TITANIC REPORT

Matteo Chianale

May 2023

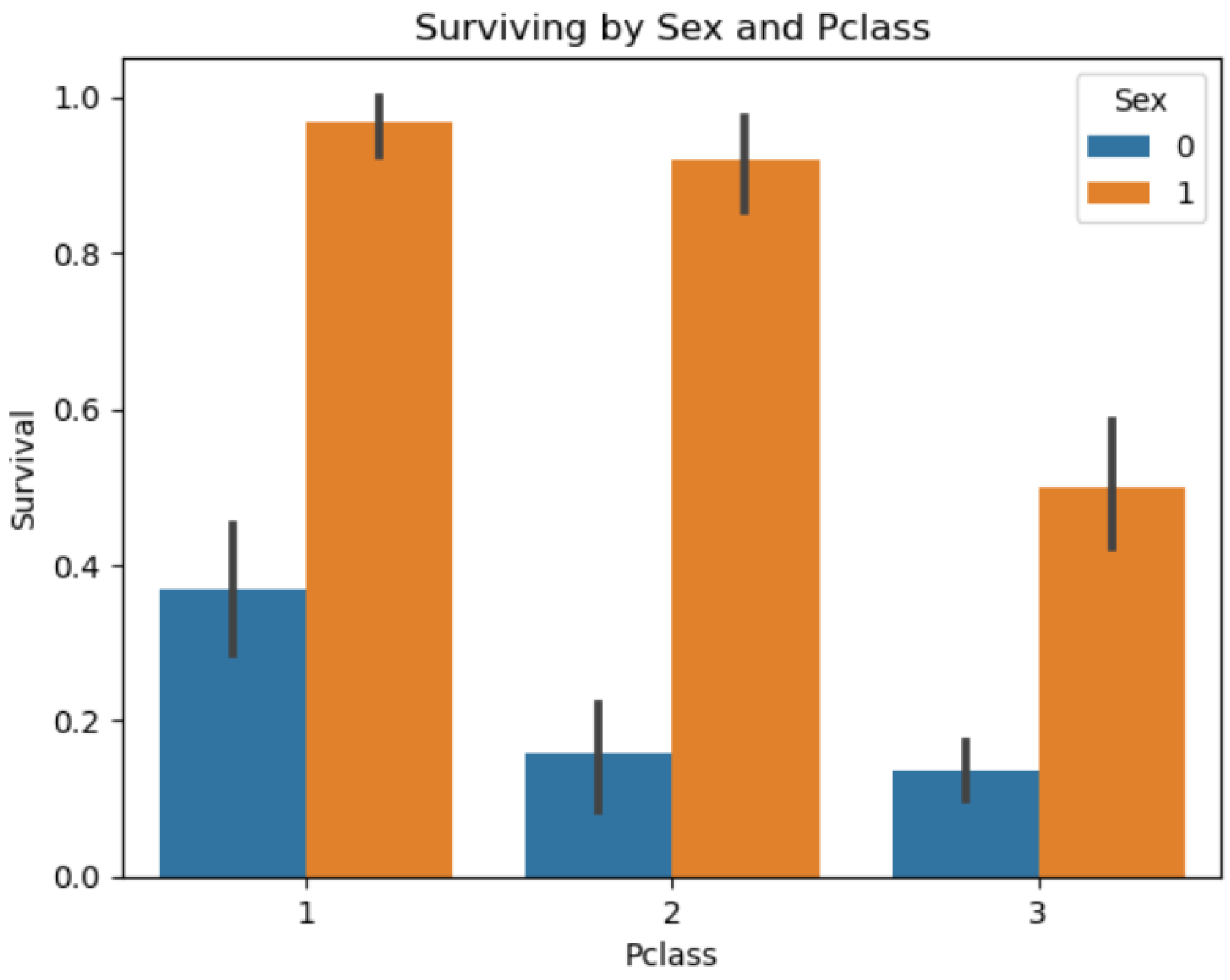
1 INTRODUCTION

GITHUB : `https://github.com/mchianale/titanic_survivor`

Titanic is a British transatlantic liner that sank in the North Atlantic Ocean in 1912 following a collision with an iceberg on its maiden voyage from Southampton to New York. Between 1,490 and 1,520 people died, making it one of the greatest maritime disasters in peacetime and the greatest for the time.he liner carried some of the wealthiest people of the time, as well as hundreds of emigrants from Britain and Ireland and elsewhere in Europe who were seeking new life in the United States.
The objective of the project is to predict whether or not a particular passenger has monitored, according to their status, their gender but also their age and to show how the belonging of an individual plays an important role in this kind of tragic event.

