Blind Spot Detection Using Extreme Gradient Boosting

Ben Noyes, Ryan Perry, Jake Miller, Daniel Phan

Computer Science Department, University of Tennessee, Knoxville
1015 Volunteer Boulevard, Knoxville, TN 37996, USA

bnoyes@vols.utk.edu

rperry21@vols.utk.edu

jmill291@vols.utk.edu

dphan5@vols.utk.edu

***Abstract*— Blind spots pose a significant threat to driver safety, especially when changing lanes. This project presents a shallow machine learning approach to blind spot detection. Leveraging the CIFAR-100 dataset, we develop two lightweight binary classifiers using a histogram of oriented gradient features extracted from grayscale images. The baseline support vector machine achieved high accuracy, but showed signs of overfitting. We improved generalization using an extreme gradient boosting model with regularization and early stopping. This approach offers high recall with minimal computational cost, making it suitable for embedded systems in modern vehicles.**

***Keywords*— Autonomous automobiles, Boosting, Machine learning, Support vector machines**

I. Introduction

A blind spot is an area along the sides of a vehicle that is not visible to a driver through their mirrors [1]. Blind spots become especially dangerous when switching lanes on a busy freeway [2]. Quick decision making as well as astute spatial awareness is needed to prevent accidents. This project addresses this critical safety issue and aims to eliminate the blind spot using a shallow machine learning model. Implementing a system that continuously monitors a vehicle's blind spot is crucial because it directly contributes to keeping people on the road safe [3]. Although drivers should always exercise caution on the road, having a second line of defense will significantly reduce the number of accidents [3]. Furthermore, blind spot detection is becoming more and more important as self-driving cars are becoming more common [4]. Collecting blind spot data sooner rather than later is essential for validating the accuracy of new safety features in modern automobiles.

Ultimately, the primary goal of this project is to develop a machine learning model that can efficiently predict whether a car or other vehicle is in an automobile's blind spot. This system should operate in real-time, offering quick alerts to drivers but more importantly correct alerts. It is crucial that the model does not inaccurately report to the driver that their blind spot is clear.

II. Dataset

Image recognition models are trained on tens of thousands of labeled images to learn to identify objects within new images. The CIFAR-100 [5] dataset lends itself to this purpose. The dataset is divided into 100 distinct classes, totaling 50,000 and 10,000 training and test samples, respectively. [5]. The t-distributed stochastic neighbor embedding (t-SNE) algorithm is used to visualize high-dimensional data while maintaining local structure [6]. We apply t-SNE to the raw pixel data for all CIFAR-100 classes to reduce the dataset to two dimensions, allowing us to visualize the relationships between different classes. In Figure 1, we see that the t-SNE visualization is scattered because raw pixel values cannot adequately capture the semantic structure of the images [7]. Furthermore, the many classes in the CIFAR-100 dataset contribute to this dispersion. Projecting 100 categories into a two-dimensional space leads to overlapping groupings.

