Detection of Sleepiness in Drivers Using EEG Brainwave Data

Group Number: Group 18

Names: Bernard Yeboah, Vaibhav Patel, Pavi Vijeyarajah

***Abstract*—** The primary aim of this project is to investigate the creation of a machine learning model that predicts driver drowsiness using EEG brainwave data, to improve road safety by enabling early identification of sleepiness. Utilizing the "Sleepy Driver EEG Brainwave Data" dataset, we performed in-depth preprocessing, feature selection, and preliminary exploratory analysis. We created multiple models, such as Logistic Regression, Support Vector Machines (SVM), Neural Networks, and ensemble methods. Additionally, we evaluated with the Random Forest model whilst achieving the highest accuracy so far 80.18% after implementing an ensemble method. Moreover, we assessed using the Random Forest model and hyperparameter tuning and cross-validation to enhance the model's performance optimization. Consequently, the Random Forest model was recognized as the best option because of its blend of accuracy and interpretability, which will be enhanced for real-world application.

# Introduction

As people progress through the years we see advancements in technology. Among these improvements, machine learning plays a key role in making lives easier by enhancing multiple different sectors that revolve around how we live. These sectors include healthcare, transportation, education, economy, threat detection, and many more. This paper uses a machine learning algorithm to detect drowsiness in drivers, which is defined by detecting the driver's mental state, given the characteristics of Electroencephalogram (EEG) brainwaves.

In Canada, there are many vehicles on the roads ranging from SUVs, sedans, pickup trucks, electric, and many others. Vehicles are utilized so often that they have become deeply engrained with our lives to the point we take them for granted. With vehicles becoming a daily necessity, it is crucial to give serious consideration to safety.

In 2022, Canadian statistics recorded nearly 1,931 fatalities from vehicular accidents. This is the highest record since 2020, averaging about 5 lives lost per day [1]. Accidents are caused by many common factors, including distractions, speeding, impaired driving, and fatigue [1]. Fatigue is a factor that goes unnoticed in car fatalities, particularly among individuals who may work early morning or come home late at night. It is a contributing factor to accidents caused by a lack of sleep, harsh schedules, stress, and other factors. Unlike alcohol and drugs, it is often difficult to detect when a driver is beginning to feel drowsy. If a driver’s mental state can be identified as awake or sleepy, we can reduce road accidents relating to fatigue. This is the motivation behind our study.

To address this problem, different machine-learning techniques were used, including supervised learning algorithms like Logistic Regression and SVM. Additionally, deep learning models, Multilayer Perceptron (MLP) Recurrent Neural Networks (RNN), and Random Forest, an ensemble technique were used. The accuracy of these four models was compared after optimizing them.

For the study, we used the “Sleepy Driver EEG Brainwave Data” dataset, which consists of data collected from four drivers using the Neurosky Mindwave sensor [2]. The dataset included a range of EEG metrics that aided in identifying patterns in drivers that indicated fatigue. The EEG metrics span several frequency bands, each reflecting different aspects of brainwave activity. Each frequency band represents a different brain state: Delta waves (deep sleep), Theta waves (transition to sleep), Alpha waves (calmness), Beta waves (alertness), and Gamma waves (higher cognitive functions) [6]. These brain waves are critical in identifying alertness and fatigue, as drowsiness can significantly impair a driver’s reaction time.

This paper is organized as follows: Section II reviews related work, Sections III, IV, V, VI, and VII describe the methodologies of Logistic Regression, SVM (Support Vector Machine), Random Forest, MLP (Multilayer Perceptron), as well as RNN, respectfully. Section VIII presents the results obtained from the models and the analysis of their performance, and Section IX evaluates the future steps that could have been taken.

# Related work

In machine learning, EEG brainwaves have been implemented in various studies outside the context of car accidents. One such study, the “Confused Student EEG Brainwave” study analyzed the brainwave patterns to study human behaviors and determine college students' confusion levels [3]. In this study, one of the models used an SVM for the binary classification to determine whether the students were confused or not. The authors reported an accuracy of 67.2% using the SVM model [3]. Another related study, which used EEG to detect confusion in students using Bidirectional Long Short-Term Memory (BiLSTM) with cross-validation, showed an accuracy of 73.3% [4]. The authors of this study concluded that gamma-1 waves are a key feature for detecting confusion in students [4].