2 DATASET

In this project we already had train and test data. The dataset of the data to be tested was separated into two: "test.csv" which contained the features and "gender_submission.csv" which contained the value to predict. However, each dataset had to be cleaned. First, it was obvious that I can delete the data for the cabins, tickets and IDs of each passenger. I have already the class for each passenger, moreover cabin and ticket looked random and didn't seem to give us more information about each passenger class.
For the missing values of the Age, the Embarked and Fare column, I preferred to replace them with their respective medians. I made this choice to avoid losing data, especially for the age where there was a lot of value missing. Then we can finally get our first interpretation of the data.



This histogram easily shows that there is an important link between the survival of the passenger, his gender and his class. For example a female passenger belonging to the first class was much more likely to survive than a man of the 3rd class

3 LOGISTIC REGRESSION

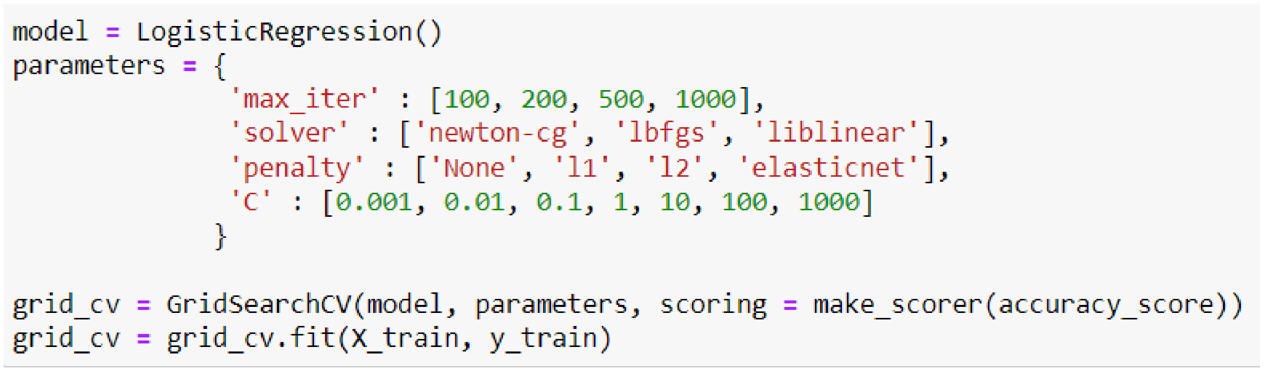
In this project, we use the Logistic Regression model which allows us to predict a binary value 1 and 0, here ”Survived”.
The purpose of this algorithm is to find a function h that predicts the y value (’Survived’) with the X parameters (the features). Specifically, we have:
y={1 if hX higher than threshold , 0 if hX lower than threshold} with default threshold usually being 0.5. The function used is thus a logistic function that

makes it possible to transform continuous values into a probability between 0 and 1.
For the first model created, I set the parameter max_iter to 1000. It determines the maximum number of iterations that the optimization algorithm must perform to adjust the model coefficients. By default, it is worth 100, I chose 1000 to have more precision but it increases the execution time.
The model obtained was made from X_train and X_test obtained in the previous parties with a text_size approximately equal to 0.46, which can be found by making the ratio of the number of lines of X_test by that of X_train.
Finally, X_test data were used to determine whether passengers survived or not. By comparing the values of y_true (belonging to ”gender_submission.csv”) we obtain an accuracy score of 93%. This means that the ratio of successful predictions to all predictions is 93%, which is really good.