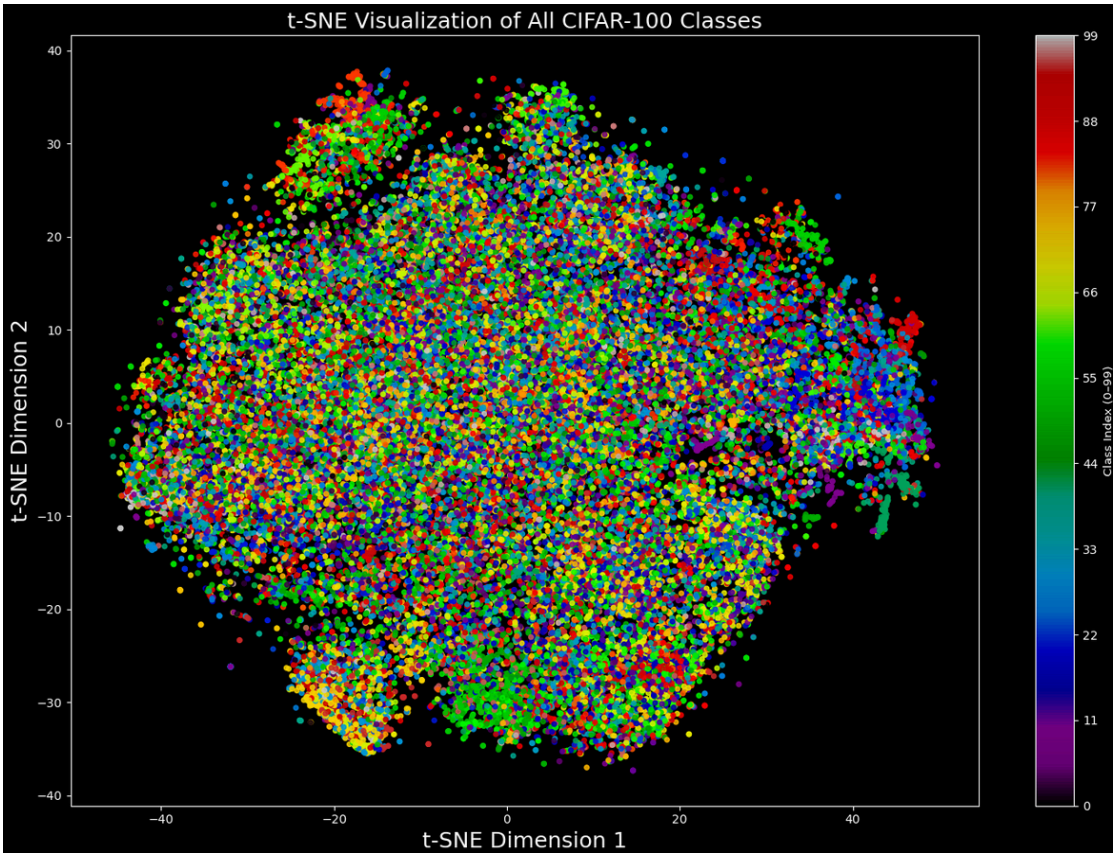


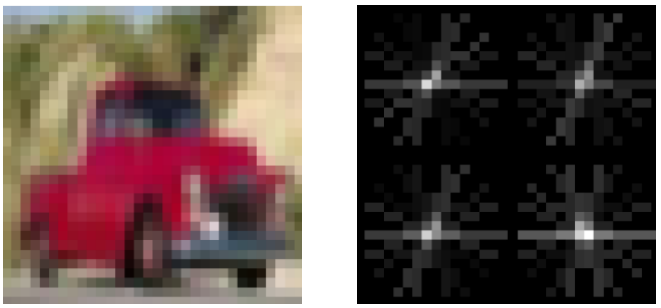
Fig. 1. t-SNE visualization of CIFAR-100 images based on raw pixel data. Each point represents a single image colored based on its class.

Given the size and format of CIFAR-100, the dataset will need to go through a series of preprocessing steps to prepare the raw data for analysis. It is important that we divide the classes into "car" and "non-car" groups, representing positive and negative predictions, respectively. Classes of objects a driver might see on the road include bicycles, buses, motorcycles, pickup trucks, streetcars, and tractors. The "car" group selected from among these classes contains 3,000 training samples. Thus, the "non-car" group contains 47,000 training samples. To balance the dataset, we select 3,000 random training samples from the "non-car" group.

Additionally, each image in the dataset is 32x32 pixels with 3 color channels (RGB) [5], resulting in a total of 3,072 features per image (1).

$$32\ pixels \times 32\ pixels \times 3\ color\ channels = 3,072\ features/image$$

However, image classification tasks do not require images directly. Instead, we use feature extraction techniques to obtain only the most relevant features. In this case, we use a Histogram of Oriented Gradients (HOG) to focus on the shape of an object in the image as opposed to its color [8]. In this scenario, color is irrelevant to our purpose. Thus, the images are processed in grayscale to reduce the cost. Figure 2 demonstrates how an image is visualized after applying HOG feature extraction and converting to grayscale:



a b

Fig. 2 . Original Image (a) and HOG Features Visualized (b)

III. Technical Approach

A. Baseline Model

Now that the data has been preprocessed, we can decide on our baseline solution. Popular object detection algorithms incorporate deep learning strategies, such as convolutional neural networks (CNN) [9]. For example, Kumar [10] leverages a three-layer CNN to make predictions about the make of a car in an image. Similarly, Asghar [11] builds a fully connected CNN to locate cars in individual frames of surveillance footage. However, many of these solutions require a high computational cost and are outside the scope of this project.

Instead, we shift our focus to shallow machine learning models that require less computational demand and are more interpretable to the user [12]. Shallow learning algorithms include linear regression, decision trees, random forest k-nearest neighbors, and support vector machines (SVMs). SVMs are useful for separating binary classes of data [10]. Because our predictions are either "car" or "non-car," an SVM is adequately suited for our purpose.

