OcculoAnnelid

Damian Lin, Ruoyang Lu, Xinchen Xu, Abhay Mahajan, Freya Singh, Karl Lin

2021

Contents

1 Executive Summary

The aim of this project was to implement the 1997 Nokia Snake game by allowing players to provide input using left and right eye movements. This was achieved by attaching electrodes to the player's forehead, which was connected to a Spikerbox to detect changes in their brain signals. The team's physics members worked to read the signals in a manner compatible for the data members to use in the classification process.

The classification methods used were K-Nearest Neighbours (KNN), Logistic Regression (LR), Naive Bayes (NB), Random Forests (RF), Support Vector Machines (SVM), and a simple heuristic classifier. An analysis on accuracy and feature selection was conducted in order to determine which classifier and features were most appropriate for the classification of left and right eye movements. After thorough investigation, the team's main findings for ideal placement of electrodes were on either side of the player's right eye. A Random Forest classifier was used to detect eye movements, with catch22 for feature extraction. No feature selection was implemented as it was found to make the accuracy and time of the classifier worse.

The team's motivation behind implementing the project in a trivial environment was fueled by the possibility of allowing future studies to reproduce the results in a more valuable setting. eye-movement tracking is an essential measure in studying human behaviour with the potential of furthering numerous fields such as medicine, marketing, education, and gaming. The analysis opens opportunities to implement alternatives in controlling physical devices, benefiting the disabled; collecting data to help marketing companies better target customers; creating engaging educational dashboards to improve students' learning. All are just a few practical applications in which the study can be used to excel human-computer interaction for the evolving technological future.

2 Data Collection and Preprocessing

Electrooculography (EOG) measures the potential differences between the positive pole formed by cornea at the front and the negative pole formed by retina at the back of an eyeball and thus can be used to detect eye movements and blinks [1]. We chose *EOG* to be the controlling signals for our Snake game [3] and categorized four distinct eye movements as user instructions to the snake: left, right, blink, and still.

2.1 Data Collection

To measure eye movements, we used a Heart and Brain SpikerBox to record 1-channel *EOG* signals, where two electrodes are positioned around the eyes to detect dipole movements. The positioning of electrodes is determined by the "cleanliness" of the *EOG* signals recorded. We identified four possible electrode positions, *1)* at either side of the right eye and above the eyebrow (*across*), *2)* directly above and below the right eye (*updown*), 3) diagonally across the right eye (*diag*), and *4)* at the left side of the left eye and the right side of the right eye (*wide*). Signals recorded from all four positions were collected and pre-processed by a Butterworth filter with bandwidth $1 \sim 100\,\text{Hz}$ and a Notch filter with centre frequency 50 Hz as per the recommended settings for the *BYB Spike Recorder* software. Fig.1a illustrates the filtered *EOG* signals from each electrode position.

To analyze the "cleanliness" of recorded signals, we used Fast Fourier Transform to transform the signals from time domain into the frequency domain where individual spectral components of the signals can be analyzed. Noises are generally characterized by their high frequencies and therefore appear as "spikes" at high frequencies in the spectrogram. Further research shows that *EOG* signals have a typical frequency spectrum of $0.1 \sim 20\,\text{Hz}$ [1] or $1 \sim 50\,\text{Hz}$ [7]. Given the above information, we decided to consider signals with frequency higher than 50 Hz to be noises. Furthermore, we defined the "noise ratio" of signals to be the ratio of the magnitude of power spectrum above 50 Hz to the magnitude of the entire power spectrum, as described by Equation (1).

$$R_{\text{noise}} = \frac{\sum_{k=f_{\text{noise}}}^{K-1} X[k]}{\sum_{k=0}^{K-1} X[k]} \tag{1}$$

where

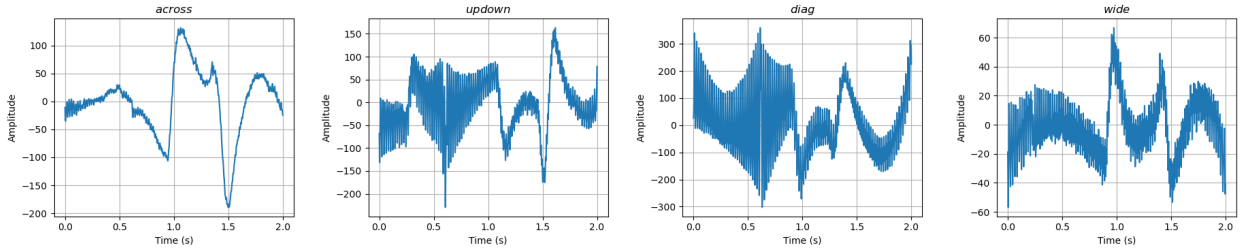
$$X[k] = \sum_{n=0}^{N-1} f[n] e^{(-i2\pi(\frac{nk}{N}))}$$