# Methodology of using Linear regression

## Data pre-processing

The dataset contained a total of 3,735 EEG signal readings from the four drivers. These readings include several frequency bands: Delta, Theta, Alpha, Beta, and Gamma [6]. These frequencies correspond to different states of alertness in the brain. For our machine learning algorithm, the data was split into a training and test set with 80% for training and 20% for testing. The scale of these frequency bands has corresponding wavelengths as shown in Fig. 1.

The data was carefully processed to prepare for model training. Missing values in the dataset were checked and removed to minimize their impact on the analysis. Outliers were checked and corrected to ensure optimal model performance.

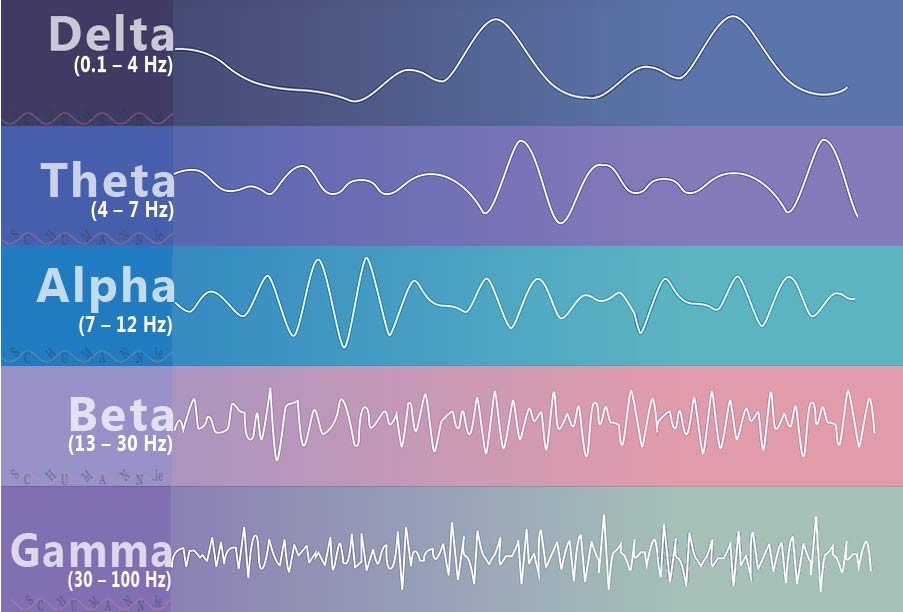


Figure 1. Image of different brainwave frequencies

## Apply Linear Regression

In the next step, linear regression was applied to the training set consisting of 2,988 signal data points. The input data included eight features, and the output was used to classify drowsiness in drivers. The model was trained and normalized using standard scaling to ensure that each feature equally contributes to the model. For the data to be compatible with the model, it was reshaped for 2D compatibility, which changes the array's shape without altering the training data [7]. This reshaping ensures that the data is in the correct format for the model because if it’s not in the right shape it may cause errors during training.

## Evaluation

To evaluate the performance of the linear regression model, we compared its metrics with the other models to identify the best-performing one to apply hyperparameter tuning. The primary performance metric we focused on was accuracy, along with other important metrics such as precision, recall, and F1-score. The accuracy was printed to show its performance.

# Methodology of using SVM

## Data pre-processing

The data pre-processing step was similar to Section III, using a dataset of 3.735 data signals. 80% of the data was allocated to the training set, with the remainder for the testing set.

## Apply SVM

Following this, the training data of 2,988 was used to train the SVM model. Input data contained eight features and the output data contained a classification of the state of the driver. The SVM was trained and normalized using standard scaling. The model was also reshaped for 2D compatibility.

## Hyperparameter Tuning

This machine learning model was selected for optimization because SVM performed better than other models. The random search was used as the hyperparameter optimization method. The search space was narrowed to only include essential parameters, such as the penalty parameter (C), kernel type, and gamma.

The C value is a regularization parameter that is inversely proportional to the margin size [12]. A smaller C results in a larger margin but increases classification errors, while a larger C reduces errors on the training set and has a smaller margin [12]. The C values chosen for optimization are 0.1, 1, and 10, which will be used in the random search to determine the optimal margin size.

The kernel type settings chosen were linear and radial basis function (RBF). The linear kernel is used for linear classification, while the RBF kernel is used for non-linear classifications [9]. The linear kernel can be calculated as

where we calculate the dot product of two input vectors xi and xj. The RBF can be calculated as follows:

where ||x-x’||2 represents the squared Euclidean distance between the vectors [10]. The represents the gamma parameter used to tune the spread of the kernel [10].