To build our model, we reference Suwal's [8] car detection SVM implementation. Suwal first imports positive and negative car images from a local dataset. Positive images contain cars, whereas negative images contain anything other than cars. In our case, we populate positive and negative directories with images imported from CIFAR-100. Next, we apply HOG feature extraction, as mentioned in Section II, to obtain a feature vector for all of the images in the positive and negative directories. With preprocessing done, we split our dataset into training and testing sets. Then, we fit the training dataset to a support vector classification (SVC) object with a regularization parameter of 1 before performing classification on samples in the test set. In Figure 4, the model correctly predicts that the image is not a car. Now that the SVC model is trained, we can evaluate its baseline performance.

Fig. 4 Model Prediction for Image

We use LibSVM to compute an accuracy score of 85.17% for the baseline model. We can improve upon our initial accuracy through hyperparameter tuning. We perform a grid search that tests different regularization parameters, kernel types, and gamma values to optimize accuracy. Table 1 summarizes the accuracy of our best SVC model using optimal hyperparameters. We see that the best test accuracy is equal to our baseline accuracy. This lack of improvement suggests that our model may be overfitting to the training data. To investigate this potential issue, we compare the training and validation accuracies. If the training accuracy is significantly higher than the validation accuracy, we confirm that our model is overfitting.

TABLE I
Best Hyperparameters for SVC

Hyperparamter	Regularization Parameter	Gamma	Kernel
Best Value	1.0	Scale	RBF
Cross-Validation Accuracy	84.19%		
Test Accuracy	85.17%		

Figure 5 plots a learning curve to illustrate how the model's accuracy changes as the size of the training set increases. We observe that the training and validation accuracies briefly converge toward each other before plateauing. While this implies that the model slightly improves as it learns new data,

the persistent gap between the lines suggests that our baseline model is overfitting.

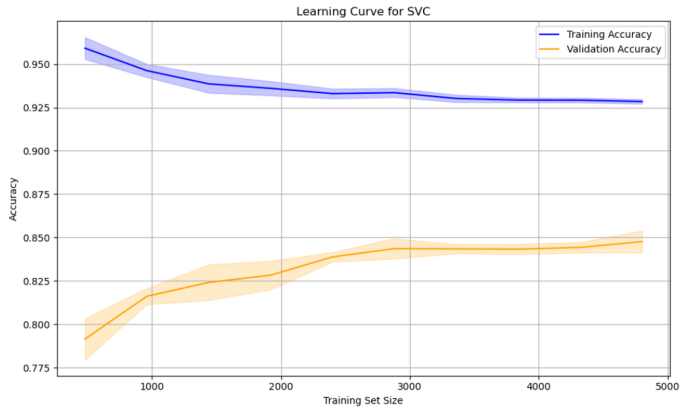


Fig. 5 Learning Curve for SVC

B. Model Improvements

We first address the problem of overfitting. SVMs are sensitive to noise [13] and thus are prone to overfitting on nonlinear kernels [14]. Therefore, we pivot to an Extreme Gradient Boosting (XGBoost) model which has been shown to resolve overfitting issues when used alongside HOG feature extraction [15]. We create an XGBoost classifier using logistic regression. We train the model using L1 ($\lambda = 5$) and L2 ($\alpha = 2$) regularization, which penalizes large weights to help prevent overfitting [16]. We also implement early stopping to halt training when the model no longer shows improvement on validation data [17]. The new model accuracy is 79.67%, a ~6% loss from the baseline accuracy. However, this reduction in performance is a tradeoff for better generalization. Figure 6 plots the log loss learning curve for the XGBoost model. We observe that the gap between the training and validation accuracies has been significantly reduced. Therefore, the XGBoost model is better suited to unseen data than our baseline.

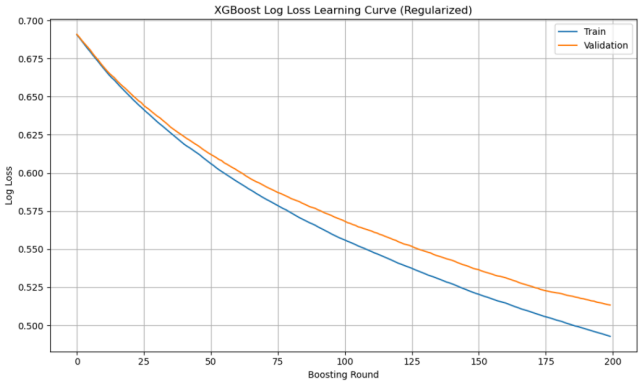


Fig. 6 Log Loss Learning Curve for XGBoost Classifier

Because blind-spot detection is safety-critical, our focus should be on minimizing false negatives. A driver should not be misinformed that the adjacent lane is clear. To this end, we utilize a confusion matrix to assess the performance of our classifier. Figure 5 shows the normalized confusion matrix for our XGBoost classification model. We can observe that the classifier achieves high recall for both classes, correctly identifying 82% of true positives and 78% of true negatives. However, the number of false negatives is relatively high at 18%.