and

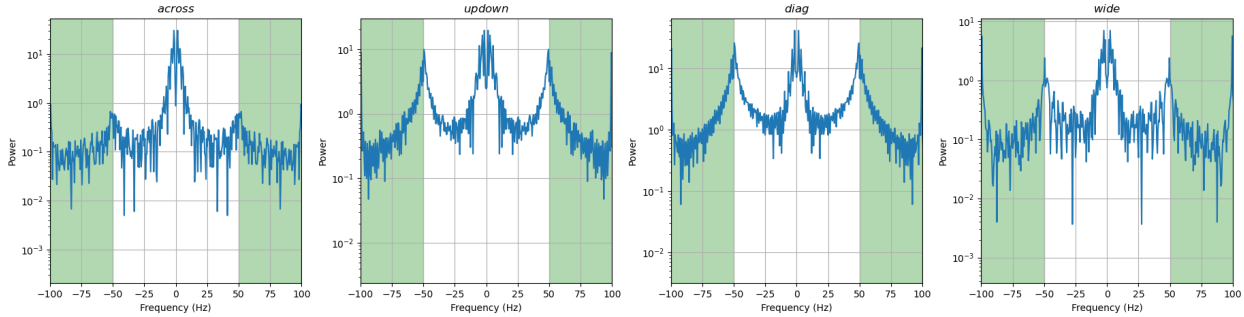
$$f_{\text{noise}} = 50\,\text{Hz}$$

is the frequency above which signals are considered noises. The noise ratio is derived by summing over the magnitude of power spectrum and that of the entire power spectrum. It reflects the "percentage" of noises presented in the signals. Fig.1b illustrates the Fast Fourier Transform of *EOG* signals from each electrode position, and the green areas highlight the noises.

To determine the best electrode position, we took 10 measurements from each electrode's position and calculated the noise ratios of the signals recorded from each position. The electrode position with the lowest average noise ratio was considered the optimal position. The results are tabulated in Table 1a. We can clearly see that the position of either side of the right eye and above the eyebrow (*across*) is the most optimal position as it produces the cleanest signals.



(a) Filtered *EOG* signals



(b) Fast Fourier Transform of the above *EOG* signals. Note that the green areas highlight signals below −50 Hz or above 50 Hz and are considered noises.

Figure 1: Filtered *EOG* signals and their spectrogram after Fast Fourier Transform. The columns present electrode positions *across*, *updown*, *diag*, and *wide*, respectively, counting from the left.

2.2 Data Prepossessing

Now that the most optimal electrode position has been determined, we further analyzed the effect filtering have on *EOG* signals. The Notch filter is a band-stop filter with a very narrow bandwidth centred at 50 Hz and attenuates 50 Hz signals while leaving signals at other frequencies intact. It was applied to the *EOG* signals to remove interference induced from AC power supply, which has 230 V amplitude and 50 Hz frequency in Australia. The Notch filter was kept in our design without further investigations, as the problem of AC power supply interfering with recording will always persist.

The Butterworth filter is a band-pass filter that attenuates all signals outside its bandwidth and leaves those inside it intact. In the previous experiment the bandwidth of the Butterworth filter was set to 1 $\sim$ 100 Hz, which was the recommended setting for the *BYB Spike Recorder* software. To find the bandwidth that yields the most distinguishable *EOG* signals between the four eye movements, we again calculated the noise ratio of each *EOG* signals filtered by bandwidths with different upper cut-off frequencies. As stated before, *EOG* signals have a typical frequency spectrum of 0.1 $\sim$ 20 Hz [1] or 1 $\sim$ 50 Hz [7]. Thus, we conducted our experiment by decrementing the upper cut-off frequency of the Butterworth filter from 100 Hz to 20 Hz in steps of 20 Hz and calculating the noise ratio of each signal.

