Model for Classification for Evaluating Cars

Diego Perdomo Salcedo

Abstract –

The main objective of this report is to demonstrate the analysis behind a model developed to classify the level of acceptance of a car dataset. Along the paper the model proposed and the reasons why will be discussed, with the addition of the limitations and possible improvements that could be made. The accuracy of the model has reached an average of 82.95% as it trains and tests each possible class separately.

1. INTRODUCTION

When a consumer goes to buy a car they have to consider many aspects to know if they've made the right decision. Not only the price is an important factor, but also the level of comfort and safety the car offers. In the automotive industry, understanding these consumer preferences and knowing how to classify them based on their level of acceptance is crucial for both manufacturers and marketers.

Usually evaluating a car's suitability for a buyer might require experts or at least someone with a lot of experience. However, with advancements in machine learning, it is possible to automate this evaluation process, providing potential buyers with recommendations that match their needs.

In this context, the development of a machine learning model that can classify cars based on their varying acceptance level, whether they are unacceptable, acceptable, good or very good, becomes a valuable tool. Considering multiple attributes such as buying price, maintenance cost, number of doors, passenger capacity, luggage space, and safety .

AI and machine learning applications are empowering everyday customers to make informed decisions. This enables users to not only assess a vehicle's overall acceptance, but also to gain insights into the factors that are more suitable to them.

The Car Evaluation Dataset, which serves as the foundation for this project, was derived from a hierarchical decision model originally developed to demonstrate decision-making processes in expert systems. It was created by Marko Bohanec and Blaz Zupan. It consists of 1,738 instances, each covering the six attributes mentioned before. The dataset directly relates these attributes to the car's acceptability.

2. DATA CLEANING

This dataset contains 6 features, and all of them are strings. Since it evaluates each feature in an orderly manner such as: low, medium, high, very high; the first thing that had to be done was convert these strings into a numeric value. So changing the values to 0, 1, 2, 3, etc. will help so they can later be processed and used to assess the model. Also since instances in the dataset are in order a shuffle had to be used so the model could train and test more effectively.

3. DATASET SPLIT

Dividing the dataset into two lets the model train first and then test the results. For this to work adequatly the best general choice is to divide it into 80% and 20%. The reasoning behind this is to be able to train the model with the most amount of available data and evaluate the model as using unseen information to simulate how it will perform in a real-world scenario.

4. BASE MODEL

The project employs a logistic regression model built from scratch, without using existing machine learning libraries. This model is used primarily for binary classification but it can also be used for multiclass classification. In view of the amount of classes that need to be predicted, a cycle needs to be done for each class, in this case it's 4. In each cycle the model approaches it with a strategy called One-vs-Rest.

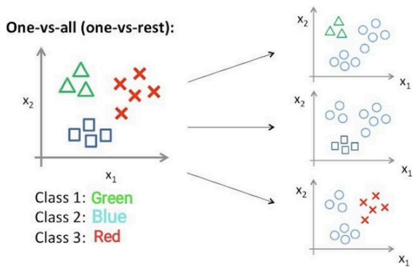


Figure 1: One-vs-Rest demonstration

What happens is that instead of trying to predict all four at once, the model focuses on one class and treats all the other classes as a single group. Then after applying the hypothesis function to each class it is able to determine the chances of it belonging to that class. After doing that for the next 3 classes it finds the index of the class with the highest probability, which corresponds to the predicted class.

4a.-Hypothesis

The hypothesis function is used to predict the probability that a given input sample belongs to a specific class. The sigmoid function which is:

$$h\theta(x) = \frac{1}{1+e^{-\theta Tx1}}$$

Its domain is the set of all real numbers, and its range is (0,1). So even if it receives a very large positive or negative number the output will always be between 0 and 1. This is why the One vs Rest Method is being used since it can only determine between 0 and 1.

4b.-Cost

The cost function quantifies the difference between the predicted probabilities and the actual class labes. Minimizing this during training allows the model to adjust its parameters to improve the accuracy. Since there are four classes the cost function is used for each class separately.

$$J(\theta) - \frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h\theta(x^{(i)})$$

4c.-Gradient Descent

The gradient descent enables the model to learn from data by minimizing the cost function. The first step is to calculate the gradient of the cost function with respect to the parameters. Then the parameters are updated in the direction that reduces the cost, thus increasing the accuracy of the predictions.

