Binary Classification using Logistic Regression

Vaibhav Mahapatra
Data Science Department
Indian Institute of Technology Madras
Chennai, India
me19b197@smail.iitm.ac.in

***Abstract*—This article describes how Logistic Regression can be used to classify data into 2 classes via a model that learns from previous data. This document demonstrates how an empirical model can learn to predict or estimate labels based on past-accumulated data. We will first explore the theory behind this approach and then apply it on a real-world data set using Python and observe it's results.**

I. Introduction

Logistic Regression is a machine algorithm that comes under the subset of supervised learning. Logistic Regression aims at reliably predicting a discrete or binary output (0 or 1) based on various numerous and categorical inputs. It does so by learning from a set of previous data which is provided to the algorithm in input-output pairs. Its applications are multifarious. It can be used for descriptive analytics, i.e., to understand relationships between multiple variables. It can also be used for prescriptive analytics, i.e., using current data to predict and prescribe actions for the future. Some industry relevant applications include customer behaviour prediction, document classification, spam filtering, advertisement click-through rate prediction, etc.

Building Logistic Regression Models broadly involve 3 iterative steps: Data preparation, Model fitting and Model evaluation. After multiple iterations of the previous 3 steps and procuring satisfactory results, we select the most suitable model. Typically, for the first iteration, it takes around 70% time to clean and prepare the data, 20% time to fit models (preferably multiple), and 10% time to validate the results of the model. The theory for the first of Data Cleaning and Preparation is not explored in detail in this article as it is a combination of intuition and statistical inferences. We will comprehensively explore the next steps of Model fitting and Validation in this article.

Logistic Regression Models can be fit by minimising the *Log Loss* of the model. The Log Loss function is a convex function, which guarantees the existence of a minima for a linear input. The explanation of why Log Loss is used as a metric for Logistic Regression is explained in the next section. The cost function can be minimised in various ways, predominantly gradient based ones, namely *Gradient Descent*, *L-BFGS*, etc. The idea behind using gradient based approaches is discussed in detail in the next section.

The final section of this article will encapsulate a real-world use case of Logistic Regression. We will build a Classification model on the popular Titanic survival dataset, which can be found on Kaggle. The process will help us formally establish how various factors that contributed to the chances of survival of a passenger aboard the Titanic. Broadly, we'll examine the influence of age, gender and socio-economic status among other factors to predict the chances of survival. This article will contain various visual plots to help us intuitively process the various inter-relationships at play, along with the results of our approach.

II. Logistic Regression

A. Defining the Problem

We have n samples of d-dimensional input vectors $x_1, x_2, ...x_d$, where each component corresponds to an observed quantity or feature. For each corresponding sample, we have a 1-dimensional binary output variable represented by y. This y is the ground truth which can take only take the value of 0,1. Our objective is to determine a functional map $\hat{f}(x)$, such that the Log Loss is minimum. the function $\sigma(x)$ is the Sigmoid function which helps to squeeze outputs of the linear combination in the range (0,1). $\hat{f}(x)$ signifies the probability of the datapoint being classified as 1. When we make predictions, we'll set a threshold, typically 0.5, and classify the instance based on whether $\hat{y}_i$ is greater than or less than the threshold.

$$\hat{y} = \hat{f}(x) = \sigma(w_0 + w_1x_1 + w_2x_2 + ...w_dx_d) \tag{1}$$

$$\sigma(z) = \frac{1}{1 + exp(-z)} \tag{2}$$

$$LogLoss = \frac{-1}{n}\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1 - y_i)log(1 - \hat{y}_i) \tag{3}$$

Finding the best functional map is equivalent to finding the best weights vector $w = (w_0, w_2, ...w_d)$. One thing to note here is the form of the loss function. Unlike the Linear Regression case, even after setting $\nabla_W(Lossfunction)$ to zero, we won't get a closed form matrix representation.