Table 1: The average noise ratio of each electrode position and from each upper cut-off frequency

(a) The average noise ratio of each electrode position

Position	*across*	*updown*	*diag*	*wide*
noise ratio	**15.87%**	43.33 %	43.30 %	38.56 %

(b) The average noise ratio from each upper cut-off frequency

Upper cut-off freq.	20 Hz	40 Hz	60 Hz	80 Hz	100 Hz
noise ratio	14.01 %	**13.79%**	13.91 %	14.13 %	15.87 %

As before, 10 measurements were taken to calculate the average values. The results are tabulated in Table 1b. We can see that the noise ratio is at its lowest values when the upper cut-off frequency of the Butterworth filter is 40 Hz. Therefore, the optimal bandwidth of the Butterworth filter is determined as $1 \sim 40$ Hz.

3 Feature Extraction

Feature Extraction is the process of calculating numeric attributes that describe the incoming signal, essentially creating a numeric signature of our input. This signature can be used to uniquely identify a left, right, still or blink using machine learning (ML) Algorithms.

Two feature extraction strategies were considered: The CAnonical Time-series CHaracteristics (catch22) Python package [8], and the tsfeatures package for R [6].

The evaluation metric was to visually compare the 2 Principle Components of features generated by catch22 and tsfeatures. The better strategy provides a stronger separation between "Left" and "Right" movements. In Figure 2 below, the image on the left is the 2 component PCA for catch22, while the image on the right is the 2 component PCA for tsfeatures [10].

It appears that catch22 provides a greater degree of separation. This improves the chances of our classifier to correctly identify left and right eye movements.

After deciding on the feature extraction strategy, the next step was to diminish these features while achieving the same performance. ElasticNetCV was initially implemented to the catch22 features in order to select the best performing features [9].However, we found that ElasticNet reduces the accuracy of the classifiers by 5.66% on average, while increase run times of our algorithms on average.

4 Machine Learning Algorithm

This section investigates a "Good" ML algorithm that achieves a balance between strong accuracy while running in minimal time. The ML classification models below were sourced from SKLearn's suite of ML algorithms [15]. The time measurements were made on a Windows 10 Machine, running on a 7^{th} Gen Intel i5.

The accuracy for each model was determined through Cross-Fold validation on the training set, whilst simultaneously determining the speed of the classifier by setting a timer for the prediction phase of the classifiers [11].

The models our team investigated were the following:

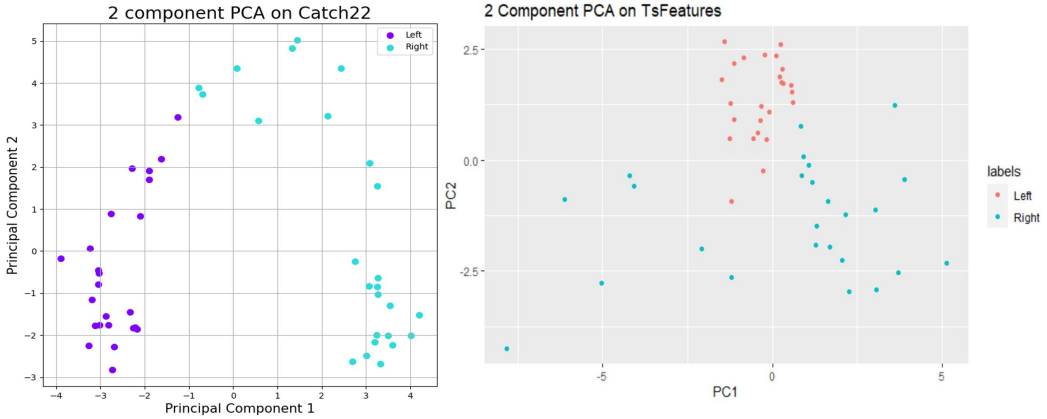


Figure 2: 2-Component PCA for catch22 and tsfeatures

Table 2: Description, cross-validation accuracy and runtime for a dataset of 91 entries.

Model	**Description**	**Accuracy**	**Time**
Heuristic	Our heuristic locates the maximum peak in the wave's signal and its minimum trough. If the former is at an earlier time than the latter, the signal is classified as a 'Left' eye movement, and it is otherwise classified and a 'Right'.	49.02%	0.001s
K-Nearest Neighbours (KNN)	KNN is a supervised ML strategy used for classification. It classifies a new data point by a plurality vote of its k nearest neighbours.[5]	75.66%	0.013s
Logistic Regression	This is an ML strategy which uses a logistic function to model binary dependent variables.[17]	80.02%	0.001s
Naive Bayes (GNB)	GNB is derived from the Bayes' Theorem, used when predictors are considered as independent, i.e. a feature's existence in one class is uncorrelated to its existence in another.[16]	78.98%	0.003s
Random Forest (RF)	This is an ML classifier that essentially creates a collection of decision trees in order to best predict testing data. It uses the outcome most frequented as the classification for its test set [2].	81.16%	0.133s
Support Vector Machine (SVM)	SVM is a supervised ML model used for classification. It works by finding the best hyper-plane between two classes, i.e. one around which data points in both classes are furthest away from each other.[4] In our project, these classes were left and right eye movements.	64.68%	0.003s

