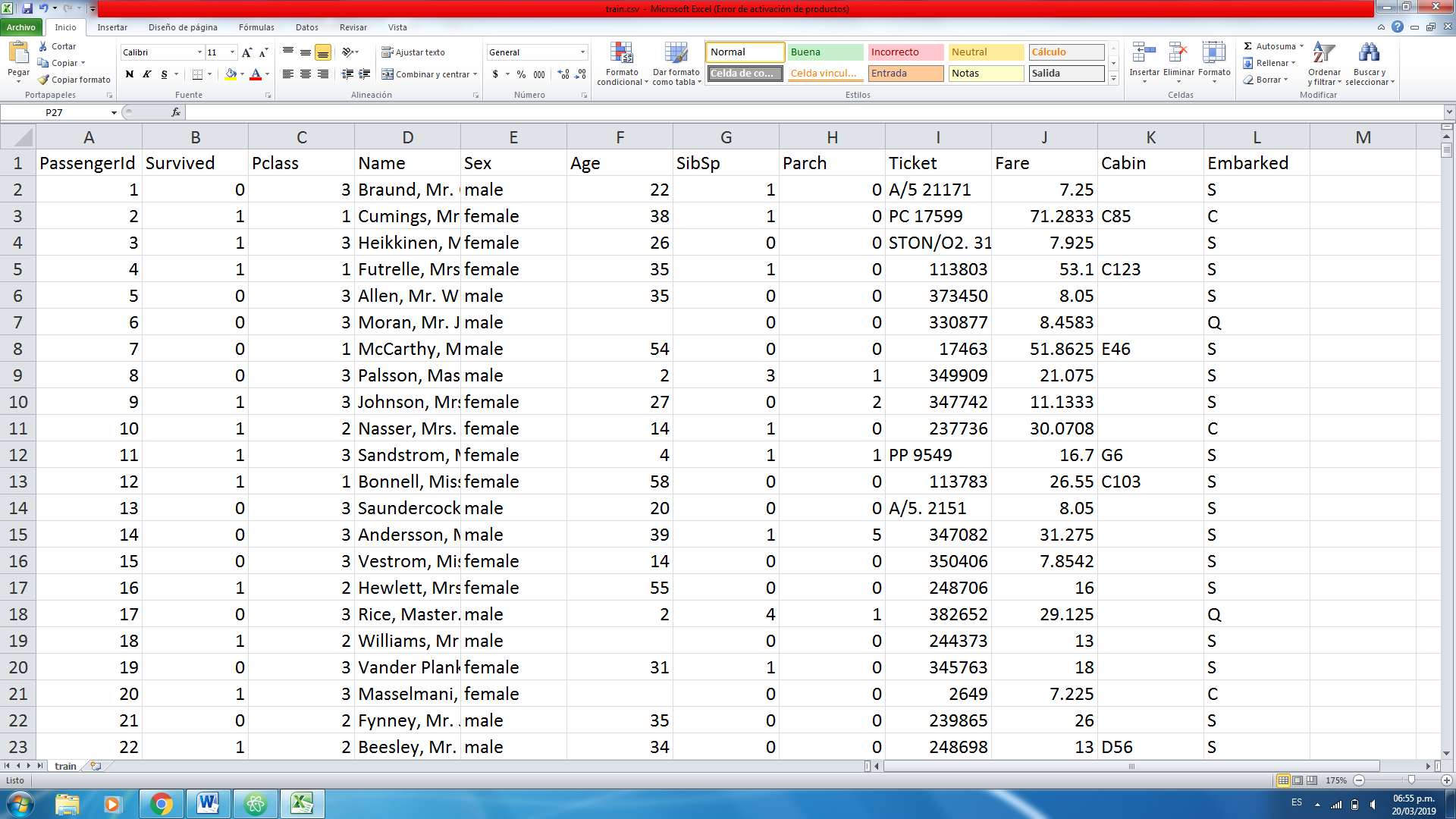
Problem 2: Not all values are numeric



If we leave the column “Sex” as it is, our code will crash when trying to make operations with the values ‘male’ and ‘female’ to update the parameters. So a very simple and straightforward solution is to replace ´male´ with 0 and ´female´ with 1.

Problem 3: Some values are empty

As the previous problem, our code will crash when trying to make operations, but this time because the value is empty. There are several solutions for this problem; one of them is to manually assign the value to be equal to the average of the rest of the samples, but to keep things even more simple, we will just leave the value as an empty string, and when we want to update a parameter, we will first check if x[i, j] is not an empty string so that the program does not crash.