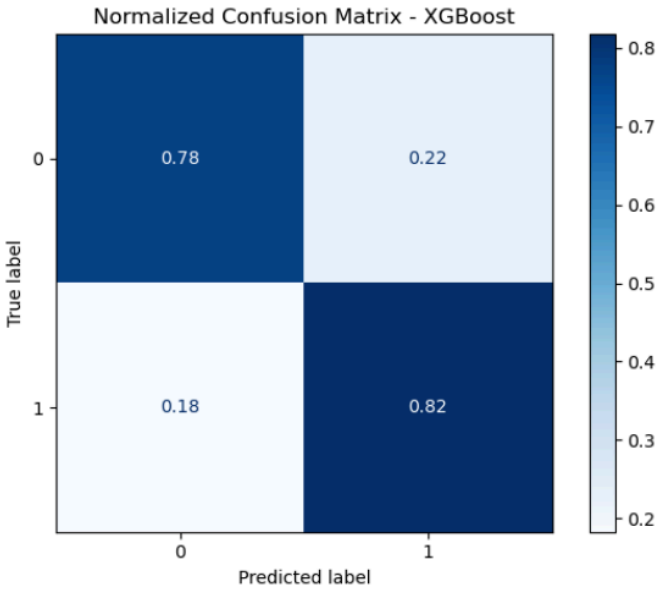


Fig. 7 Confusion Matrix for XGBoost Classifier

Industry standards typically require that a model have at least 70% accuracy to be considered useful [18]. In our case, we specifically aim to optimize recall, which helps minimize false negatives, while still ensuring that accuracy is at least 70%. To achieve this, we increase the number of boosting rounds during training and reduce the number of early stopping rounds, allowing the model to recognize true positive patterns [19]. Furthermore, we apply the Dropouts meet Multiple Additive Regression Tree (DART) algorithm, which randomly drops some decision trees during each boosting iteration, preventing overfitting to the negative class and improving correct positive class predictions, thus reducing false negatives [20].

IV. Results and Discussion

After applying these improvements, the final model accuracy is 70.50%. Figure 8 displays the new confusion matrix. We can see that the final model achieved a recall of 88%, successfully minimizing false negatives, though at the cost of a higher false positive rate (48%). In this scenario, however, a driver's safety is not jeopardized when they are inaccurately informed that a car is in their blind spot. At most, they are inconvenienced.

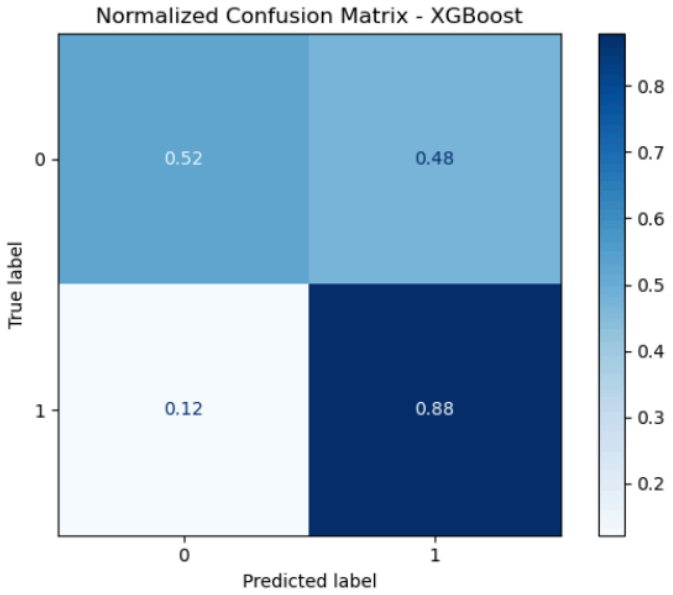


Fig. 8 Confusion Matrix for Recall-Focused XGBoost Classifier

To ensure that our model has maintained its generalizability, we must again check for overfitting. Figure 9 plots the log loss learning curve for the recall-focused XGBoost model. We can see that the gap between the test and validation accuracies is still small, so it is likely that our model is generalizable to unseen data.

