Classifying Right vs Left Tapping from TD-fNIRS Data using Machine Learning

Introduction/Motivation:

We analyzed data from a very recent paper, which was published on January 18, 2022 by Kernel, a neurotechnology startup that developed a novel TD-fNIRS headset, called Flow. TD-fNIRS is one of the most accurate noninvasive optical brain imaging devices but is typically a very costly large machine. Kernel is currently conducting many studies using Flow, such as assessing the effects of meditation and aging on the brain. Kernel's ultimate goal is to quantify the brain by developing metrics that would allow users to understand their brain by wearing Flow for a few minutes every day. In short, Kernel wants to create a fitness tracker for the brain. This project aims to give another purpose to the data collected with Flow. By using machine learning to classify hemodynamic data, we could use Flow for another compelling idea: control. Users would potentially be able to replace their computer mouse with a mere thought, or finger tap in this case.

Problem Definition:

The dataset from the paper aforementioned is a record of the hemodynamic responses to finger tapping tests. This data was originally analyzed by Ban et al. through cluster-based permutation tests in order to identify right vs left tapping using oxygenated (HbO) and deoxygenated hemoglobin (HbR) concentrations in the brain. The aim of this analysis was to validate that the Kernel Flow headset maintains or improves upon the performance of existing benchtop TD-fNIRS systems. The problem we are trying to address is assessing which, if any, classification-based machine learning model will provide the most accurate way of classifying TD-fNIRS data. We decided to implement three commonly used algorithms and investigate their performance.

Methods:

1. **Data Preprocessing**
    a. **Channel Selection**

We were given data from 2 patients who performed finger tapping tasks over the course of ~800 seconds with breaks in between. Patient 1 completed two runs, whereas patient 2 only completed one. This gave us a total of 3 datasets to work with. Each dataset was a matrix with rows=2*number of channels and columns=number of data points per channel. Each original matrix had a number of rows that equaled twice the number of channels on the headset because each channel detected both HbO and HbR data (Fig. 1A). On top of that, some channels did not detect any data, represented

by the white lines in Fig. 1A. We removed these channels to get preprocessed datasets, which only had about 1000 channels out of the ~2000 channels used.

b. Labels Generation

In order to create a binary array for the labels, where 0 corresponds to right tapping and 1 corresponds to left tapping, we needed to manually assign points to the right and left intervals. We were given right and left tapping labels in excel sheets for only ~583 data points for the ~800 second time interval for each run, compared to an average of 2000 data points (excluding rest periods) per channel per run. A sample of these excel sheets (task_events data frame) can be found in our code under the Labels Setup section. We used a for loop and if statements to assign each recorded point to either left or right tapping based on the intervals of tapping we were given. We also removed all data points that correspond to resting periods (labeled “nan” in the experiment_type column of the task_events data frames).

c. Data/Signal Processing

To ensure we don’t introduce any additional factors that might affect the machine learning algorithms’ performance, we followed the same signal processing techniques as Ban et al. First, we applied a moving average with 100 windows on both the datasets and labels. Second, we implemented a Butterworth lowpass filter with a cutoff frequency of 0.1 Hz. A sample of this processing flow is shown in Fig. 1B.