4 HYPERPARAMETER

In this last step, we wanted to increase the accuracy score obtained previously by playing with the parameters of the Logistic Regression model, which can be found directly on the site of the sklearn library:
`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

Moreover, I used the GridSearchCV library to be able to test each combination of parameters that I set myself. Now we have this :



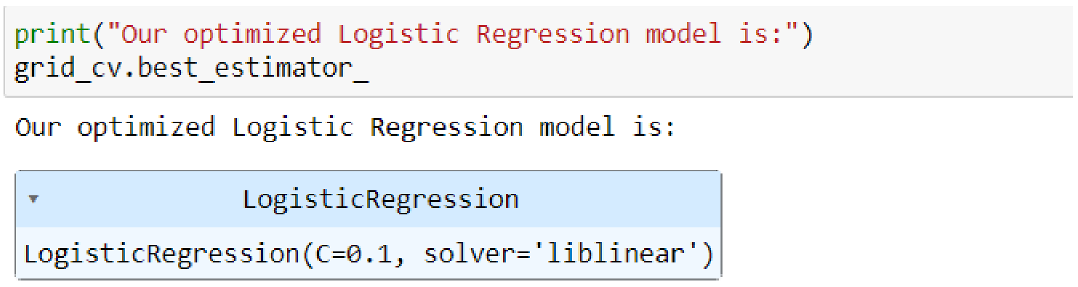
```
model = LogisticRegression()
parameters = {
                'max_iter' : [100, 200, 500, 1000],
                'solver' : ['newton-cg', 'lbfgs', 'liblinear'],
                'penalty' : ['None', 'l1', 'l2', 'elasticnet'],
                'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
               }

grid_cv = GridSearchCV(model, parameters, scoring = make_scorer(accuracy_score))
grid_cv = grid_cv.fit(X_train, y_train)
```

- Max_iter, as previously stated, the higher this parameter, the higher the execution time. So I do varied max_iter from 100 to 1000 to try to get the lowest value for which we have the best accuracy score.
- Solver is the algorithm to be used in the optimization problem. To make the choice we must consider the size of each sample and their ratios, but also the number of features.
- The penalty parameter is a hyperparameter for logistic regression that specifies

the penalty used for model regularization. Regularisation is a technique used to reduce overfitting (overlearning) by adding a constraint to the cost function.
- The parameter C is a numeric value, which when it is low will give a strong regularization, which can be useful to avoid overfitting, while a value of C high will give a low regularization, which can be useful to increase model accuracy on drive data.

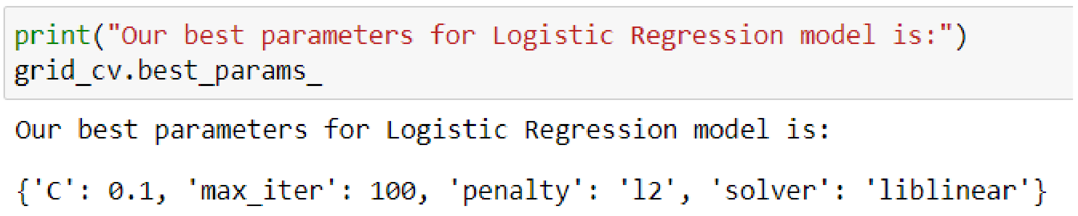
Finaly, we get this :



```
print("Our optimized Logistic Regression model is:")
grid_cv.best_estimator_
```

```
Our optimized Logistic Regression model is:
```

```
LogisticRegression
LogisticRegression(C=0.1, solver='liblinear')
```



```
print("Our best parameters for Logistic Regression model is:")
grid_cv.best_params_
```

```
Our best parameters for Logistic Regression model is:

{'C': 0.1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
```

- C = 0.1 is a very small value and allows us to specify stronger regularization
- We get max_iter = 100 which is very good because it allows us to get less time of execution.
- Penality L2 adds a penalty to the cost function of logistic regression, which is the sum of the squares of the characteristic coefficients multiplied by a regularization factor (alpha).
- We get as solver liblinear. This is logical because we have a medium size dataset with a fairly large number of features. Specifically, the liblinear solver implements a SGD variant called coordinate descent, which resolves the model coefficients one at a time, updating them to a component at a time of the cost function. This approach is very effective for binary classification problems (two classes).