The gamma parameter settings chosen were scale and auto. This parameter essentially determines the extent of influence a single training example has [11]. Five random combinations of hyperparameters were sampled using these parameters, and five-fold cross-validation was applied. The time taken to tune the model was about 5.70 seconds.

## Evaluation

To evaluate the performance of SVM, we compared metrics and determined that this model would be utilized in hyperparameter tuning. Using the hyperparameter tuning, we determined that the optimal parameters to modify are a penalty parameter of 10, RBF kernel type, and scale as the gamma type. Using these optimal parameters we will generate better metrics than our default hyperparameters.

# Methodology of using Random Forest

## Data pre-processing

The EEG dataset was preprocessed to make it ready for implementation with the random forest model. The characteristics, which encompass different brainwave frequency bands like Delta, Theta, Alpha, Beta, and Gamma, were normalized with a StandardScaler to guarantee uniformity among all input variables. Identified missing values were eliminated, and outliers were handled to preserve the dataset's integrity. The dataset was subsequently divided into training and testing subsets, allocating 80% for training purposes and reserving 20% for testing, which guarantees adequate data for model training and impartial assessment.

## Feature engineering

To enhance the performance of the random forest model, feature engineering was performed to assess how each EEG metric affects drowsiness detection. Features were progressively incorporated and evaluated to determine their impact on the model's accuracy. The most relevant features were kept to enhance efficiency and minimize possible noise in the data.

## Apply Random Forest

The preprocessed data was used to train the random forest model. This data was made up of several decision trees, where each tree was trained on a random selection of the data and features. The model was optimized with hyperparameters, for instance, the number of trees (n\_estimators), maximum depth of trees (max\_depth), and the least number of samples needed for a split (min\_samples\_split) to reach peak performance. Following training, the Random Forest employed an ensemble method, which combined predictions from all decision trees to categorize drivers as either "awake" or "drowsy."

## Hyperparameter Tuning

Hyperparameter tuning is the process of choosing the best values for a machine learning model's hyperparameters, which stated simply are settings that govern the model's learning process [8]. In our Sleepy Driver Project, hyperparameter tuning served very essential for enhancing the performance of our machine-learning models. The main objective of hyperparameter tuning is to identify the values that generate the highest performance on a specific task. For our project, we concentrated on optimizing the random forest model, which is a popular ensemble learning method. In our optimization process, we used Optuna, which is a Bayesian optimization framework that examines a broad set of hyperparameter combinations and then determines the optimal configuration. The primary hyperparameters focused on included the number of trees in the ensemble (`n\_estimators`), the maximum height of the trees (`max\_depth`), the least number of samples needed to divide a node (`min\_samples\_split`), and the number of features taken into account for splitting at every node (`max\_features`).

After Optuna assessed each trial using the model's performance metrics, the tuning process indicated that a model configuration of 440 trees, a maximum depth of 27, a minimum sample split value of 8, and 50% of features used per split gave the best results. We additionally noticed from these results that setting max\_features to 0.5 (half of the features) performed better all around compared to other settings like sqrt or log2. Furthermore, we noticed trees with moderate depth combined with appropriate splitting thresholds reduced the likelihood of overfitting while preserving strong predictive capabilities.

# Methodology of using MLP

## Data-preprocessing

At the beginning of our project, the EEG dataset originally contained features such as attention, meditation, and different brainwave frequency bands (Delta, Theta, Alpha, Beta, and Gamma). These features underwent preprocessing to guarantee quality and consistency for training the model. The target variable classification, was the first to be transformed, being converted into numerical values with a label encoder to set it up for binary classification. The feature set was next and it was normalized with a standard scaler to standardize the input value range which improved the stability and performance of the neural network model (NNM). After preprocessing, the dataset was divided into training and testing sections, with 80% for training and 20% for testing. This allowed for an unbiased evaluation of fresh data while also guaranteeing the model received enough data for learning.

## Apply RNN

After training, we tested the MLP model's performance using the test set, which included predictions made by applying a 0.5 threshold to the model's output probabilities and classifying the results as either "awake" or "drowsy." The model's performance was also assessed to evaluate its effectiveness in categorizing the test data. This helped in evaluating the model's capability to identify drowsiness through EEG characteristics, which offered an understanding of its possible use in real and practical situations.

