Assignment 2: A Mathematical Essay on Logistic Regression

Aman Kumar
(EE21B013)
Dept. of Electrical Engineering
Indian Institute of Technology Madras
Email: ee21b013@smail.iitm.ac.in

***Abstract*—The aim of this assignment is to examine the mathematical foundations of logistic regression and apply them to a real-world scenario. Logistic regression is used to identify the factors that could have predicted an individual's likelihood of survival during the historical sinking of the *Titanic*. Data visualization, cleaning, and modeling are conducted using Python. The analysis demonstrates that, while luck was a major factor in survival, other elements such as socioeconomic status and gender also significantly influenced the chances of survival.**

***Index Terms*—logistic regression, python, visualization, predictive modeling**

I. Introduction

The sinking of the *Titanic* is one of the most tragic incidents in human history. On April 15, 1912, during her maiden voyage, the widely regarded "unsinkable" RMS *Titanic* sank after colliding with an iceberg, resulting in the deaths of 1502 out of 2224 individuals (including passengers and crew). While luck undoubtedly played a role in determining who survived, it is important to investigate whether other factors, such as socioeconomic status, influenced survival outcomes. Identifying such factors may reveal underlying biases that could disadvantage certain groups in society.

In this assignment, the goal is to apply logistic regression to address the aforementioned problem. Logistic regression is a widely used mathematical model for estimating how a binary variable is influenced by multiple known factors. By conducting a logistic regression analysis, we can understand how each feature affects the target variable. This model can then be employed to predict the target variable in situations where its actual value is unknown.

In this specific context, the aim is to build a logistic regression model to predict whether an individual would survive the *Titanic* disaster, given certain characteristics of the individual, such as socioeconomic status and gender. Using data where the survival status is known, we can develop a model to predict survival for cases where it is unknown. The accuracy of the model can then be used to evaluate the strength of the identified relationships.

Section II provides an overview of the methods used for data cleaning, feature engineering, and initial exploratory analysis. Many insights can be gleaned from qualitative observations of the data. Section III presents a brief description of the mathematical formalism underlying logistic regression. Section IV discusses the results obtained from applying logistic regression in this case. Finally, Section V summarizes the key conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we outline the process of data cleaning, feature engineering, and data visualization.

A. Data Cleaning

We are provided with two datasets: a training dataset and a test dataset. The training dataset contains 891 rows and 11 columns, while the test dataset contains 418 rows and 10 columns. A brief overview of the training dataset is presented in Figure 1, and of the test dataset in Figure 2. In the training dataset, we have the ground truth values for whether each individual survived, whereas this information is not available in the test dataset. We will set the test dataset aside and use it only after we have completed building the model.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