4d.-Training

The model learns by adjusting the parameters through the gradient descent, minimizing the eror in predictions across epochs. The One-vs-Rest strategy is used to train the model separately for each class, and then the cost is tracked to also monitor the progress. After training, the model is testted on unseen data to evaluate its generalization ability. The final test accuracy gives an indication of how well the model is likely to perform in a real-world scenario.

5. BASE MODEL RESULTS

After training the model and testing it, the results show that the model achieved an average accuracy of 85.26% on the test set.

Figure 2: Accuracy and cost over epochs

The model's performance was evaluated across the four distinct classes. Even though the model might seem good enough there are very interesting things to notice.

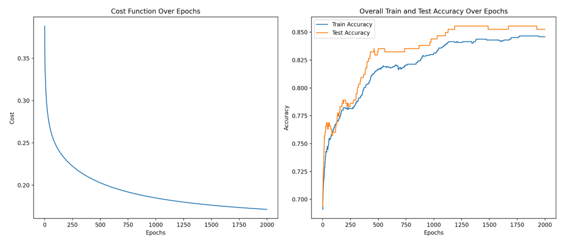


Figure 2: Accuracy and cost over epoch

These are the results of the accuracy for each individual class:

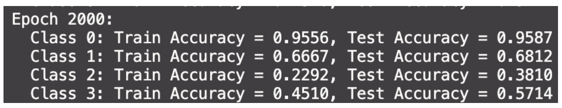

```
Epoch 2000:
  Class 0: Train Accuracy = 0.9556, Test Accuracy = 0.9587
  Class 1: Train Accuracy = 0.6667, Test Accuracy = 0.6812
  Class 2: Train Accuracy = 0.2292, Test Accuracy = 0.3810
  Class 3: Train Accuracy = 0.4510, Test Accuracy = 0.5714
```

Figure 3: Accuracy for all classes

The test accuracy is amazing for Class 0. This can be appreciated more in the following graph.

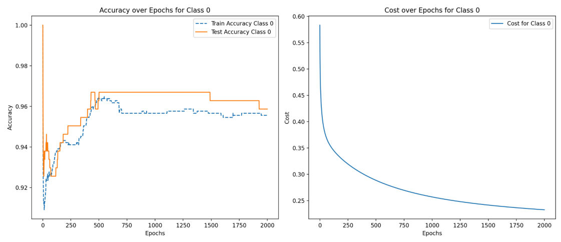


Figure 4: Accuracy and cost for Class 0

The model fits very well with Class 0 as it indicates a good balance between bias and variance and having the highest accuracy of all classes.

The fascinating information is what comes next. Following the success of Class 0 things start going downhill as the accuracy starts decreasing, the bias increasing and the variance as well. For Class 1 the variance is still relatively low as both the training and testing accuracy are close to each other. But it exhibits a moderate bias as the test accuracy reaches 68.12%. This also is where signs of the model underfitting starts to show as the accuracy starts going down.

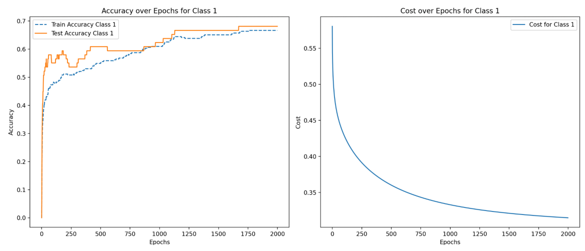


Figure 5: Accuracy and cost for Class 1

For Class 2 the model starts showing a high bias, underfitting and a high variance as the accuracy goes even lower and the difference between the training and test accuracy demonstrates a difference close to 20%.

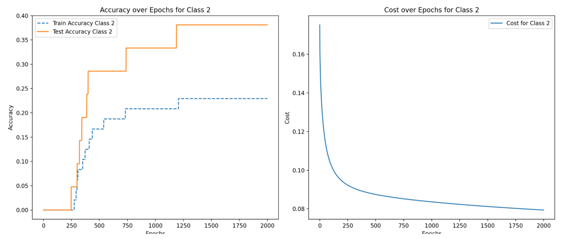


Figure 6: Accuracy and cost for Class 2

Last but not least is Class 3. Even though it's not as evident as Class 2 it's still visible that there is a high level of bias and variance. The train and test accuracy are low and far from each other.