## Evaluation

After training, we tested the MLP model's performance using the test set, which included predictions made by applying a 0.5 threshold to the model's output probabilities and classifying the results as either "awake" or "drowsy." The model's performance was also assessed to evaluate its effectiveness in categorizing the test data. This helped in evaluating the model's capability to identify drowsiness through EEG characteristics, which offered an understanding of its possible use in real and practical situations.

# Methodology of using RNN

* 1. Data-preprocessing

For our project, the EEG dataset was arranged to make using a Recurrent Neural Network (RNN) easier. To ensure our model was consistently scalable and efficient, we normalized features using a StandardScaler. In the RNN model, the target variable was converted into categorical labels, which were used for classification tasks. The feature set was converted into a 3D format, which included dimensions that were suggestive of samples, time steps, and features, that follow the specifications of RNN architectures. This change allowed the model to handle a series of data efficiently, which assisted in recognizing temporal dependencies in EEG signals.

## Apply RNN

In our drowsy driver initiative, we constructed the RNN model employing a SimpleRNN layer containing 50 neurons and a ReLU activation function, succeeded by a dense output layer featuring a softmax activation for binary classification. This model was developed with the Adam optimizer and categorical cross-entropy loss. The training took place for 10 epochs with a batch size of 32, utilizing preprocessed 3D data. By employing this setup, the RNN successfully recognized patterns linked to driver fatigue from EEG signals.

## Evaluation

After the training, we assessed the performance of the RNN model using the test set. From here, predictions were made and the model's accuracy was determined by assessing the predicted classes versus the actual classes. From this, a classification report was developed that included the precision, recall, and F1-score, which helped to provide an accurate assessment of the RNN's ability to differentiate alert and sleepy states. These outcomes were compared alongside Logistic Regression, SVM, and MLP models to show that the RNN was very efficient when it came to analyzing a series of data.

# Results and Analysis

After comparing multiple models, we compared their performances. Initially, the models we used were set to their default values.

The default performance metrics of the logistic regression is 70%, for SVM it was 74%, for the random forest it was 76%, for MLP it was also 74%, and for RNN it was 73%.

Our overall results of the models provide important insights into their ability to detect driver sleepiness based on EEG inputs. Among the models evaluated, the random forest with an ensemble approach was the most effective, giving the highest accuracy of 80.18%. The metrics for the random forest are seen in Table I. This highlights the model's ability to combine predictions from multiple decision trees to deliver robust and consistent results..

Table I: Performance metric of the random forest using an ensemble approach

| Precision  (%) | Recall  (%) | F1-Score  (%) | Accuracy  (%) |
| --- | --- | --- | --- |
| 77 | 74 | 75 |  |
|  |  |  | 80 |

A heat map was used to display a confusion matrix that shows a correlation between true and predicted classification states of the final model as shown in Fig. 2. This model can tell the true awake and sleep states from the false awake and sleep states. The final model excels at recognizing alert individuals. The model exhibits a low rate of false positives, indicating that it isn't highly likely to incorrectly categorize alert individuals as sleepy. Taking everything into account, the model is well-balanced, demonstrating high accuracy in both alert and drowsy classifications.