Fig. 1. Summary of the training dataset

Our next task is to account for the missing values in the dataset. We see that there are missing values in three columns: `"Age"`, `"Cabin"`, and `"Embarked"`. Each column is dealt with separately, and the procedure is described below.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 418 entries, 892 to 1309
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Pclass    418 non-null    int64
 1   Name      418 non-null    object
 2   Sex       418 non-null    object
 3   Age       332 non-null    float64
 4   SibSp     418 non-null    int64
 5   Parch     418 non-null    int64
 6   Ticket    418 non-null    object
 7   Fare      417 non-null    float64
 8   Cabin     91 non-null     object
 9   Embarked  418 non-null    object
dtypes: float64(2), int64(3), object(5)
memory usage: 35.9+ KB
```

Fig. 2. Summary of the test dataset

The values in the column `Age` are depicted in a boxplot in Figure 3. The plot reveals that **the distribution is right-skewed**. In such cases, it is preferable to impute missing values using the **median rather than the mean**. Consequently, missing values in the `Age` column are replaced with the computed median, which is **28**.

For the `Cabin` column, we have only **202 non-null values out of a total of 891, which is just 22.67%**. Due to this low proportion, we **drop this column** from our predictive modeling. This decision is further supported by the fact that most data points in the test set also lack cabin values. However, we will use the available cabin values to explore any potential **correlation** with the chance of survival.

For the `Embarked` column, there are only **2 missing values**. Since this is a minimal number of data points, we **remove these rows**. After addressing the missing values, the dataset is summarized in Figure 4.

Additionally, the `Fare` feature has a missing value in the test set but not in the training set. We **impute this missing value using the mean** of the `Fare` feature in the test set.

B. Data Visualization/ Qualitative Analysis

To get a qualitative picture, we create plots of relevant features against the **survival rate**, which is defined as the number of individuals who survived divided by the total number of individuals.

First, we produce a **barplot** of passenger class, represented by the column `Pclass`. The resulting plot is shown in Figure 5. From the plot, it is evident that a **higher fraction of passengers** in higher passenger classes survived compared to those in lower passenger classes. This disparity may be due to the fact that higher class tickets were often held by individuals who were **economically and socially privileged**, thereby providing them with an advantage during such crises.

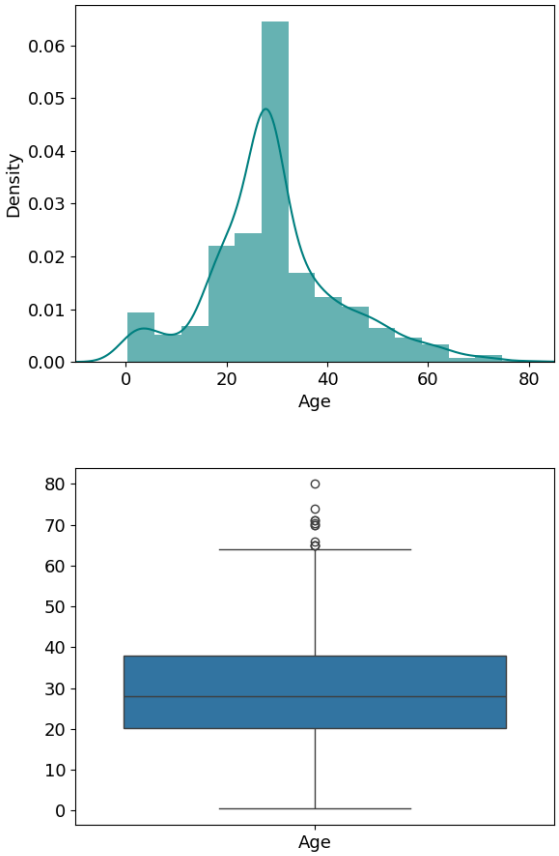

Fig. 3. Distribution of age

This observation also highlights an **inherent bias** present in society.

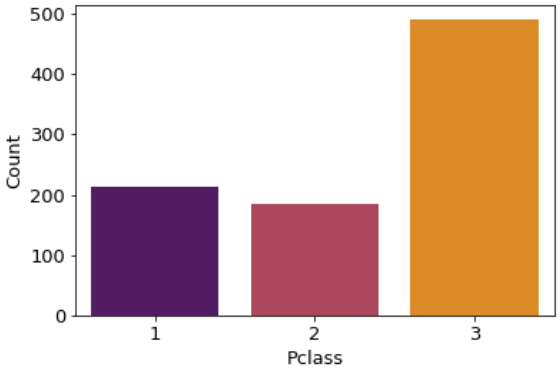

Fig. 4. Survival rate vs Passenger class

Next, we create a **barplot** of gender, represented by the column `Sex`. The resulting plot is shown in Figure 6. From the plot, it is evident that the **survival rate is higher for females** compared to males. This disparity could be attributed to the **social norms of the time**, which emphasized giving

preference to women and children during disasters.

Next, we create a **kdeplot** (kernel density estimate plot) of age, represented by the column `Age`. The resulting plot is shown in Figure 7. From the plot, we make two observations:

- There was a **greater chance for children to survive**. This could be because priority was given to women and children during evacuation.
- A **greater fraction of people in the age group 20-40 did not survive** compared to other age groups. This may be because individuals in this age group stayed behind to **aid younger and older individuals** during evacuation.