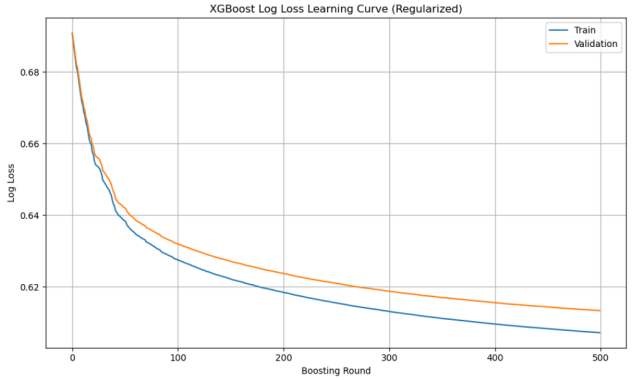


Fig. 9 Log Loss Learning Curve for Recall-Focused XGBoost Classifier

Next, we evaluate the model using cross-validation with five folds. Then, we compute the average accuracy, average recall, and 95% confidence interval across the five folds. The results are summarized in Table 2 below:

TABLE 2
Model Cross-Validation

Avg. Recall	Avg. Model Accuracy	95% Confidence Interval
79.90%	70.48%	[69.96%, 71.01%]

These results suggest several key takeaways about the final model. First, the high average recall is a great indicator of success. For blind spot detection, recall is more important than overall accuracy. The model is able to correctly flag most vehicles present in the blind spot with a small likelihood of false negatives. This aligns with the safety-critical nature of the application, where failing to detect a car could result in an accident. Secondly, the average model accuracy satisfies industry standards for practical use in real-world applications. This indicates that the model is an effective second line-of-defense for drivers. Furthermore, the narrow confidence interval strengthens the credibility of our results. It indicates that the model's performance is relatively stable across different data partitions, meaning that our model is not overly dependent on quirks in any single subset.

However, the model is not without limitations. Despite the high recall, the false positive rate is relatively high, meaning that the model frequently predicts that a car is in the blind spot when there is none. While this does not compromise driver safety, it could lead to mistrust in the system potentially causing drivers to ignore the alert system. This metric highlights how optimizing for recall in safety-critical applications often comes at the expense of precision. Another limitation is the overall accuracy tradeoff. While we managed to reduce overfitting compared to the baseline SVM model, we did so at the cost of ~6% in accuracy. This demonstrates the tension between performance and generalization. A deep learning approach likely would have offered better accuracy but would

require significantly more computational resources and time.

V. Conclusion

This project set out to eliminate blind spots using a shallow machine learning approach. By leveraging the CIFAR-100 dataset and HOG feature extraction, we built SVM and XGboost binary classifiers for car detection. Our baseline SVM model achieved high accuracy but suffered from overfitting, prompting a shift to XGBoost which incorporated regularization, early stopping, and DART boosting to improve generalization. The final XGBoost model prioritized recall, achieving 88% recall and 70.5% accuracy, reducing the risk of false negatives and meeting industry standards. Although this came at the cost of a higher false positive rate, it aligned with our primary goal of protecting drivers rather than avoiding unnecessary alerts. Cross-validation results confirmed the model's stability and generalizability.

Throughout this project, we learned the importance of proper data preprocessing and balancing performance metrics. We also gained insight into how regularization can dramatically influence a system's effectiveness. While further refinement is possible, such as incorporating video sequences and exploring higher-resolution inputs, this project demonstrated a real-time, lightweight model useful for embedded systems in modern vehicles.

VI. Distribution of Work

Ryan Perry was responsible for implementing portions of the baseline code and model improvements. He also wrote portions of the introduction, data set, technical approach, discussion, and conclusion sections of the final report. Ben Noyes was responsible for researching baseline solutions and model improvements, writing portions of the introduction and baseline sections of the final report and the proposed extension sections of the midterm report, and implementing portions of the baseline code. Daniel Phan was responsible for researching and writing portions of the baseline code and writing the conclusion section of the final report. He was also involved in improving the model to reduce overfitting. These members contributed to the baseline code roughly equally. Jake Miller was responsible for finalizing the report, helped make the slides, and helped out when people needed it.