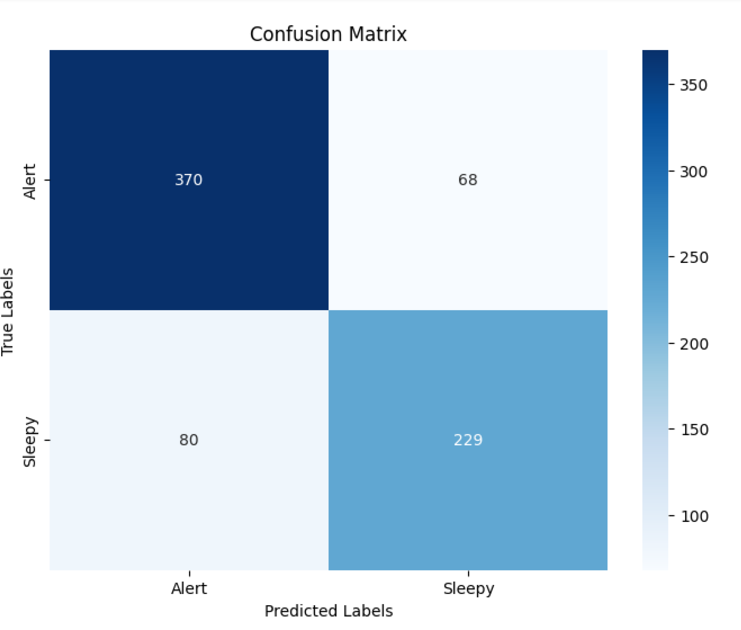


Figure 2: Confusion matrix for final ensemble model

The Support Vector Machine (SVM) model also performed strongly after tuning the parameters using random search, achieving an accuracy of 74% as seen in Table II. Additionally, SVM was also tuned using grid search but achieved a similar accuracy of 74%, shown in Table III. This additionally makes SVM a reliable alternative when it comes to classification tasks. The Neural Network and the RNN models were also close and had scores of 74% and 73%, while Logistic Regression achieved a lower accuracy of 70%, indicating that when it comes to complex tasks, it might not be as reliable as Random Forest or SVM.

Table II: Performance metric of SVM using random search

| Classification | Precision  (%) | Recall  (%) | F1-Score  (%) | Accuracy  (%) |
| --- | --- | --- | --- | --- |
| 0 | 82 | 72 | 77 |  |
| 1 | 66 | 77 | 71 |  |
|  |  |  |  | 74 |

Table III: Performance metric of SVM using Grid Search

| Classification | Precision  (%) | Recall  (%) | F1-Score  (%) | Accuracy  (%) |
| --- | --- | --- | --- | --- |
| 0 | 82 | 72 | 77 |  |
| 1 | 66 | 77 | 71 |  |
|  |  |  |  | 74 |

A heat map was used to show a correlation between true and predicted classification states in SVM after grid search as shown in Fig. 3. The model can accurately tell the true awake state from the false awake state but it has some trouble detecting true from false sleep state.

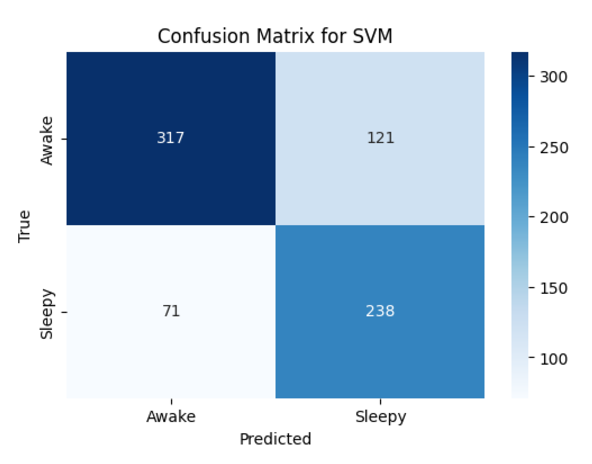
****

Figure 3: Confusion matrix for SVM after grid search

The random forest model also performed strongly after tuning the parameters using Bayesian optimization (BO), achieving an accuracy of 78% as seen in Table IV. The random forest model was also tuned using grid search but achieved an accuracy of 79%, as shown in Table V. This additionally makes SVM a reliable alternative when it comes to classification tasks.

Table IV: Performance metric of random forest using Bayesian optimization

| Classification | Precision  (%) | Recall  (%) | F1-Score  (%) | Accuracy  (%) |
| --- | --- | --- | --- | --- |
| 0 | 81 | 82 | 82 |  |
| 1 | 75 | 73 | 74 |  |
|  |  |  |  | 78 |

Table V: Performance metric of random forest using Grid Search

| Classification | Precision  (%) | Recall  (%) | F1-Score  (%) | Accuracy  (%) |
| --- | --- | --- | --- | --- |
| 0 | 81 | 84 | 82 |  |
| 1 | 76 | 72 | 74 |  |
|  |  |  |  | 79 |

Another heat map was used to show a correlation between true and predicted classification states in random forest as shown in Fig. 4. The model has difficulty detecting true from false awake state but it performs better in detecting true from false sleep state.