B. Sigmoid function and Log Loss as a optimization metric

We'll first discuss the form of the Sigmoid function (2) and why it is a suitable choice for Logistic regression. By looking at it's formula we can make the following observations:

- $\lim_{z\to-\infty} \sigma(z) = 0$

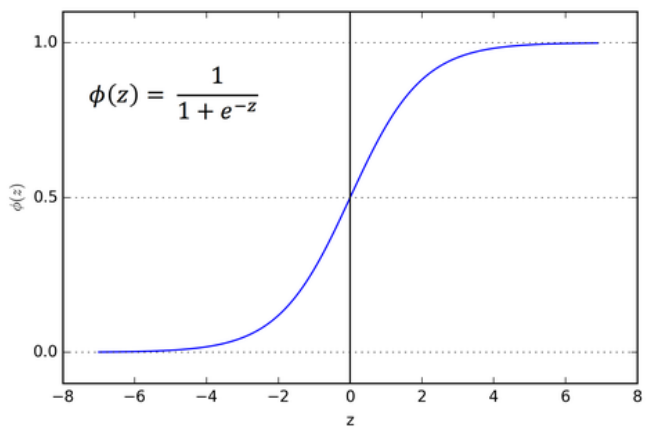


Fig. 1. Sigmoid Function

- $\lim_{z \to +\infty} \sigma(z) = 1$
- The sigmoid function squeezes all inputs into the range (0,1)
- The output of the sigmoid function can be considered as a probability map of the input

A visual representation of the same can be seen in Fig. 1

The reason for choosing the sigmoid function in the Logistic Regression model can be seen from the derivation of the Gaussian Naive Bayes Classifier. The main underlying assumption here is that the Conditional Class probabilities, $P(X|Y=1)$ and $P(X|Y=0)$, are Gaussian in nature. The derivation for the same can be seen in [3]. In slide 19, we can see the derived sigmoid function.

$$-\begin{cases} \log a_i, & y_i = 1, \\ \log(1-a_i), & y_i = 0. \end{cases}$$

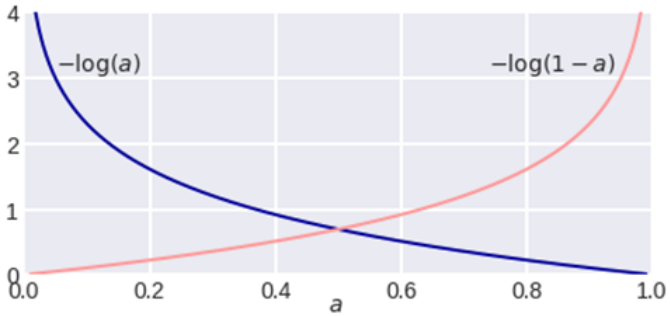


Fig. 2. Log Loss Function

We will now analyse the loss function that is optimised to build the Logistic Regression Model. The Log Loss function can be seen Fig. 2. From the graph, we can make some intuitive observations. For $y_i = 1$ observations, the loss function is equivalent to $-log(\hat{y}_i)$ (a_i in the figure is the same as $\hat{y}_i$). If we move in the descent direction of this graph, we progressively move towards a prediction of 1. We can make a similar observation for $y_i = 0$ observations.

C. Determining Weights by Minimising Log Loss

As discussed earlier, we cannot derived a close formed matrix formulation to find the optimum weights. Hence, we'll be discussing the use of Gradient Descent to optimise weights.

Let the Log Loss function be L(w). To find the $\nabla_w L$, we can apply chain rule. Before that, we can use a simple identity with regards to the derivative of the Sigmoid function.

$$\sigma'(z) = \frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z))$$

Now to derive $\nabla_w L$

$$\nabla_w L = \frac{-1}{n}\sum_{i=1}^{n}\left(\frac{y_i}{\hat{y}_i} - \frac{(1-y_i)}{(1-\hat{y}_i)}\right)\nabla_w \sigma(w^T x_i)$$

$$= \frac{-1}{n}\sum_{i=1}^{n}\left(\frac{y_i}{\hat{y}_i} - \frac{(1-y_i)}{(1-\hat{y}_i)}\right)\sigma(w^T x_i)(1-\sigma(w^T x_i))x_i$$

Based on the aove equation, we can formulate the weight updation rule for weights as follows:

$$W_{k+1} = W_k - \alpha \nabla_W L \tag{4}$$

In (4) α refers to a the step-size, a parameter chosen by the user, and k refers to the k^{th} iteration of the algorithm. As we move closer to the minima, the gradient reduces and the parameter updation gets slower. We stop running the algorithm based on a user-set tolerance for either the change in weights, change in L(w), etc. At the end of running the algorithm, we have an estimate of the weights which predicts scenarios well with less error.