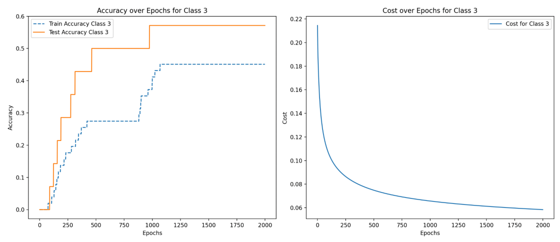
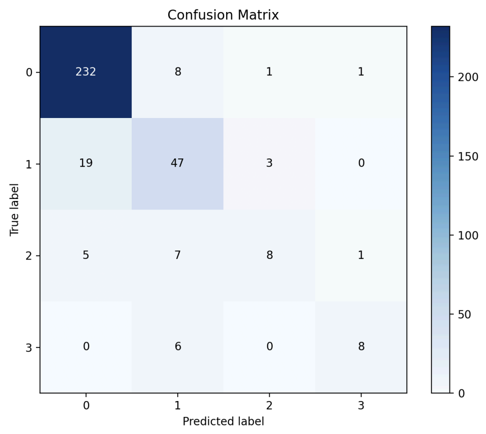


Figure 7: Accuracy and cost for Class 3

The model is having trouble with predicting classes other than Class 0. The following Confusion Matrix displays this as it compares the actual target values with the model's predicted values.



	0	1	2	3
0	232	8	1	1
1	19	47	3	0
2	5	7	8	1
3	0	6	0	8

Figure 8: Confusion Matrix

As a whole even though the model fits well for Class 0, indicated by the high amount of correct classifications. For the other classes the model appears to be underfitting since there is a high number of misclassifications. This can be seen as well in Figure 2 since overall the model is underfit with an average accuray of 85.26%. Also even if the amount of epochs increases, the cost stays around the same without improving. This is happening for a few reasons:

- Since the dataset has a significant imbalance between the number of instances in each class the model is more biased towards the more frequent class. In this case the most common is Class 0. (1210/1728)
- The model is relatively simple so it might not be good enough at capturing the complex relationships between features in multiclass classification problems like this one.

The main reason why the final result of the accuracy test is 85% even if the last 3 classes are much lower than that is because the dataset is mainly from the Class 0 which has a 95% accuracy.

6. FUTURE IMPROVEMENTS

There are various things that can be done to improve the results

6a.-Increase Model Complexity

The logistic regression model may be too simple. Since it's a multiclass problem, experimenting with a neural network with multiple layers and neurons might offer the model a more capacity to learn.

6b.-Increase the Number of Features

Including more features can help the model better differentiate between classes. This can be done by multiplying to past features together, squaring them, etc.

6c.-SMOTE

It's a powerful technique for learning from imbalanced data. It helps to balance the class distribution of the original dataset by generating synthetic samples of the minority class.

7. IMPROVED MODEL

Through some extra testing new flaws were detected. One of them being consistency when shuffling the instances in more orders. The last model was tested on seed #40.

Original model accuracy	
Seed Number	Accuarcy
40	85.26%

This gave a rather optimistic accuracy when compared to testing it with other seeds. For example, when its changed to #20 the results are very different.

Original model accuracy	
Seed Number	Accuarcy
20	82.08%

Figure 10: Original model accuracy (seed #20)

Therefore to improve the consistency of the model there were some techniques applied to improve it. The first thing done was cross-validation.

7a.-CROSS VALIDATION

In cross-validation the dataset is split into multiple subsets (folds). The model is trained and evaluated multiple times, each time using a different fold as the validation set and the remaining folds as the training set. By averaging the performance across these folds, cross-validation provides a more stable and reliable estimate of the model's performance.

8. IMPROVED MODEL RESULTS

After implementing this technique the model was able to stay at a consistent accuracy level despite how it was shuffled. Previously on seed #40 it had an accuracy of 85.26%. Now it has an accuracy of 84.72%.

Improved model accuracy	
Seed Number	Accuarcy
40	84.72%

Even though it's lower than the original, overall it's an improvement that demonstrates the truth of the model and how it will work in the real world. Seed #20 was at 82.08% accuracy, and now it's at 84.61%.