In conclusion, by applying these parameters to our model, we obtain an accuracy score of 95% or a gain of 2% compared to before.

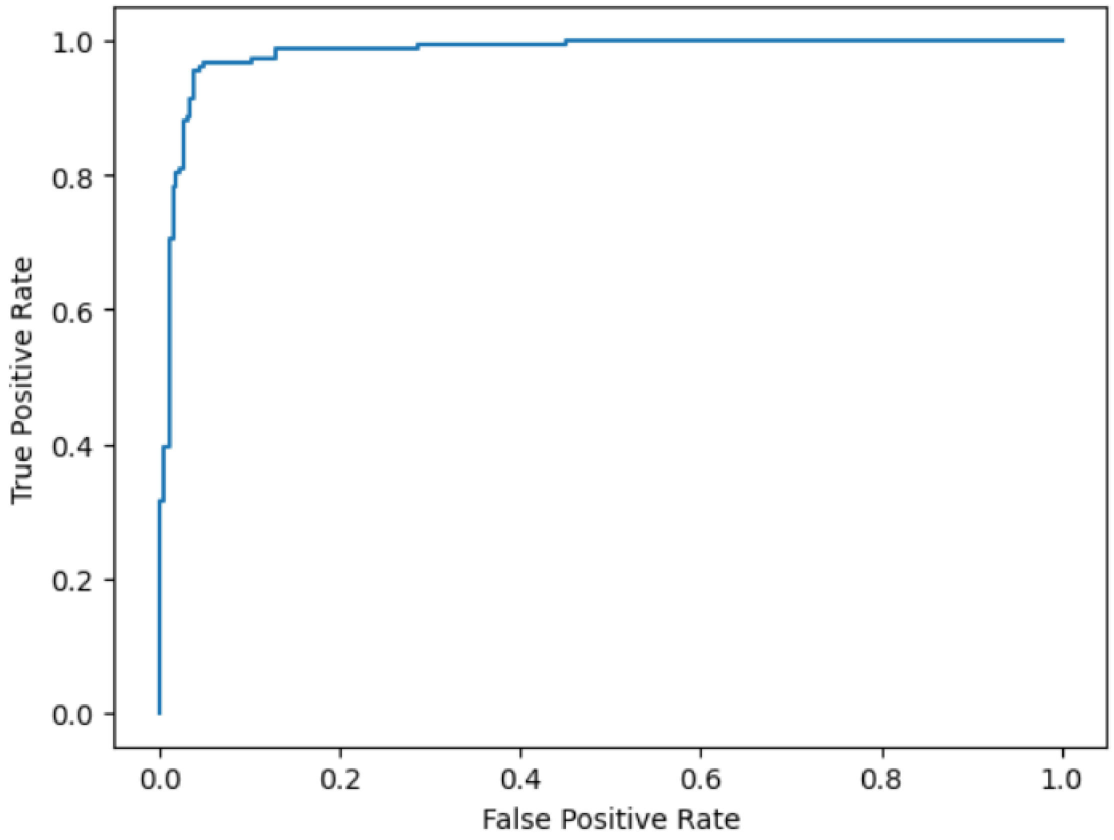
5 CONCLUSION

In this conclusion we analyse the result of my model:

First, we get the confusion matrix:

```
[[256  10]
 [ 10 142]]
```

We have just 10 False Positive and 10 False Negative. And we have a precision for positive equals to 93% :
p = TP/(TP+FP) = 142/(142+10) = 0.93. So our model is very efficient.
After, I plot a ROC Curve, which stands for "receiver operating characteristic" curve. This is a plot that displays the sensitivity and specificity of a logistic regression model.



The more the curve follows the top left corner of the graph, the better the model categorizes the data. As we can see from the above graph, this logistic regression model does a pretty good job of classifying the data into categories. To quantify this, we can calculate the AUC (area under the curve) which tells us which part of the graph is below the curve : auc = 0.98.The closer the AUC gets to 1, the better the model is and we are.

In conlusion our model fits well, this means that passenger characteristics really had an impact on their survival.