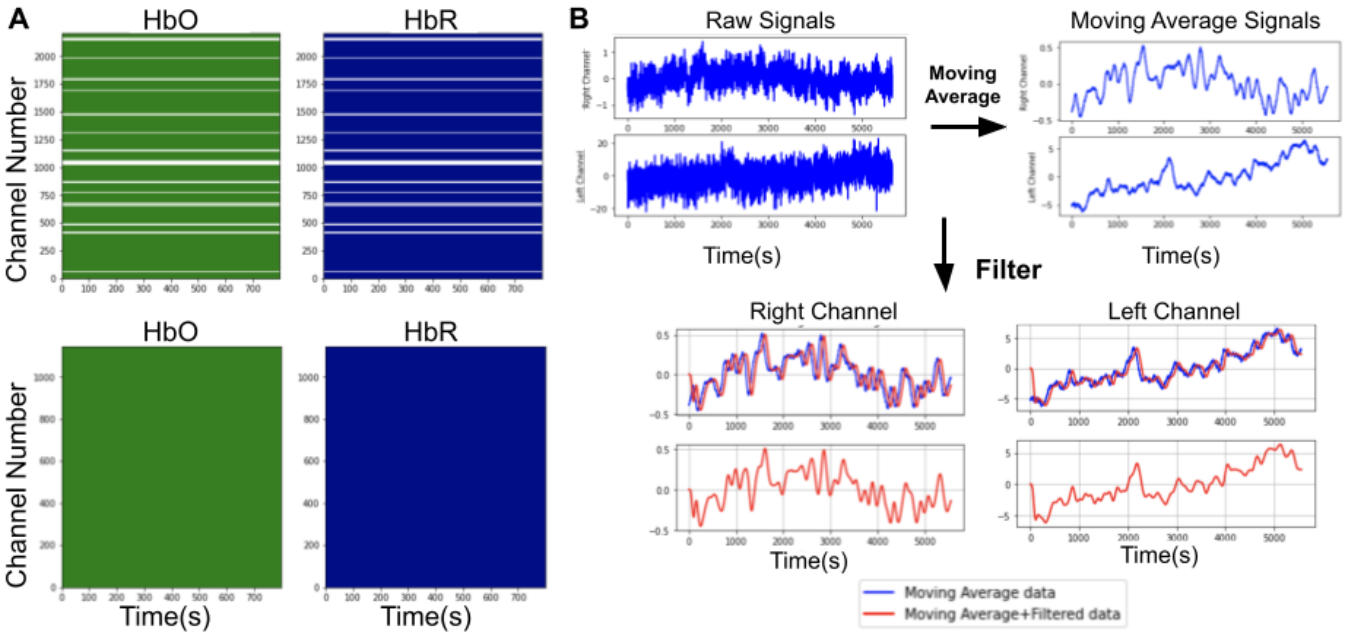

Figure 1: Data Preprocessing and Visualization. (A) Identifying which channels picked up signals for both oxygenated (HbO) and deoxygenated (HbR) hemoglobin. White channels are empty and were removed. (B) Applying moving average and low pass filters to eliminate noise. Sample channel picked from patient 1 run 1.

2. **Logistic Regression**

To implement logistic regression in JupyterNotebook, we used the LogisticRegression function inside linear_model from the Scikit-learn library (sklearn). Specifically, we used scikit-learn 1.0.2. This model can be easily imported using the following statement: **from** sklearn.linear_model **import** LogisticRegression. Overall, this model was chosen because it provides easy explainability and extrapolation. It is also safe to assume that the independent variable (tapping) is linearly related to the dependent variable (hemoglobin levels). First, we fitted a logistic regression model on a single channel's data to assess if that was enough for the model to train properly. We explored hyperparameter tuning through estimations of prediction errors based on a 20-fold stratified cross-validation. Specifically, we optimized the regularization penalty and the C parameter for this single-channel model. Second, we fitted logistic regression models on all channels for the 3 different datasets, for a total of 3 models. Finally, we fitted logistic regression models on HbO and HbR data separately for each dataset, for a total of 6 models. We used two evaluation metrics to compare the models: accuracy and confusion matrix. We used the accuracy score of the predicted vs true labels from a 10-fold cross-validation and confusion matrices to see the number of TP, TN, FP, and FN the models predicted.

3. **Partial Least Squares-Discriminant Analysis (PLS-DA)**

To implement PLS-DA, we used the PLSRegression model from the Scikit-learn library, with a y-matrix with binary values of 0 or 1 to represent the categorical output variable. This model can be imported using the following statement: **from** sklearn.cross_decomposition **import** PLSRegression. We picked this model because it is relatively easy to train and it was specifically created for binary classification of categorical variables, such as the right- and left-tapping blocks in the data, making it a perfect fit for this experiment. Additionally, it allows for easy visualization of the correlations between values using the scores and loadings matrices. First, we fitted PLS-DA models on all channels for the 3 different datasets, for a total of 3 models. Second, we fitted PLS-DA models on HbO and HbR data separately for each dataset, for a total of 6 models. Therefore, we had 9 PLS-DA models overall. We used accuracy (with 10-fold cross-validation) and confusion matrix as evaluation metrics. Since PLSR outputs continuous predicted values, we had to create a new predicted values array for each PLS-DA model by assigning 0 to predicted values smaller than 0.5 and 1 to predicted values greater than or equal to 0.5.

