April 11, 2023

Detecting Eating Motions using Machine Learning Algorithms

Pooja Parameswaran, Jessilyn Dunn PhD

Introduction

Motion trackers have proved to be extremely useful in the upcoming industry as many actions are now detected. Eating is one of the actions that are not currently tracked, but doing so could yield positive outcomes. Nutrition apps can track users' eating habits as they continuously log their meals, alerted by timed notifications every day. Timed notifications can be tricky when users eat at different times every day. We hypothesize implementing a method to alert the user to log their meal intake while they are actively eating will work efficiently in a new nutrition app. I propose an algorithm that can detect the motion of eating using spatial data from a motion sensor, normally implemented in a smartwatch or smart phone. Using this data, I can detect patterns in the data and relate them to activities users perform in a day- focusing on eating. To achieve this, the algorithm must predict on all incoming actions from a motion detector and recognize a specific action as eating. In this project I explore which algorithms can distinguish eating movements.

Methods

A study at Fordham University occurred in October 2019, during which the Wireless sensor data mining lab tracked a range of activities participants performed. The study had 51 test subjects and each subject performed eighteen different activities for three minutes each, with an accelerometer tracking their movement. The dataset was structured as a function of time allowing activity classification as a function of time to make predictions. The accelerometer detected changes in velocity along the x, y, and z axes at a sampling rate of 20 hertz [2]. I modify the dataset to be represented with binary variables, classifying true eating activities as one, and the remaining activities as zero. A small portion of the data is held-out, which will be the subset that will not run through the model at all during training or validation iterations. The remaining data be divided for cross-validation. The dataset is divided into smaller subsets of "training" and "validation" and then trained and validated accordingly- k number of times. I utilize a five-fold cross validation by duplicating the data set five times and assigning train and validation subsets in different areas for every iteration.

We implement a sliding window to section the data into equivalent periods representing eating activity duration. Time is now accounted for as each window represents how long one eating motion should take. To reduce dataset complexity and represent the input data in the best way for eating activity prediction, we perform feature engineering: the process of transforming raw data into new features that represent the dataset with a singular value. I feature engineer the three axes in each window to create eleven singular measurements: mean, range, sum, absolute average deviation, average square root, average square, mode, standard deviation, minimum, and maximum. The novel dataset now contains only engineered features and the activity label (actual output). The activity label was derived from taking the mode of the binary activity label in each window. The novel feature-engineered dataset is run through three different ML models best suited for binary classification. A logistic regression model finds the mathematical significance of the relationship between two data factors, and then uses this relationship to predict one's value based on the other. A random forest model builds many decision trees that make predictions based on a previous set of decisions and outputs the majority vote between all its decision trees. A support vector machine creates a hyperplane in an N-dimensional space to classify binary data points [3].

Results

Model performance is analyzed using performance metrics: comparing the predicted output with the true output. I chose to analyze model performance with a specific set of metrics, including accuracy, Matthew's correlation coefficient (MCC), specificity, negative predicted value (NPV), and positive predicted value (PPV). The performance for each validation set is calculated and then averaged for each metric for all three classification networks (figure 1). The accuracy is the ratio of accurate predictions out of total predictions. MCC looks at the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions and calculates a ratio accounting for all four prediction classes. Specificity represents the percentage of true negatives. Precision measures the ratio of which the model correctly predicts on eating actions. The NPV and PPV describe the probability of a negative or positive result truly being a non-eating or eating measurement, respectively. All metrics but MCC can be represented between the values of zero and one, while MCC is represented between negative one, and one. The closer the performance metrics are to the lower end of the specified range, the worse the model is doing in this performance category. The closer the performance metric value is to the higher end, the better the model does in completing this task. The validation data is data the model has also been trained on, so an overfit model will also perform good as it has seen this data and learned it. The testing data is the held-out data that the three models have not seen before. The performance for the testing data across Logistic Regression and

Support Vector Machine performs a little worse compared to the performance from the validation data, but Random Forest performs similarly in both sets, and the high value metrics hold.