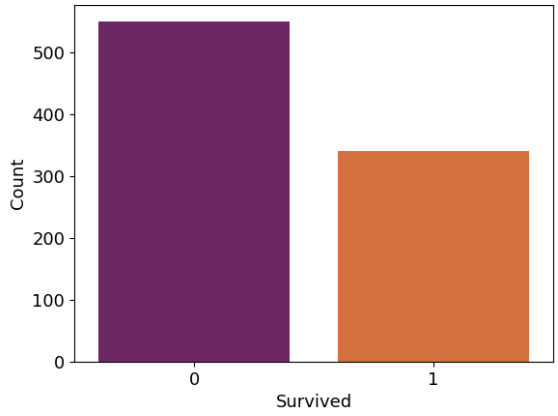

Fig. 5. Survival rate v/s Gender

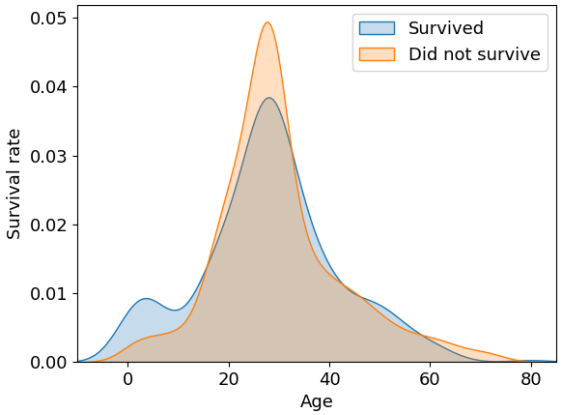

Fig. 6. Survival rate v/s Age

Next, we create a **kdeplot** (kernel density estimate plot) of fare, represented by the column `Fare`. The resulting plot is shown in Figure 8. It can be observed that, in general, individuals who paid a **higher fare** had a **higher chance of survival**. This association could be linked to **higher social and economic status** and the associated privilege, as generally only the wealthier sections of society can afford to pay a higher fare.

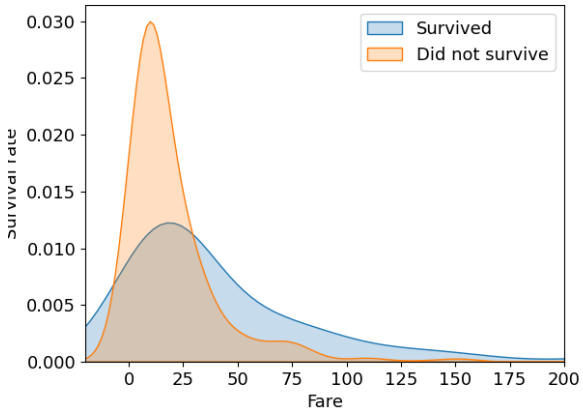

Fig. 7. Survival rate v/s Fare

Next, we explore if there is any relationship between `cabin` and `survival rate`. Since considering each cabin individually is impractical, we focus on the **cabin group**, which is represented by the letter preceding the number. Note that this analysis is limited to the data points with available cabin information; the rest are assigned the cabin group `U`. The resulting plot is shown in Figure 9.

Due to the **limited data available**, it is not possible to make conclusive statements regarding the impact of cabin group on survival rate. However, with more data, it might be feasible to **correlate cabin group with survival rate**, as we intuitively expect certain cabin groups to have **more access to lifeboats** and other similar factors.

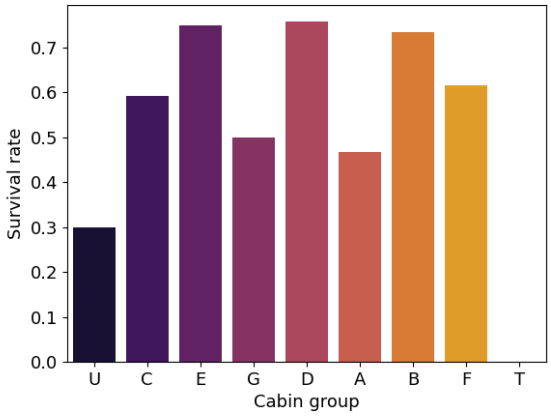

Fig. 8. Survival rate vs Cabin group

C. Feature Extraction

In this section, we focus on transforming some of the existing features to enhance their utility.

While we do not expect the name of an individual to directly affect their chance of survival, the honorifics associated with the name might have an impact. Therefore, we extract these

honorifics and store them in a new column titled `Title`. We anticipate that this feature will reflect the socioeconomic status of an individual.

The categorical features we now have are `Pclass`, `Sex`, `Embarked`, and `Title`. Of these, `Pclass`, which represents passenger class, is already numerical. As we observed earlier, a higher passenger class is associated with a higher chance of survival. Therefore, we retain the numerical ordering for this feature.

For the remaining categorical features, we use **one-hot encoding**. This approach avoids imposing an artificial order on categories that do not naturally have a ranking. Although one-hot encoding can sometimes increase the number of features significantly, it is manageable in this case due to the limited number of categorical features. After applying one-hot encoding, the dataset is summarized in Figure 10.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 1 to 891
Data columns (total 25 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  889 non-null    int64
 1   Pclass    889 non-null    int64
 2   Age       889 non-null    float64
 3   SibSp     889 non-null    int64
 4   Parch     889 non-null    int64
 5   Fare      889 non-null    float64
 6   female    889 non-null    int64
 7   C         889 non-null    int64
 8   Q         889 non-null    int64
 9   Capt      889 non-null    int64
 10  Col       889 non-null    int64
 11  Don       889 non-null    int64
 12  Dr        889 non-null    int64
 13  Jonkheer  889 non-null    int64
 14  Lady      889 non-null    int64
 15  Major     889 non-null    int64
 16  Master    889 non-null    int64
 17  Miss      889 non-null    int64
 18  Mlle      889 non-null    int64
 19  Mme       889 non-null    int64
 20  Mr        889 non-null    int64
 21  Mrs       889 non-null    int64
 22  Ms        889 non-null    int64
 23  Rev       889 non-null    int64
 24  Sir       889 non-null    int64
dtypes: float64(2), int64(23)
memory usage: 212.9 KB
```