4. **Support Vector Machines (SVM)**

To implement SVM in JupyterNotebook, we used the SVC function inside svm from sklearn. This model can be easily imported using the following statement: **from** sklearn.svm **import** SVC. We picked this model because it is relatively easy to train and

is a binary classifier, which is exactly what we need for this dataset. Indeed, SVM models determine a dividing hyperplane between two classes of data by maximizing the margin (distance between support vectors). We initially used patient 1 run 1's dataset to fit 4 SVM models, one for each kernel, to determine the kernel with the highest accuracy. We then used the optimal kernel to fit SVM models on all channels for the 3 different datasets, for a total of 3 models. Lastly, we fitted SVM models on HbO and HbR data separately for each dataset, for a total of 6 models. We thus had 9 SVM models overall. Here again, we used accuracy (with 10-fold cross-validation) and confusion matrix as evaluation metrics. We explored hyperparameter tuning through estimations of accuracy based on a 10-fold cross-validation. Specifically, we optimized the C and gamma parameters for the patient 2 model.

Results:

1. **Logistic Regression**

The initial logistic regression model had accuracy metrics displayed in the first row of Table 1. The accuracy corresponds to the testing error associated with applying the model to testing data from another patient that the model hasn't seen before. The cross-validation error was from a 20 fold cross-validation. The confusion matrix for this model is displayed in Fig. 2A. There was an increase in precision and recall, while accuracy stayed the same when we performed hyperparameter tuning for regularization (Table 1). The model's accuracy was highest at 60.8% for C=0.001 (Fig. 2B).

Table 1: Single Channel Logistic Regression Model Evaluation Metrics for Regularization Hyperparameter Tuning.

Regularization Penalty	Accuracy	Precision	Recall
l2	0.597721667645331	0.532269407121151	0.531645569620253
l1	0.59848796241926	0.81816674552928	0.812567518905293
Elasticnet	0.598487962419260	0.817926666554264	0.812207418077061

The confusion matrix along with the reported accuracy values (10-fold cross-validation) for models built on all channels across the 3 different datasets are shown in Fig. 2C. Fitting the models using all channels showed a significant change in accuracy only for patient 2's dataset (accuracy=0.62) compared to the single-channel model. The average accuracy of the logistic regression models over the three datasets was 56.2% when it was trained with all HbO and HbR channels. We also tried running

models on HbO and HbR data separately and observed no significant change in accuracies (not shown). We observed that the single-channel model tends to predict 1s more than 0s.