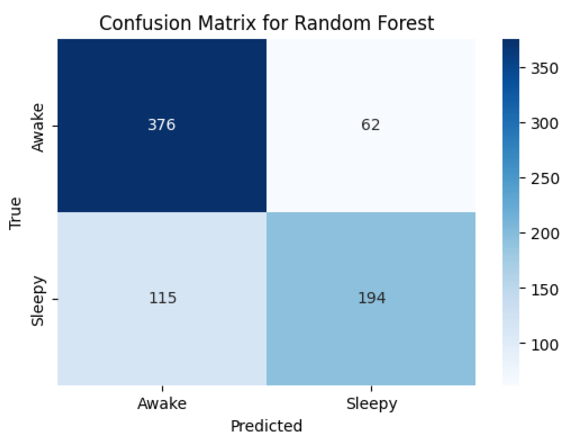
****

Figure 4: Confusion matrix for random forest

To summarize our overall conclusion, these results display the importance of ensemble methods and model selection when working with EEG data, especially for our project in drowsiness detection where the Random Forest ensemble stood out as the best solution.

# Next steps

After the initial hyperparameter optimization for the random forest, we switched to Bayesian Optimization using Optuna, as described in Section V, Part D. Based on the hyperparameters provided by Optuna, a few trials resulted in an accuracy rate of 0.79, which was the highest recorded accuracy, as mentioned in Section VIII. Following this, feature engineering was performed on the model using feature importance to identify the most impactful features to make predictions. The final model was retrained using the seven most important features. Grid search optimization was used to find the best parameters. These parameters along with those found in the Bayesian optimization, were used on an ensemble model with a voting classifier. The ensemble model achieved an accuracy of approximately 80%.

Appendix

Vaibhav Patel: Contributed to the proposal, progress report, and coding

Bernard Yeboah: Contributed to the presentation, progress report, and project report

Pavi Vijeyarajah: Contributed to the presentation, progress report, and project report

Code: <https://colab.research.google.com/drive/1hrsFR-IKXZ_xg6BzykDdInNnAZpUMEEr?usp=sharing>

References

[1] “Canadian Motor Vehicle Traffic Collision Statistics: 2022,” Government of Canada, https://tc.canada.ca/en/road-transportation/statistics-data/canadian-motor-vehicle-traffic-collision-statistics-2022.

[2] N. Hantsh, “Sleepy driver EEG Brainwave Data,” Kaggle, https://www.kaggle.com/datasets/naddamuhhamed/sleepy-driver-eeg-brainwave-data?resource=download.

[3] H. Wang, “Confused student EEG Brainwave Data,” Kaggle, https://www.kaggle.com/datasets/wanghaohan/confused-eeg.

[4] Z. Ni, A. C. Yuksel, X. Ni, M. I. Mandel, and L. Xie, “Confused or not confused?: Proceedings of the 8th ACM International Conference on Bioinformatics, computational biology,and Health Informatics,” ACM Digital Library, https://dl.acm.org/doi/10.1145/3107411.3107513.

[5] “Brain Waves,” Schumann, https://schumann.ie/brain-waves/.

[6] C. Terlouw, C. Bourguet, and V. Deiss, “Consciousness, unconsciousness and death in the context of slaughter. Part I. Neurobiological mechanisms underlying stunning and killing,” ScienceDirect, https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/brain-waves#:~:text=Slow%20waves%20correspond%20to%20theta,fast%20processing%20information%20during%20wakefulness.

[7] M. Asaduzzaman, “What is x\_train.reshape() and what it does?,” Stack Overflow, https://stackoverflow.com/questions/61555486/what-is-x-train-reshape-and-what-it-does.

[8] “Hyperparameter tuning,” GeeksforGeeks, https://www.geeksforgeeks.org/hyperparameter-tuning/.

[9] P. Shrestha, “Tuning SVM hyperparameters: Making your Classifier Shine like a pro!,” Medium, https://medium.com/@prayushshrestha89/tuning-svm-hyperparameters-making-your-classifier-shine-like-a-pro-8673639ddb16#:~:text=Imagine%20the%20kernel%20as%20a,and%20straightforward%20like%20a%20ruler.

[10] Sidharth, “The RBF kernel in SVM: A complete guide,” PyCodeMates, https://www.pycodemates.com/2022/10/the-rbf-kernel-in-svm-complete-guide.html.

[11] “RBF SVM Parameters,” Scikit Learn, https://scikit-learn.org/stable/auto\_examples/svm/plot\_rbf\_parameters.html.

[12] C. Sampaio, “Understanding SVM hyperparameters,” Stack Abuse, https://stackabuse.com/understanding-svm-hyperparameters/.

[13] C. Stryker, “What is a recurrent neural network (RNN)?,” IBM, https://www.ibm.com/topics/recurrent-neural-networks.