D. Model Evaluation Metrics

Although Log Loss function helps us fit the model, it isn't a very intuitive metric to estimate the quality of our model's predictions. In this section, I will discuss the various methods of evaluating models with few intuitive metrics. We first split the training data into 2 sets: training data and validation data. In practice, a 70-30 or a 80-20 training-test split is fairly reasonable. The training set is used to train the model, whereas the validation data acts as the *unseen* data to evaluate the quality of a model. The metrics I have used have been mentioned below

- Accuracy: This metric is the most intuitive one. We evaluate the accuracy by calculating the proportion of correct predictions to total predictions made by the model. Although a high score of this metric may look like the model is performing well, it doesn't account for the number of errors made with regards to each class.
- F1 score: This metric accounts for errors made with respect to a specific class. Equation (5) shows how F1 encapsulates elements of precision as well as recall. In general, a high F1 score is desirable.

$$Precision = \frac{True\_Positives}{True\_Positives + False\_Positives}$$

$$Recall = \frac{True\_Positives}{True\_Positives + False\_Negatives}$$

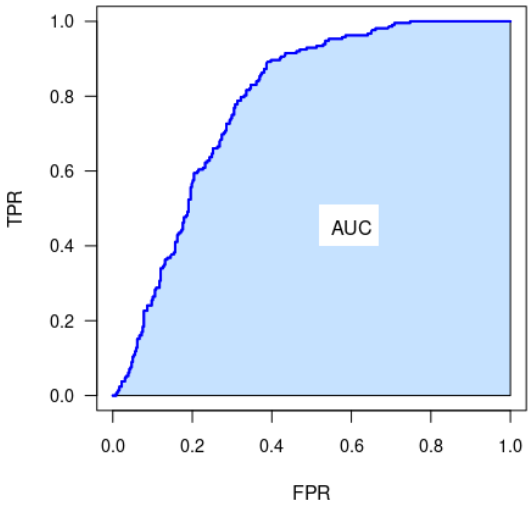


Fig. 3. ROC Curve and AUC Score

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

- ROC Curve and ROC-AUC-Score: An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters, True Positive Rate (TPR) and False Positive Rate (FPR). AUC Score (Area under curve) provides an aggregate measure of performance across all possible classification thresholds. An example of the same can be visualised in Fig. 3

$$TPR = \frac{TP}{TP + FN}$$

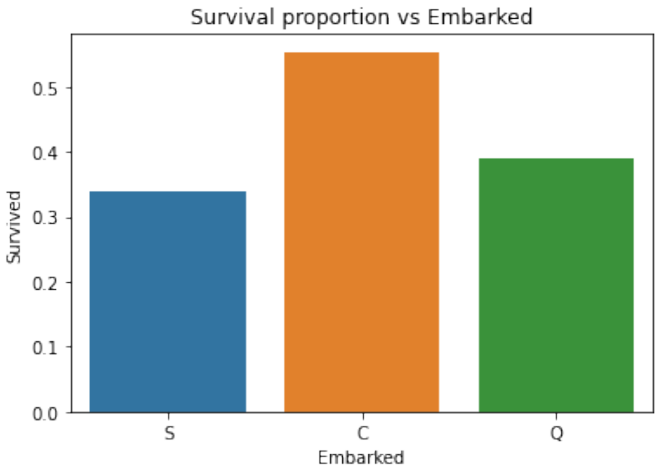
$$FPR = \frac{FP}{FP + TN}$$

There are mainly two advantages to using this metric, the first being that AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values. It also measures the quality of the model's predictions irrespective of what classification threshold is chosen.

III. Logistic Regression on Titanic Dataset

A. Dataset and Objectives

We will be working with the Titanic dataset with attributes like Name, Passenger Class, Age, No of family members, ticket fare, etc. Our objective is to perform descriptive analytics to understand how these factors correlate to survival rate of passengers. Then, use this information to predict if a new passenger with certain attributes will survive or not. Broadly, I have to clean data, generate relevant features and use Logistic Regression to build a classification model that predicts chances of survival.