Improved model accuracy	
Seed Number	Accuarcy
20	84.61%

This can be further proved by displaying another set of results with a different seed. In this one the original model has an accuracy of 82.08% and the improved version ends with an accuracy of 84.61%. This shows that the model is in fact more consistent. Also added the two previous seeds to be able to visualize the difference between models accuracy easier:

Original vs Improved model accuracy		
Version	Seed Number	Accuracy
Original	10	82.08%
Improved	10	84.61%
Original	20	82.08%
Improved	20	84.61%
Original	40	85.26%
Improved	40	84.72%

Also the bias, variance and the fit stayed around the same across all classes. It still demonstrates a good fit with a low bias and variance for Class 0 as it has a high accuracy in both the train and validation.

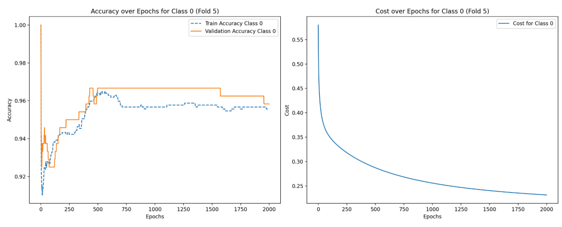


Figure 9: Accuracy and cost for Class 0 (Improved)

Underfitting starts to become more prevalent with the next three classes. In Class 1 the model is consistent when it comes to having a low variance, but the high bias illustrated by the low accuracy indicates that model is underfit.

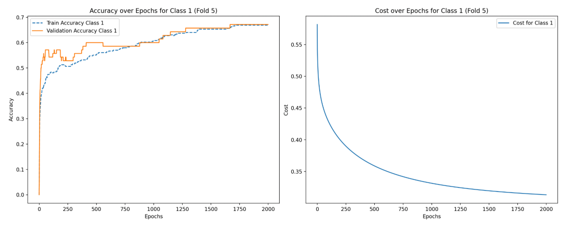


Figure 10: Accuracy and cost for Class 1 (Improved)

Although a higher bias, variance and underfitting can still be seen in Class 2, it achieved a lower amount of variance compared to the original model. As the difference between the train and test/validation accuracy decreased by around 5%.

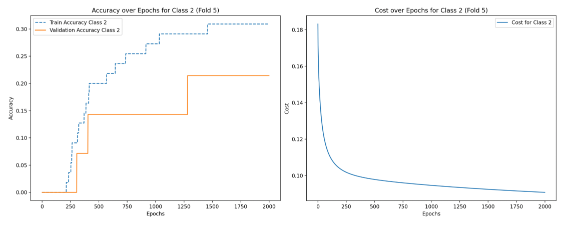


Figure 11: Accuracy and cost for Class 2 (Improved)

For Class 3 it still portrays a high amount of bias and variance just like the original model did. Its underfitting as both accuracies are relatively far from each other and low from what it should be.

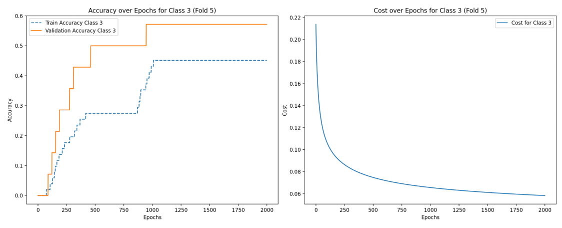


Figure 12: Accuracy and cost for Class 3 (Improved)

Although the model is still exhibiting the same problems as before (those being high bias, variance and underfitting on classes other than 0).

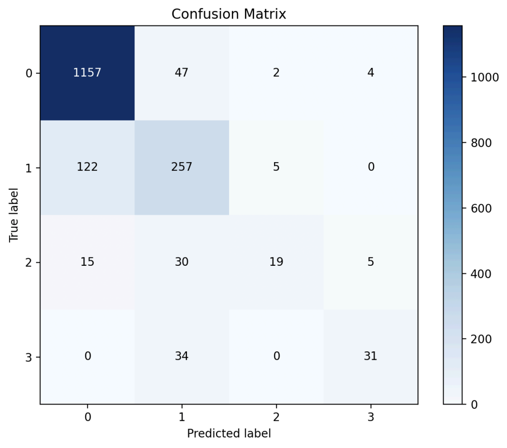


Figure 12: Confusion Matrix (Improved)

It has improved when it comes to being a more consistent model, being capable of correctly classifying around 84% of the instances. The next model will address this problem adequately by applying new techniques.

9. MODEL WITH FRAMEWORK

The model with framework has numerous advantages compared to coding it manually. It provides more speed, accuracy and a wide range of built-in features that help reduce the time spent on implementation while improving the overall quality of the model. One of those used was XGBoost.