2. PLS-DA

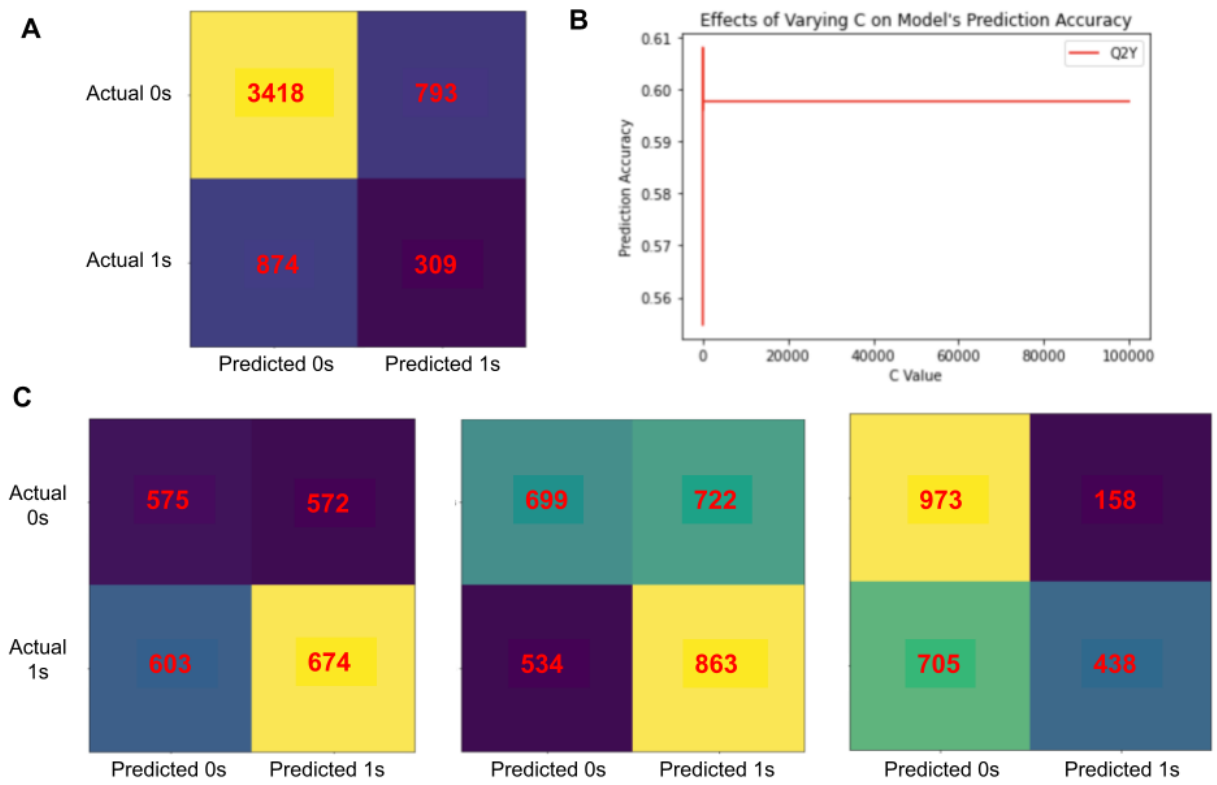

Figure 2: Evaluation Metrics for Logistic Regression Model. (A) Confusion matrix for single channel logistic regression model. Cross validation accuracy=0.59772166.(B) Effects of varying the C parameter on the prediction accuracy for the single channel model. (C) Confusion matrices for models fit to all channels in order from left to right (patient 1 run 1, patient 1 run 2, patient 2). The 10 fold cross validation accuracy values in that same order were (0.5152640264026402, 0.5542938254080908, 0.6204925241864556).

The PLS-DA models had an average accuracy of 55.1% when it was trained with all HbO and HbR channels. The confusion matrices for all 3 datasets can be seen below in Fig. 3. Like with logistic regression, there was very little change when the model was run with HbO and HbR data separately (not shown).