B. Data Pre-processing and Visualization

Firstly, we will drop attributes that are either irrelevant or too sparse. Passenger_Id and Name are dropped as they are all unique values and intuitively don't play a role in chances of survival. Cabin is also dropped because it is too spare, with more than 75% empty values. Ticket column consists of alpha-numeric values which don't convey any new information in the presence of PClass and Embarked columns. As a result, we drop Ticket column also. Age has a fair number of null values ($\tilde{2}0\%$) but on plotting histograms, we see than certain age groups have higher survival rates than others. Hence, we retain Age in our analysis. We will now plot various graphs and draw some conclusions.

1) Pclass: There seemed to be more survivors in class 1, than class 2 and 3 by proportion.
2) Sex: Almost 75% female passengers survive in comparison to 20% male passengers.
3) SibSp: Individuals with less SibSp seemed to have a higher survival rate.
4) Parch: Similar to SibSp, individuals with lesser Parch have a higher chances of survival.
5) Embarked: Passengers who embarked from 'C' had more than 50% chance of survival, whereas those who embarked from 'S' and 'Q' had around 30-40% chance of survival as evident in Fig. III-A. There were 2 null values in this column. I substituted them with the mode of the column.
6) Fare: Majority of tickets belong to the fare bracket of 0-45. In this bracket significantly more people died than survived. From fares 45-350, there were relatively more survivors than deaths. Fares above 350 had 100% survival. Histograms visualising this can be seen in Fig. 5
7) Age: After plotting a histogram, it became fairly evident that certain age groups had a higher survival rate than others. Children under 15 survived more on average than adults in the band of 15-75 years. All elderly people