References

[1] R. Tehrani, "What Are Blind Spots in Driving? Tips on How to Avoid Them." Driven2Drive.com. Accessed: May 2, 2025. [Online.] Availabl e: https://driven2drive.com/blog/blind-spots-in-driving-what-they-are-and-how-to-avoid-them/

[2] "Why Are Blind Spots So Dangerous?." DaneshgarLaw.com. Accessed: May 2, 2025. [Online.] Available: https://www.daneshgarlaw.com/blog/why-are-blind-spots-so-dangerous

[3] "4 vehicle safety technologies that can help reduce accidents." Secura.net. Accessed: May 2, 2025. [Online.]Available: https://www.secura.net/blog/4-vehicle-safety-technologies-that-can-help-reduce-accidents

[4] "Autonomous vehicles worldwide - statistics & facts." Statista.com. Accessed: May 2, 2025. [Online.] Available: https://www.statista.com/topics/3573/autonomous-vehicle-technology/

[5] A. Krizhevsky, Mar. 2024, "CIFAR-100," University of Toronto. [Online.] Available: https://www.cs.toronto.edu/~kriz/cifar.html

[6] K. Erdem. "t-SNE clearly explained." Medium.com. Accessed: May2, 2025. [Online.] Available: https://medium.com/data-science/t-sne-clearly-explained-d84c537f53a

[7] Y. LeCun, Y. Bengio, G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 14539, pp. 436-444, May, 2015, doi: 10.1038/nature14539.

[8] M. S. Suwal, "Car Detection using SVM." Kaggle.com. Accessed: May 2 , 2025. [Online.] Available: https://www.kaggle.com/code/merishnasuwal/car-detection-using-svm

[9] "Object Detection: The Definitive 2025 Guide." Viso.ai. Accessed: May 2, 2025. [Online.] Available: https://viso.ai/deep-learning/object-detection/

[10] K. Kumar, "Car Images classification using CNN. " Kaggle.com Accessed: May 2, 2025. [Online.] Available: https://www.kaggle.com/code/kshitij192/car-images-classification-using-cnn

[11] M. F. Asghar, "Vehicle Detection with Convolutional Neural Network." GitHub.com. Accessed: May 2, 2025. [Source code.] Available: https://github.com/faisalfaissi/Vehicle-Detection-with-Convolution-Neural-Network-

[12] H. Idrees, "Shallow Learning vs. Deep Learning: Is Bigger Always Better?." Medium.com. Accessed: May 2, 2025. [Online.] Available: https://medium.com/@hassaanidrees7/shallow-learning-vs-deep-learning-is-bigger-always-better-51c0bd21f059

[13] H. Li, J. Yang, G. Zhang, B. Fan. "Probabilistic support vector machines for classification of noise affected data," *Inf. Sciences*, vol. 221, pp. 60-71, Feb. 2013, doi: 10.1016/j.ins.2012.09.041.

[14] H. Han and X. Jiang. "Overcome Support Vector Machine Diagnosis Overfitting," *Cancer Inform*, vol. 13, no. Suppl 1, pp. 145-158, Dec. 2014, doi: 10.4137/CIN.S13875.

[15] I. Barkiah and Y. Sari. "Overcoming Overfitting Challenges with HOG Feature Extraction and XGBoost-Based Classification for Concrete Crack Monitoring, " *International Journal of Electronics and Telecommunications*, vol. 69, no. 3, pp. 571-577, Sept. 2023, doi: 0.24425/ijet.2023.146509.

[16] A. Nagpal. “L1 and L2 Regularization Methods, Explained.” BuiltIn.com. Accessed: May 2, 2025. [Online.] Available: https://builtin.com/data-science/l2-regularization

[17] “Configure XGBoost ‘early_stopping_rounds’ Parameter.” XGBoosting.com. Accessed: May 2, 2025. [Online.] Available: https://xgboosting.com/configure-xgboost-early_stopping_rounds-parameter/

[18] “Which is more important: model performance or model accuracy?.” Fiddler.ai. Accessed: May 2, 2025. [Online.] Available: https://www.fiddler.ai/model-accuracy-vs-model-performance/which-is-more-important-model-performance-or-model-accuracy

[19] “What is Boosting?.” Amazon.com. Accessed: May 2, 2025. [Online.] Available: https://aws.amazon.com/what-is/boosting/

[20] “Configure XGBoost Dart Booster.” XGBoosting.com. Accessed: May 2, 2025. [Online.] Available: https://xgboosting.com/configure-xgboost-early_stopping_rounds-parameter/