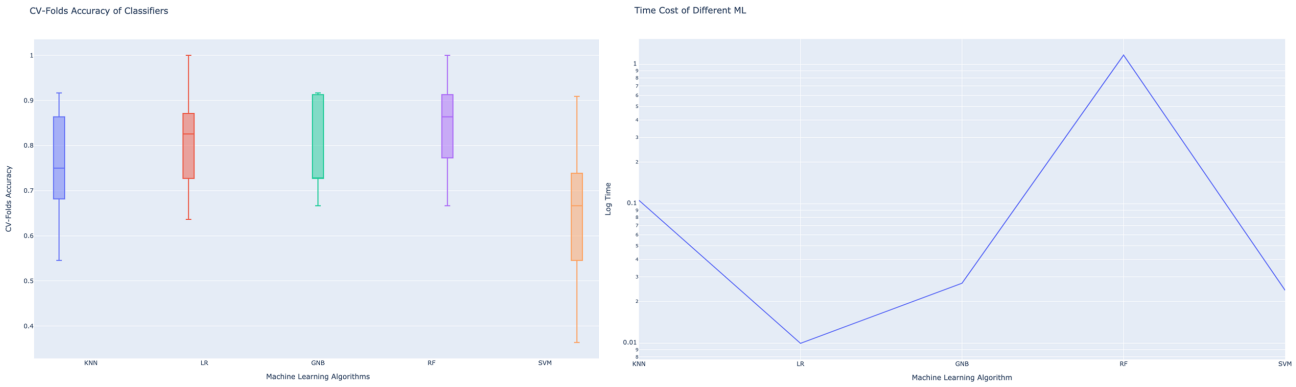


Figure 3: Comparison of ML Classifiers

Investigating the speed and accuracy of ML classifiers

A key factor in our evaluation process for selecting a classifier was the trade-off between accuracy and speed. We established our acceptance criteria for the classifier to have at least 80% accuracy, with under 0.5 seconds for predictability. From the illustration of ML comparisons in Figure 3 above, we can see RF to have the best accuracy, but the slowest speed [11]; whilst the Heuristic performed the fastest, but with worst accuracy.[13] We could potentially turn towards GNB, which performs faster than RF; however, its accuracy does not satisfy our criteria despite being second fastest. We believe the nature of the Snake game is such that accuracy would be valued more than speed. If the game is unable to classify the correct eye movement, the playability of our project decreases as it would be extremely difficult for the player to ever achieve their goal.

Therefore, with the primary criteria being controlling the Snake exactly as desired, RF was chosen as we were willing to reduce the speed of the game for the sake of accuracy. It is also worth noting that the IQR of RF is 17.05%, indicating that 50% of its values are found in this range of accuracy. This is another factor supporting RF as a narrower IQR indicates better accuracy.

The next step was to perform hyperparameter tuning on our RF classifier. RF has several default settings that can be tuned by trial and error. Accuracy of predictions under different combination of these settings are analysed, yielding a model that performs the best for the given data. Risk of over-fitting is avoided by using Cross-Fold Validation, specifically the GridSearch algorithm [15]. Our analysis [12] concluded that $n_{\text{estimators}} = 100$ gives the best performance of the classifier.

5 Accuracy under Streaming Conditions

The previous section captures performance under cross fold validation; however, this is a theoretical accuracy value, not performed under streaming conditions.

Therefore, the following five experiments explore the precision of the classifier when connected to the SpikerBox during real-time streaming conditions. Elements in the "Actual" row represent the ground truth, while the "Pred" row captures the movement by the snake.

Symbols:

L: Left R: Right S: Still

Experiment 1:

Actual	L	R	S	L	L	R	R	S	L	R
Pred	L	S	L	L	R	R	R	S	L	R

Experiment 2 :

Actual	L	L	S	L	L	R	R	S	R	R
Pred	L	S	S	L	L	R	R	S	R	L

Experiment 3:

Actual	L	R	L	R	S	R	R	L	S	L
Pred	L	R	S	L	S	R	R	L	S	L

Experiment 4:

Actual	R	L	L	S	L	S	L	R	R	S
Pred	R	L	L	R	L	L	L	R	R	S

Experiment 5:

Actual	R	S	R	S	L	L	R	R	L	L
Pred	R	S	R	L	L	R	R	R	S	L

The results of these experiments are analysed below.