Fig. 9. Summary of dataset after feature extraction

III. Model: Logistic Regression

In this section, we provide a brief overview of the mathematical formalism behind the logistic regression model.

Logistic Regression (also known as Logit Regression) is a model used to estimate the probability that an instance belongs to a particular class out of K possible classes. The most common case is when $K = 2$, resulting in binary classification. In this scenario, if the estimated probability is greater than 50%, the model predicts that the instance belongs to the positive class (labeled `1`). Otherwise, it predicts that the instance belongs to the negative class (labeled `0`).

Logistic Regression is a linear model for classification, which means the decision surfaces are linear functions of the input vector $\mathbf{x}$. The simplest case is to model $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ so that y is a real number. But for a classification problem, we seek to obtain probabilities that lie in the (0,1) range. Hence, we apply a non-linear function $f()$ such that

$$y(\mathbf{x}) = f(\mathbf{w}^\top \mathbf{x} + w_0)$$

returns the posterior probabilities. Decision surfaces correspond to $y(\mathbf{x}) = \text{constant}$, which implies $\mathbf{w}^\top \mathbf{x} + w_0 = \text{constant}$ for one-to-one f, which makes the decision boundary linear. An example of such a decision boundary is given in Figure 11.

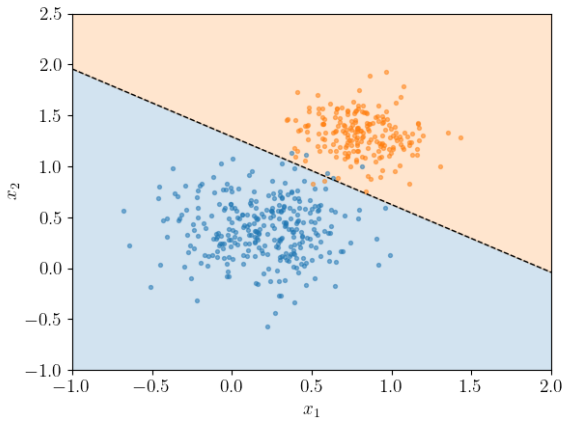

Fig. 10. Linear decision boundary of logistic regression

The non-linear function $f()$ used in logistic regression is a sigmoid function (also known as a logistic function) that outputs a number between 0 and 1:

$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

$$\hat{p} = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$

The estimated probabilities can easily be converted into a binary classifier by predicting

$$\hat{y} = \begin{cases} C_1, & \text{if } \hat{p} < \text{Threshold} \\ C_2, & \text{otherwise} \end{cases}$$

where C_1 and C_2 are the two classes. The threshold is usually set to 0.5.