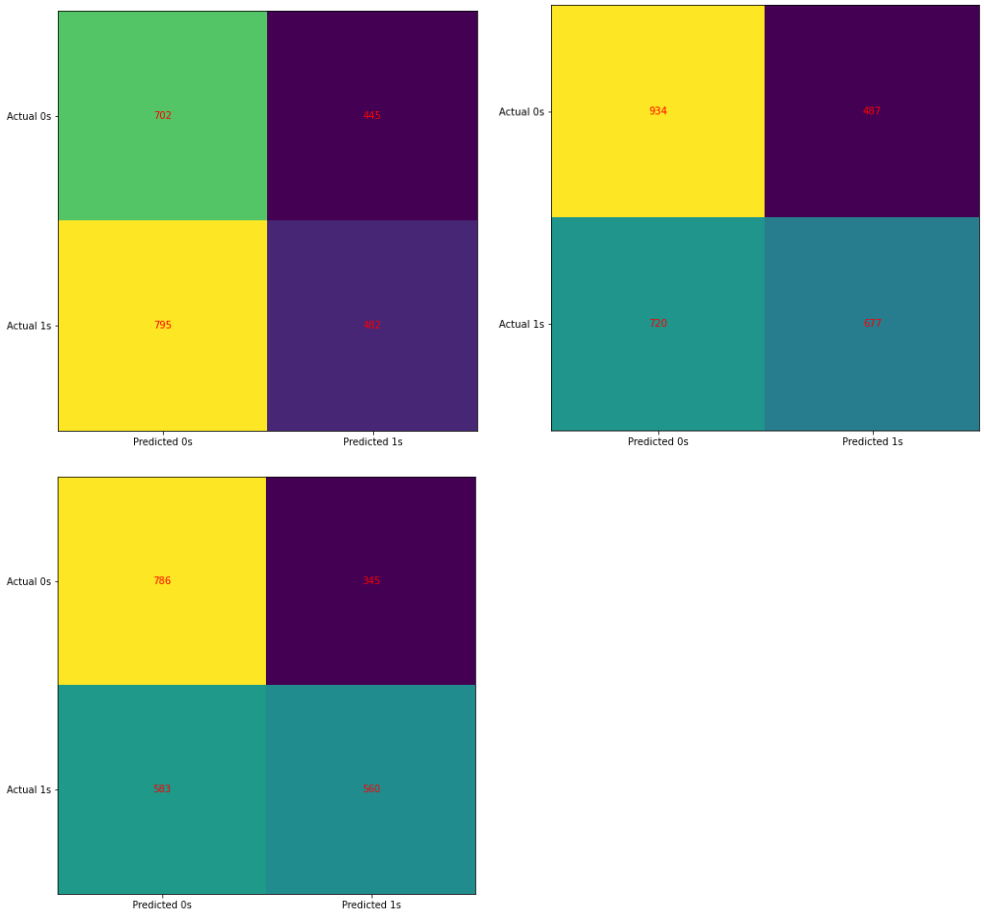

Figure 3: Confusion matrices for the PLS-DA predictions. Top left: patient 1 run 1. Top right: patient 1 run 2. Bottom left: patient 2.

A loadings plot was created from the data for Patient 1, Run 1, shown below in Fig. 4. There was no separation between HbO and HbR channels observable.

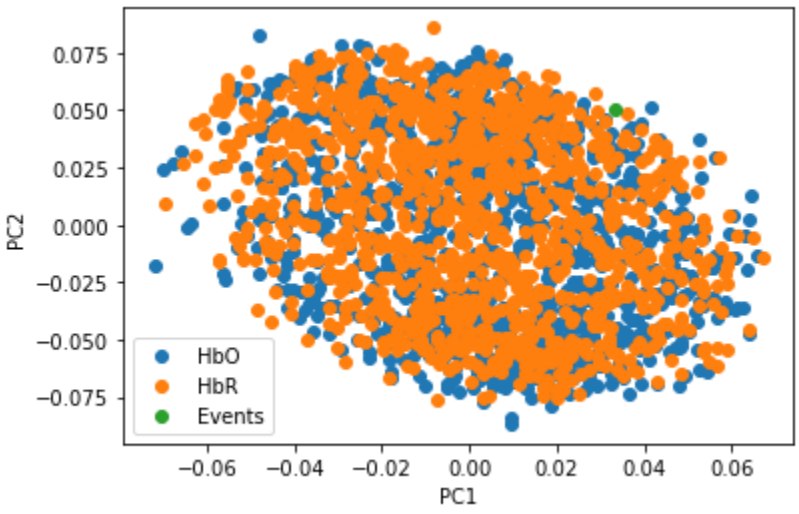

Figure 4: Loadings Plot for PLS-DA model for Patient 1, Run 1

3. SVM

Fitting SVM models with different kernels on patient 1 run 1's dataset led us to pick radial basis function (rbf) as the optimal kernel; the rbf model had the highest accuracy (not shown). The SVM model had an average accuracy of 58.6% over the three datasets when it was trained with all HbO and HbR channels. The confusion matrices for all 3 datasets can be seen below in Fig. 5. Similarly to logistic regression and PLS-DA, there was no significant change in accuracies when the SVM model was run with HbO and HbR data separately (not shown). We also tried running models on HbO and HbR data separately and observed no significant change in accuracies (not shown).