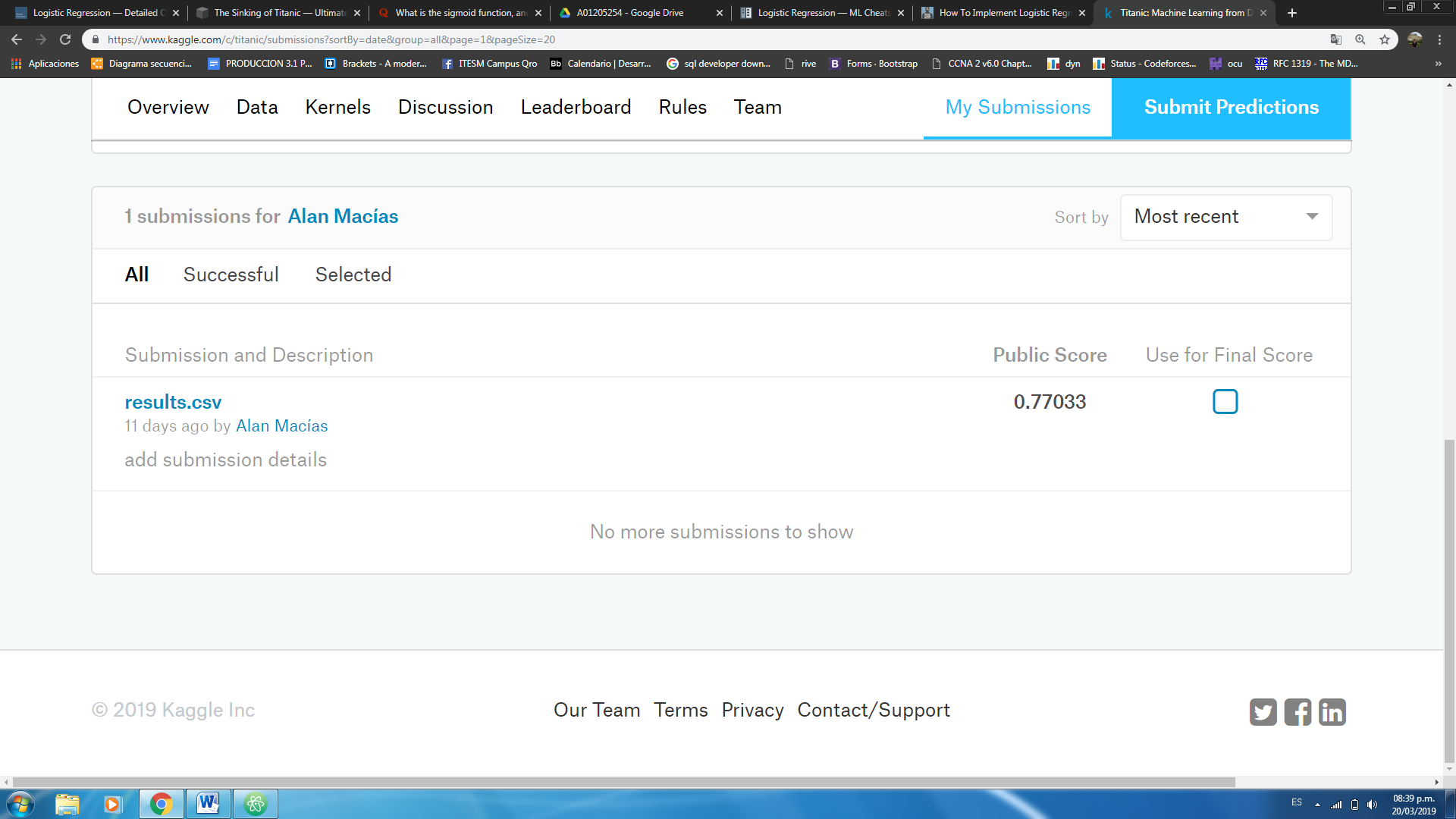
**Training the model**

We will update our parameters using gradient descent for a specific amount of epochs. For each iteration we will store the error based on the decimal prediction obtained using the sigmoid function so we can graph all the errors and see how our model improves over time. We also want to graph how well our model predicts if a passenger will survive or not, so each time a prediction is made, we will round it up to 0 or 1 and compare it with the real label, so that way each iteration we can see how accurate our model is and also see the improvement over time.

At the end we will save the final parameters, because those values are the ones that we will use to test our model and see how well it works with an unknown dataset.

**Testing the model**

Once we have the final parameters, we will use them to make one prediction for each sample of the “test.csv” file and upload the results to kaggle to compare the predictions with the real labels.



We got 77% accuracy which is not the best, but it is still a great problem to solve using logistic regression to get familiar with machine learning basics.

**Conclusions**

Logistic regression is probably the simplest model used to predict the class that a sample will have, but in some problems the simplicity of the algorithm limits its performance when facing more complex problems.

Some things that could be done to improve the model are (Note that improving the model does not necessarily mean a better accuracy):

* Regularization
* Cross validation

There will be a second version of the report covering these topics with their respective explanation and why they are useful.

The code that contains the implementation of the logistic regression is available at: <https://github.com/maci2233/machine_learning/tree/master/Logistic_regression/titanic>

**REFERENCES**

[**https://www.ultimatetitanic.com/the-sinking**](https://www.ultimatetitanic.com/the-sinking)

[**https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc**](https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc)

[**https://ml-cheatsheet.readthedocs.io/en/latest/logistic\_regression.html#cost-function**](https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html#cost-function)

[**https://machinelearningmastery.com/implement-logistic-regression-stochastic-gradient-descent-scratch-python/**](https://machinelearningmastery.com/implement-logistic-regression-stochastic-gradient-descent-scratch-python/)

**https://www.kaggle.com/c/titanic/data**