9a.-Boosting

Boosting methods work by building a family of models that are aggregated to obtain a strong learner thet performs better. It fits multiple weak learners in a sequence, giving more importance to observations in the dataset that were badly handled by the previous models in the sequence. Boosting mainly focuses on reducing bias which is one of the problems encountered in the manually developed model.

9b.-XGBoost

There are two important boosting algorithms, adaboost and gradient boosting. XGBoost is a machine learning library that implements a highly efficient, scalable, and flexible gradient boosting framework. Gradient boosting is a machine learning technique that builds models in a sequential manner, where each new model attempts to correct the errors made by the previous models. It also enhances this technique by incorporating regularization methods, parallel processing and other optimizations that make it one of the most popular and powerful tools for structure data.

9c.-Implementation

Currently the model is using a variety of frameworks to be able to efficiently build a model capable of predicting the multiple classes in the dataset.

Scikit-learn is a comprehensive library for machine learning. It is used in multiple ways. First of all it is used to split the dataset into training and testing subsets. Also a standard scaler to help ensure that all the features are able to contribute equally to the model. After all of this comes the key method.

To use XGBoost it has to receive specified parameters that are previously defined. These parameters define the method being used which in this case is 'multi:softmax' to do multiclass classification. It also utilizes **regularization** to prevent overfitting. L1 or reg-alpha will make the model more likely to ignore some features (by setting their coefficients to zero), effectively performing feature selection. Also L2 or reg_lambda is defined to make the model more resistant to overfitting by shrinking the coefficients of less important features, leading to a more stable model. With 'fit' trains the model on the training data while evaluating it on both the training and test set to track its performance.

10. MODEL WITH FRAMEWORK RESULTS

After implementing the model using frameworks the accuracy incremented a lot more than expected. On the improved version of the manual model it reached an average of about 84%. Due to the usage of XGBoost, which is a framework that helps the exact problem faced in the previous models, the accuracy skyrocketed. It reached a maximum of 100% accuracy on training and 99.71% on testing.

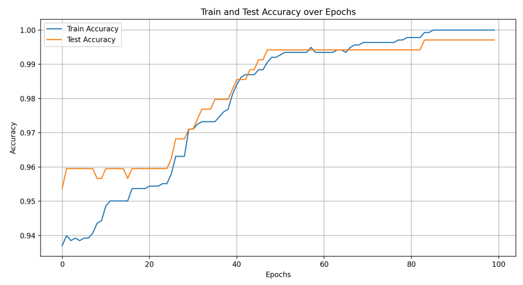


Figure 13: Accuracy over Epochs (Framework)

The model is now able to successfully learn from its mistakes done by the previous model, helping it predict the classes it had trouble predicting before. It has a low variance and low bias indicating a good fit as the train and test accuracy are both high and close to each other.

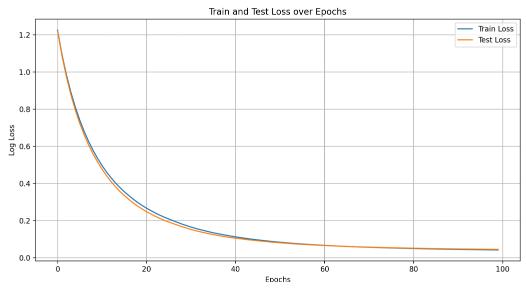


Figure 14: Train and Test Loss over Epochs (Framework)

The previous graph shows that the model has been trained effectively. Both training and test losses decrease and stabalize, which means that it is learning correctly without under or overfitting. The model is performing well on both the training and unseen test data.

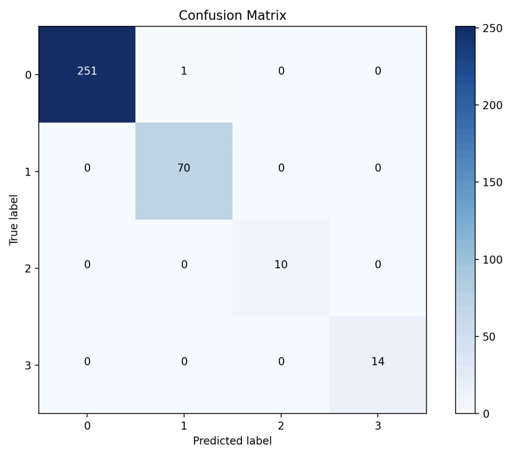


Figure 15: Confusion Matrix (Framework)

With this confusion matrix it is visible that out of all the predictions it was wrong only one time.