A. Parameter Estimation

The parameters can be estimated using Maximum Likelihood Estimation (MLE). Let the number of data points be M and i be an integer such that $1 \leq i \leq M$; i is used to index a datapoint in the dataset. Let y_i denote the ground truth corresponding to datapoint $\mathbf{x}_i$. Then the Likelihood, given by $L(\mathbf{w})$ is:

$$L(\mathbf{w}) = P(y_1, y_2, \ldots, y_M | \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$$

$$= \prod_{i=1}^{M} P(Y_i = y_i | \mathbf{X}_i = \mathbf{x}_i, \mathbf{w})$$

$$= \prod_{i=1}^{M} \sigma(y_i \mathbf{w}^\top \mathbf{x}_i)$$

$$\log L(\mathbf{w}) = \sum_{i=1}^{M} \log \sigma(y_i \mathbf{w}^\top \mathbf{x}_i)$$

We could maximize log-likelihood or minimize negative log-likelihood to estimate the parameters. Or equivalently, minimize Empirical Logistic Loss function, $\hat{R}(\mathbf{w})$ defined as:

$$\hat{R}(\mathbf{w}) \equiv -\log L(\mathbf{w}) = \sum_{i=1}^{M} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i))$$

The goal is to minimize $\hat{R}(\mathbf{w})$; however, there exists no closed-form solution. So we make use of Gradient Descent. The steps involved in the gradient descent algorithm for logistic regression are briefly outlined below:

- Initialize $\mathbf{w}_t = \mathbf{w}_0$ randomly.
- Repeat till convergence:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \hat{R}(\mathbf{w}_t)$$

$$= \mathbf{w}_t - \eta \sum_{i=1}^{M} \sigma(-y_i \mathbf{w}_t^\top \mathbf{x}_i)(-y_i \mathbf{x}_i)$$

where η is the learning rate and t is the iteration number.

B. Regularized Logistic Regression

In order for the model to generalize well and to prevent over-fitting, a penalty function could be added to the loss function. We choose the L_2 norm, and hence this is termed Ridge regression. For the L_2 norm penalty function, the empirical loss function becomes:

$$\hat{R}(\mathbf{w}) = \sum_{i=1}^{M} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)) + \frac{\lambda}{2} ||\mathbf{w}||^2$$

where λ denotes the regularization parameter, which is treated as a hyper-parameter. The modified expression of $\hat{R}(\mathbf{w})$ is then used for performing gradient descent.

IV. Modeling

In this section, we discuss the application of the logistic regression model to our problem.

The logistic regression model is trained on the training set using gradient descent, with the training process abstracted through the `sklearn` library. The model has one hyperparameter, λ, which is the regularization parameter. The optimal value of λ is determined using cross-validation, a technique that involves splitting the training data into multiple subsets, evaluating different values of λ, and selecting the most suitable one.

To evaluate the predictive model for a classification problem, several metrics can be used. Below is a brief description of the most commonly used metrics:

- **Accuracy:** Accuracy is the ratio of the number of correct predictions to the total number of predictions. While intuitive, accuracy can be misleading in cases with skewed class distributions, as it may not reflect the model's performance on imbalanced datasets.
- **Precision:** Precision is the ratio of true positives to the total number of positive predictions. It is particularly important when the cost of false positives is high, such as in email spam detection.
- **Recall:** Recall is the ratio of true positives to the total number of positive ground truths. It is crucial when the cost of false negatives is high, such as in fraud detection or identifying sick patients.
- **F1-score:** The F1-score is the harmonic mean of precision and recall. It is useful when seeking a balance between precision and recall, especially in cases with uneven class distributions.