Analyzing a ROC-AUC curve and a Precision-Recall curve allows us to look at how predictions are made throughout the duration of testing. A receiver operating characteristic (ROC) curve displays the performance of classification models at all thresholds. The RF ROC curve has an area under the curve (AUC) of nearly 1.0, while Logistic Regression AUC is 0.9443, and SVM AUC is 0.914. The ROC curve allows us to analyze how many positive predictions are made through the different classification thresholds, while alerting us if the positive prediction is true or false. Useful, but in order to look at negative predictions and how the model handles class imbalance, we look at precision-recall curves. Precision-Recall (PR) curves display the tradeoff between precision and recall for different thresholds. Precision measures the model's ability to accurately pick out positive cases, while recall measures the model's ability to pick out all positive cases. F1 is a metric that can summarize the harmonic mean of precision and recall and was calculated to a ratio of 0.7645 with the Support Vector Machine, 0.764 with Logistic Regression, and 0.997 with Random Forest.

		Accuracy	Matthews Correlation Coefficient	Specificity	Precision	Negative Predicted Value	Positive Predicted Value
Validation	Logistic Regression	0.880	0.765	0.998	0.996	0.83340	0.9964
	Random Forest	0.996	0.992	0.998	0.996	0.9961	0.9973
	Support Vector Machine	0.642	0.470	1.0	0.396	0.4974	0.10
Testing	Logistic Regression	0.876	0.754	0.998	0.996	0.8316	0.9961
	Random Forest	0.998	0.996	0.999	0.998	0.9988	0.99821
	Support Vector Machine	0.640	0.137	0.769	0.421	0.7218	0.4213

Figure 1: *Performance of LR, RF, SVM, models between validation and testing sets*

Discussion

The performance metrics provided insight into each model's performance. We term eating activity predictions as positive and non-eating activity predictions as negative. Random Forest is structured as an aggregation of independent decision trees, so it undergoes heavy analysis and strict rules to make a prediction, for this reason I believed Random Forest would perform the best. The results agree, and there is up to 99% accuracy in the RF model's predictions. Looking at the MCC and Positive Predicted Value for Random Forest, we can see the model is able to predict on the occurrence of eating activity; and able to predict correctly as the prediction input data is fed through the model. Random Forest performs well with the ROC-AUC curve as well with a high AUC metric representing that many positive predictions the RF model performs are true. The precision-recall curve also displays the model's capability of accurate predictions in testing. As more positive cases are found by the RF model, the accuracy of picking out only true positive cases is retained- suggesting Random Forest can predict eating activity from incoming data while also not confusing negative predictions (non-eating) as positive (eating). Logistic Regression proves predict well also, looking at the performance metrics. Logistic regression maintains a high accuracy of 87.6% during testing, indicating the model can make 87% of accurate predictions out of total predictions on novel input data. Logistic Regression also has a high PPV value at 0.996, indicating the model's confidence in predicting positive actions, however, the NPV value is at 0.832 indicating the model has mistaken non-eating motions for eating motions. A logistic regression model weighs which input parameters most directly correlate with the activity output. This is a very proportional relation, so error can be prone compared to a RF model. The SVM has a lower performance with 64% accuracy in testing but is understandable as the model creates a 32-dimensional hyperplane between positive and negative classes and makes predictions splitting the line. I noticed that Random Forest performed well and has results optimal enough to begin using in an application to track nutrition. This binary classification algorithm is ready to be utilized with an accelerometer-contained device to recognize eating actions.

References

[1] Gandhi, Rohith. "Support Vector Machine - Introduction to Machine Learning Algorithms." *Medium*, Towards Data Science, 5 July 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[2] *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*, https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+.

[3] Noble, William S. "What Is a Support Vector Machine?" *Nature News*, Nature Publishing Group, https://www.nature.com/articles/nbt1206-1565.

[4] *Code Studio*, https://www.codingninjas.com/codestudio/library/sliding-window.

[5] Yiu, Tony. "Understanding Random Forest." *Medium*, Towards Data Science, 29 Sept. 2021, https://towardsdatascience.com/understanding-random-forest-58381e0602d2.