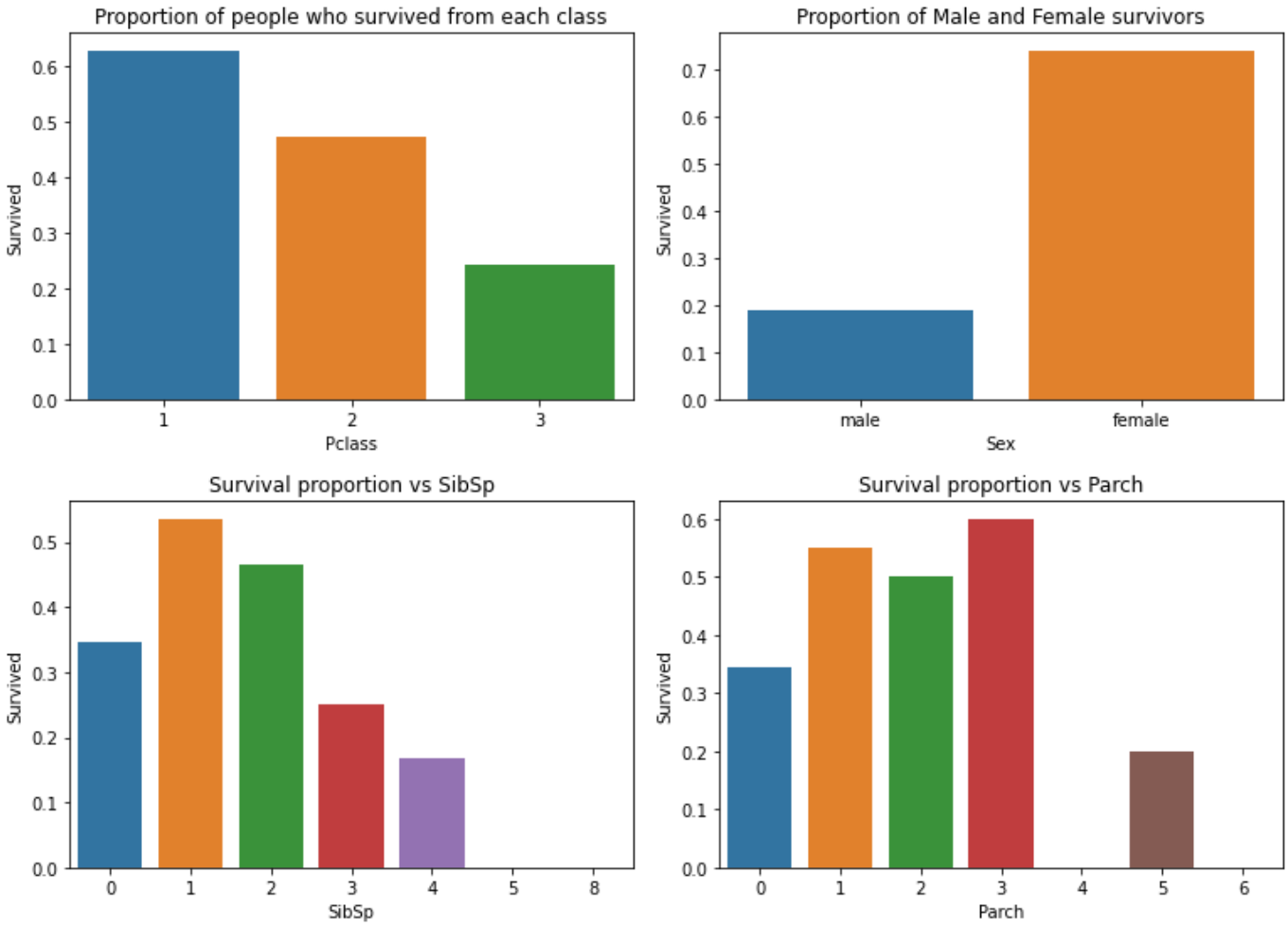


Fig. 4. Bar Graphs of Categorical Features against Survival Rate

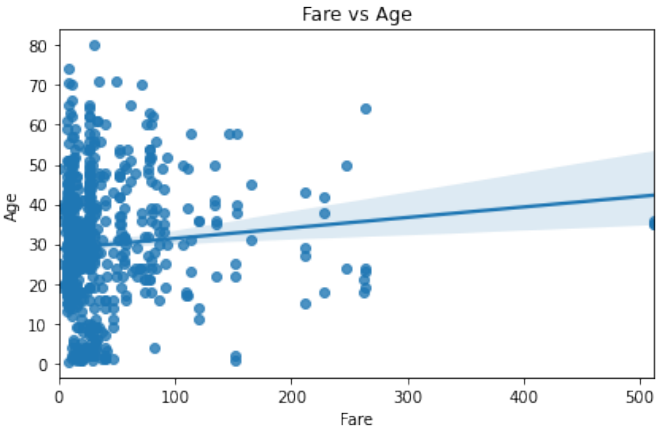
older than 75 survived. This column has a lot of Null values. The previous observations show that there is definitely a correlation between Age and Survival Rate. Hence, in the next section I will engineer this feature and fill the null values.

C. Feature Engineering the Age Column

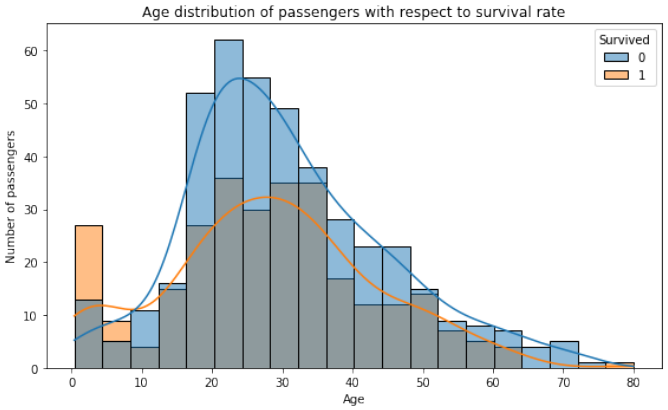
I chose to fit a Regression model to the data points which have Age values and use the model to predict the Age in null rows. I did some Exploratory Data Analysis on other columns with respect to Age, before fitting a Regression. This gave me the following Insights:

1) Fare: There seems to be some loose positive correlation between Fare and Age.
2) PClass, SibSp, Parch: These categorical features seemed to have some relationship with age. Some categories seemed to have people of a higher age than others.
3) Sex, Embarked: There didn't seem to be much correlation between Age and these categorical variables.

I took Fare, PClass, SibSp and Parch as the attributes to push into a Regression Model to generate Age. the Regressor



does not have a less RMSE (Root mean squared error) which implies that I have to carefully while using these Age estimates.