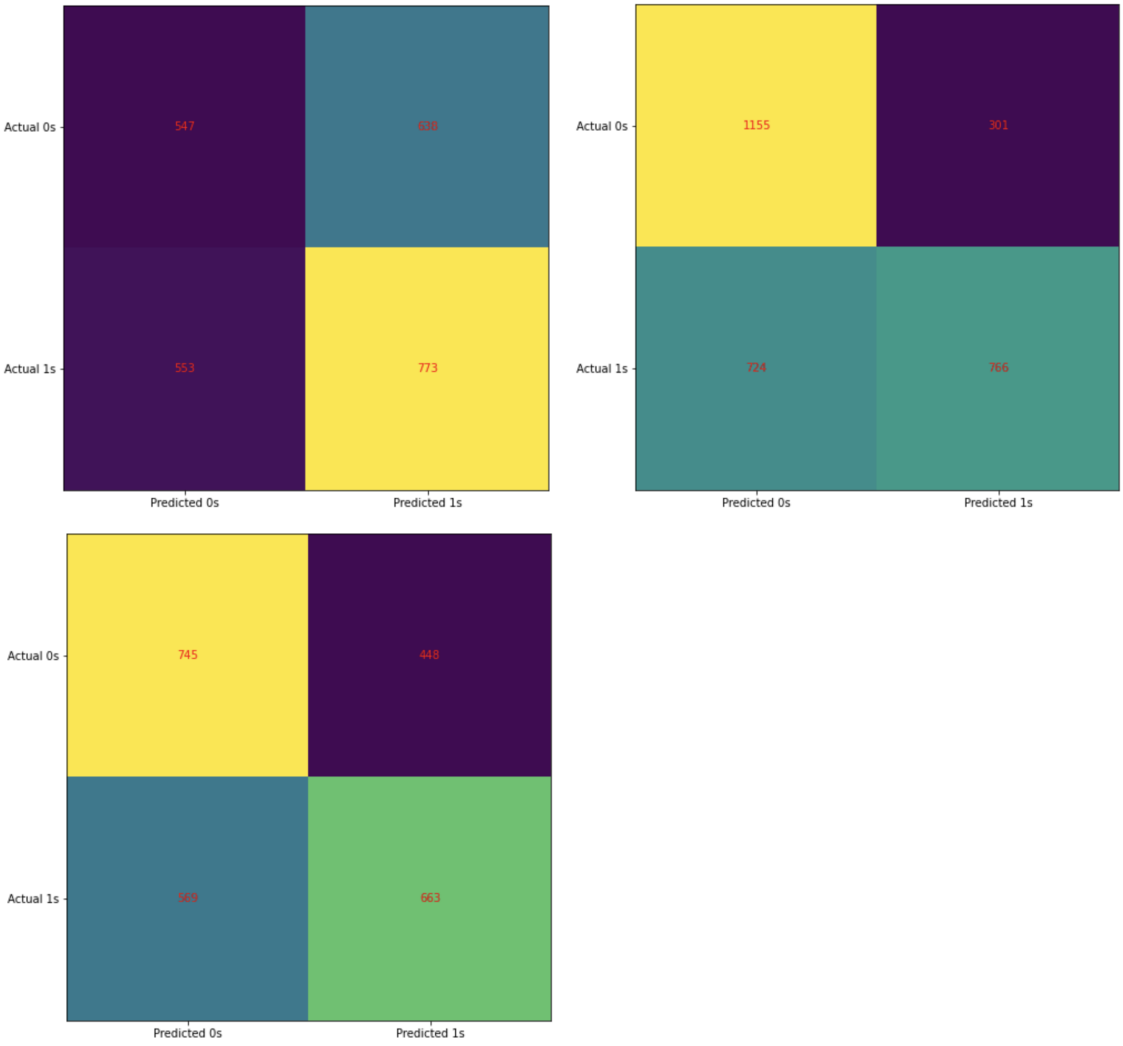

Figure 5: Confusion matrices for the SVM predictions. Top left: patient 1 run 1. Top right: patient 1 run 2. Bottom left: patient 2.

The effects of the C and gamma values on the patient 2's SVM model are presented in Fig. 6. For all values of C greater than or equal to 0.1, the model's accuracy was highest at 59.5%. For all values of gamma smaller than or equal to 0.001, the model's accuracy was highest at 59.5%.