TABLE I
Evaluation of Logistic Regression Model

Metric	Score
Accuracy	0.831
Precision	0.791
Recall	0.759
F1 Score	0.775

The confusion matrix for the training set is shown in Figure 12, and the evaluation metrics are summarized in Table I. This analysis demonstrates that the model can reasonably predict an individual's survival chance based on the features utilized. It indicates a significant correlation between survival chances and factors such as socioeconomic status and gender. Predictions were also generated for the test set entries; however, the quality of these predictions could not be assessed due to the absence of labels.

The ROC curve is another commonly used tool for evaluating binary classifiers. A Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is generated by plotting the **true positive rate (TPR)** against the **false positive rate**

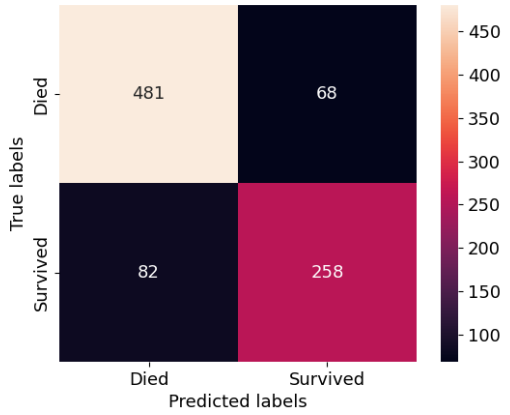

Fig. 11. Confusion matrix for the training set

(FPR) at various threshold settings. The ROC curve for the logistic regression model is shown in Figure 13.

The dotted line in the plot represents the ROC curve of a purely random classifier. A good classifier should be positioned as far away from this line as possible, ideally toward the top-left corner of the plot. One common way to compare classifiers is by measuring the **area under the curve (AUC)**. A perfect classifier has an ROC AUC of 1, while a purely random classifier has an ROC AUC of 0.5. The ROC AUC for our logistic regression classifier is **0.876**.

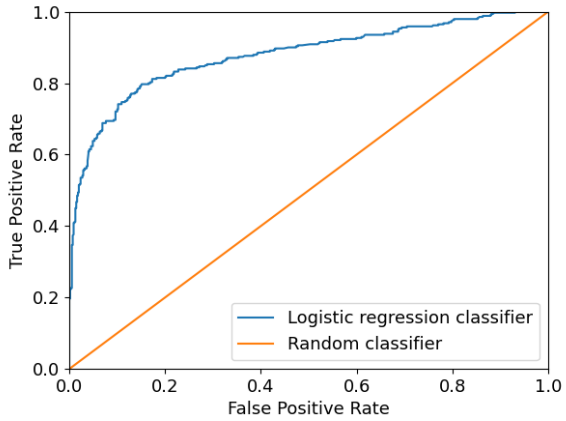

Fig. 12. ROC curve

V. Conclusions

From the logistic regression model, we were able to arrive at the conclusion that although luck did play a role in determining who survived and who did not, it also depended on a variety of other factors. In particular, it was found that passengers in first class, women, children, and people who paid a higher fare were all more likely to survive. We were also able to make predictions of whether an individual will survive or not for cases where the ground truth was unknown. This analysis helps bring out biases which society as a whole may voluntarily or involuntarily hold against sections of the population. Closely reflecting on such analyses and focusing on their implications can help better everyone's lives.

VI. Avenues for Further Research

Although logistic regression is a powerful model, it is only a linear model. Making use of non-linear models, either by considering non-linear models or by using kernels for introducing non-linearity, is an avenue worth exploring. Collecting more features (like proximity to lifeboats) could also help improve modeling.

References

1. A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc., 2nd ed., 2019.
2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
3. J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.
4. W. McKinney, "Data Structures for Statistical Computing in Python," in Proceedings of the 9th Python in Science Conference (S. van der Walt and J. Millman, eds.), pp. 56–61, 2010.
5. "Logistic regression." Available: https://en.wikipedia.org/wiki/Logistic_regression.