D. Model Fitting and Evaluation

I fit 2 Logistic Regression models, one without the Age column and another with the Age column filled with regressed values. The general Model fitting process I followed is as follows-

1) Choosing X & y- Select features from the dataset and store data in a matrix X. Store the outputs as a vector y
2) Polynomial transform- Generate polynomial features from X to capture non-linear relationships between X & y
3) Normalise data- Normalising all attributes of X the range of [0,1]. This increases the training speed of the Regression Model.
4) Train-test split- Split the X matrix and y vector into training and validation matrices. Preferably jumble the rows before doing so, to make the model more robust.
5) Train model- train a Logistic Regression Model on (X_train, y_train)
6) Evaluate model and reiterate- evaluate the model using various previously discussed metrics. Redo steps with different features and polynomial degrees till we get the best performing model.

After fitting 2 models, the better performing model was the one that includes Age. With this under consideration I fit one last model with *all* the data, including the Age column. For the final model, I also *oversampled* the minority class as the survival-deaths ratio was imbalanced. I used the *SMOTE(Synthetic Minority Over-sampling Technique)* to do so. Finally, the performance of my model can be evaluated as below.

TABLE I
LOGISTIC REGRESSION MODEL'S PERFORMANCE

Metric	Performance
Accuracy	81.42%
F1_Score	0.8101
ROC_AUC_Score	0.8142

Using this model, I made predictions on test data with unknown outputs.

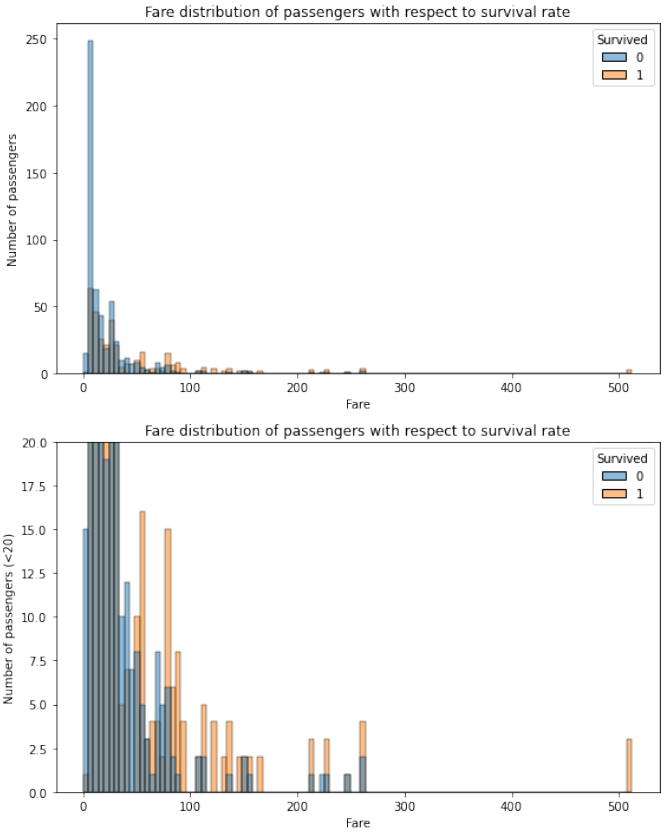


Fig. 5. Analysing influence of Fare on Survival Rate

IV. CONCLUSION

Logistic Regression has enabled us to establish some relationships between the inputs and survival rate. The data analysis we've done plus the models we built have captured some underlying correlations between data but are still are only performing nominally with an 80% accuracy. To improve the prediction capacity, we can build more complex models on data which capture non-linear relationships between data. Examples of such models are Tree-based models (Decision Tree, Random Forest), Artificial Neural Networks, etc.

REFERENCES

[1] Peng, Joanne & Lee, Kuk & Ingersoll, Gary. (2002). An Introduction to Logistic Regression Analysis and Reporting. Journal of Educational Research - J EDUC RES. 96. 3-14. 10.1080/00220670209598786.
[2] https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11
[3] https://www.cs.toronto.edu/ urtasun/courses/CSC411/tutorial4.pdf
[4] https://web.stanford.edu/ jurafsky/slp3/5.pdf
[5] https://sites.google.com/site/harishguruprasad/teaching/prml-aug-2022?authuser=0