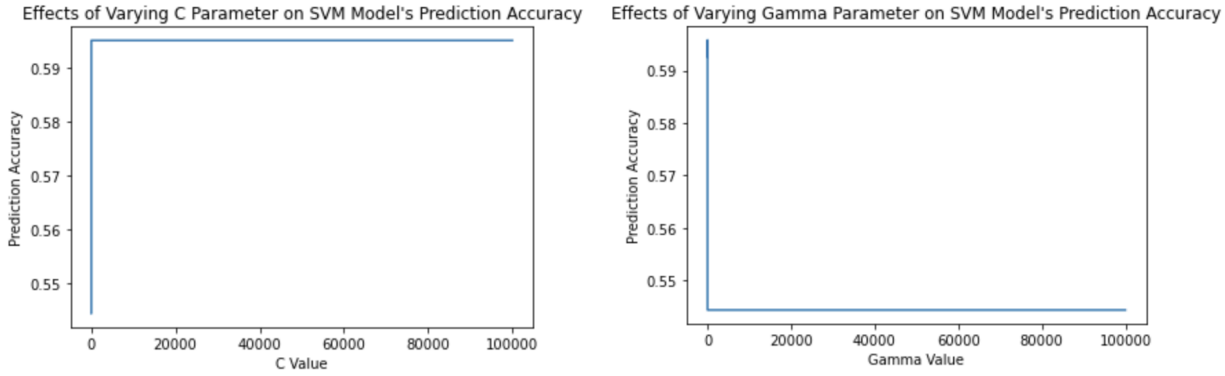

Figure 5: Effects of the C and gamma values on the SVM model's accuracy. Left: C parameter's effect on SVM model's accuracy. Right: Gamma parameter's effect on SVM model's accuracy.

Conclusion and Closing Remarks:

Out of the three machine learning algorithms we implemented, SVM with an rbf kernel performed best. Out of the three datasets we used, patient 1 run 2's had the highest accuracy across all three models. Unfortunately, all of the models developed had accuracy scores between 50 and 65%, which leads us to consider these models as inefficient or decent at best. These models need much improvement or are simply not the right ones to classify these datasets. One thing to note is that our initial goal was to train a model using two of the datasets and classify the third dataset. However, each dataset had a different number of channels that picked up data. This inconsistency in the number of rows of the data matrix prevented us from concatenating datasets or from classifying a new dataset. Another thing to note is that we tried scaling our data before fitting the models but that led to much less accurate models. Finally, for all models, right tapping (labeled as 0) was predicted many more times than left tapping (labeled as 1), which could be caused by an imbalance in the data.

Project breakdown:

Ahmed:

- Code: PLS-DA, labels setup
- Report: PLS-DA methods, PLS-DA results

Sara:

- Code: Logistic regression, labels setup, data processing, hyperparameter tuning, confusion matrix

- Report: Problem definition, data preprocessing, logistic regression methods, logistic regression results

Soulaimane:

- Code: SVM, data preprocessing, data processing, cross-validation, accuracy scores
- Report: Introduction/motivation, SVM methods, SVM results, conclusion and closing remarks

References:

- Karim, H., Schmidt, B., Dart, D., Beluk, N., & Huppert, T. (2012). Functional near-infrared spectroscopy (FNIRS) of brain function during active balancing using a video game system. *Gait & Posture*, *35*(3), 367–372. https://doi.org/10.1016/j.gaitpost.2011.10.007
- *Products*. kernel. (n.d.). Retrieved March 6, 2022, from https://www.kernel.com/products
- Ban, H. Y., Barrett, G. M., Borisevich, A., Chaturvedi, A., Dahle, J. L., Dehghani, H., Dubois, J., Field, R. M., Gopalakrishnan, V., Gundran, A., Henninger, M., Ho, W. C., Hughes, H. D., Jin, R., Kates-Harbeck, J., Landy, T., Leggiero, M., Lerner, G., Aghajan, Z. M., … Zhu, Z. (2022). Kernel flow: A high channel count scalable time-domain functional near-infrared spectroscopy system. *Journal of Biomedical Optics*, *27*(07). https://doi.org/10.1117/1.jbo.27.7.074710
- Yamada, Y., Suzuki, H., & Yamashita, Y. (2019). Time-domain near-infrared spectroscopy and Imaging: A Review. *Applied Sciences*, *9*(6), 1127. https://doi.org/10.3390/app9061127