11. CONCLUSIONS

The development of a logistic regression model made to classify the acceptance level of cars based on various features provided insights into the strengths and limitations of the model.

In the original model It performed really well for the Class 0. This means it can effectively predict the most common class. However, the model struggles with the classes that have less instances on the dataset. Leading to a higher bias and variance, also exhibiting underfitting.

The improved model became more consistent with the implementation of cross-validation, overall being able to successfully classify correctly the classes with an accuracy of 84%. Although it still had the same problems as the model before (high bias/variance, underfitting on classes other than 0) it proved to be better at assessing new data which will help it in more real-world scenarios.

For the model with framework XGBoost and regularization was implemented successfully. Usually boosting causes overfitting, however it was avoided by utilizing regularization and the fact that the model was previously underfitting so with both it was able to reach a basically perfect fit with very low variance and bias as the training accuarrcy reached 100% and the testing reached 99.71%.

In conclusion, the best model between all of them is the one with the framework being to more successfully predict all of the classes without much trouble.

9. References

- "UCI Machine Learning Repository." *Archive.ics.uci.edu*, archive.ics.uci.edu/dataset/19/car+evaluation.
- "In Machine Learning, Do They See at Test or Train Accuracy? If Test Accuracy Is 100, Does That Mean Overfitting?" *Quora*, 2019, www.quora.com/In-machine-learning-do-they-see-at-test-or-train-accuracy-If-test-accuracy-is-100-does-that-mean-overfitting.
- "Regularization." *C3 AI*, c3.ai/introduction-what-is-machine-learning/regularization/#:~:text=Regularization%20is%20a%20method%20to.
- Guido Muscioni. "XGBoost - Get Probabilities after Multi:softmax Function." *Stack Overflow*, 2024, stackoverflow.com/questions/54895470/xgboost-get-probabilities-after-multisoftmax-function.
- https://www.facebook.com/jason.brownlee.39. (2014, February 18). *How To Choose The Right Test Options When Evaluating Machine Learning Algorithms*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/
- Markham, Kevin. "Simple Guide to Confusion Matrix Terminology." *Data School*, Data School, 26 Mar. 2014, www.dataschool.io/simple-guide-to-confusion-matrix-terminology/.
- Saeed, Mehreen. "A Gentle Introduction to Sigmoid Function." *Machine Learning Mastery*, 24 Aug. 2021, machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.
- Saxena, Shipra. "Binary Cross Entropy/Log Loss for Binary Classification." *Analytics Vidhya*, 3 Mar. 2021, www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/#:~:text=Binary%20Cross%20Entropy%20is%20used
- "The Complete Guide on Overfitting and Underfitting in Machine Learning." *Simplilearn.com*, www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting
- Um, Albert. "L1, L2 Regularization in XGBoost Regression." *Medium*, 12 Apr. 2021, albertum.medium.com/l1-l2-regularization-in-xgboost-regression-7b2db08a59e0.
- V., Ashwinkumar, et al. "One-Vs-Rest vs. Voting Classifiers for Multi-Label Text Classification: An Empirical Study." *E3S Web of Conferences*, vol. 491, 2024, p. 01014, www.e3s-conferences.org/articles/e3sconf/pdf/2024/21/e3sconf_icecs2024_01014.pdf, https://doi.org/10.1051/e3sconf/202449101014.
- "What Is the Best Way to Manage Overfitting and Underfitting in Statistical Validation for ML?" *Linkedin.com*, 2023, www.linkedin.com/advice/3/what-best-way-manage-overfitting-underfitting-ii1pf#:~:text=Cross%2Dvalidation%20can%20help%20reduce.
- "XGBoost Parameters — Xgboost 2.0.0-Dev Documentation." *Xgboost.readthedocs.io*, xgboost.readthedocs.io/en/latest/parameter.html#learning-task-parameters.
- Yennhi95zz. "#4. A Beginner's Guide to Gradient Descent in Machine Learning." *Medium*, Medium, 31 May 2023, medium.com/@yennhi95zz/4-a-beginners-guide-to-gradient-descent-in-machine-learning-773ba7cd3dfe#:~:text=III