Table 3: Confusion Matrix Summary

	Predictive Left	Predictive Right	Predictive Still	***Classification Overall***	**User's Accuracy (Precision)**
Actual Left	15	2	3	***20***	**75%**
Actual Right	2	16	1	***19***	**84.211%**
Actual Still	3	1	7	***11***	**63.636%**
Truth Overall	***20***	***19***	***11***	***50***	
Producer's Overall	**75%**	**84.211%**	**63.636%**		

A 3 x 3 confusion matrix is used to assess the accuracy of our classifier. The precision for left, right and still are 75%, 84.2%, 63.6% respectively (Appendix 2), indicating that 75% of the classifier's "Left" predictions were correct. Hence, considering the precision and sensitivity, the classifier is good for predicting left, right, and still.

The classifier's mean prediction accuracy is 74.28%, while the CV accuracy is 81.16%. It is important to consider that different streaming conditions may introduce more noise, and confounding factors such as varying calibration between different Spikerbox equipment. Therefore, we believe 74.28% to be reasonably close to our expected accuracy.

6 Results

To implement OA, we used a SpikerBox Heart and Brain, with electrodes attached near the eyes to detect eye-movements. The voltage difference signal between the two electrodes was 'streamed in' by the SpikerBox – voltage amplitude measurements were recorded in two second windows, then fed into the OA program. The SpikerBox recorded data at a sample rate of $10\,000\,\mathrm{bit\,s^{-1}}$, which was streamed in an encoded format in which each bit of data was represented by two bits, corresponding to a bitrate of $20\,000\,\mathrm{bit\,s^{-1}}$. A statistical classification algorithm based on ML was used to map decoded voltage amplitude measurements to eye-movement classifications, based on a set of features extracted from the voltage waveform. The four eye-movement classifications were 'Left', 'Right', for eye movements of the eyes to either side, then immediately back to centre-of-vision; 'Blink' for blinks, and 'Still' for movements that are none of the above. From the results of these classifications, the snake turns left, right or keeps moving in its current direction. The 'Blink' classification is not to implement a feature to the game, rather to stop waveforms created by blinks being misclassified as 'Left' or 'Right'. These classifications are relayed to a modified version of the classic Snake game, modified from the freegames Python package [3] and allow the user to control the snake.

The two electrodes on either side of the right eye, 2 cm above each end of the eyebrow. Another ground electrode is placed behind the right ear, in contact with the Mastoiditis, the skull bone behind the ear. This yielded the best signals in terms of the ability to distinguish between the four eye-movement classifications above, as this electrode position has the lowest noise-to-signal ratio out of all four positions at 15.87%.

The signals are filtered by a Butterworth and Notch filter. The Butterworth filter is a band-pass filter with bandwidth $1 \sim 40\,\mathrm{Hz}$, where the upper cut-off frequency was determined experimentally to produce the cleanest outputs. The Notch filter is a band-stop filter with a narrow bandwidth centred at 50 Hz. It is to attenuate 50 Hz signals interference from AC power supply.

A 'treadmill' algorithm was implemented to prevent eye-movements going undetected or being incorrectly classified when over the boundary of two consecutive windows. For example, a true 'Left' movement may be classified as 'Still' due to being split over two windows. In this case, the classification algorithm must be performed on the two most recent windows, ensuring no relevant portion of the eye-movement signal is excluded due to truncation. Whilst this introduces a small time lag between the user's eye-movement and the movement of the snake in the game, potentially detracting from the user experience, it results in no eye-movements being 'missed' by the classifier, hence reducing user frustration as a whole.

To extract features from the voltage waveform over the two-second interval, we used the CAnonical Time-series Characteristics (catch22) Python package. This gave us a set of 22 numerical properties of the time-series. Using Principle Component Analysis in Figure 2, catch22 was determined to provide a good degree of separation between Left and Right movements, aiding in a more accurate classification of the two. This decision was backed up by a numeric objective function. Given two clusters A and B, this anti-clustering objective function calculates the average Euclidean distance between each point in a cluster A to the centroid of cluster B. This function was applied to clusters of 'Left' and 'Right' data inside the projection of each feature space onto the subspace spanned by its first and second principal components. Tsfeatures had an anti-clustering objective of 116.0 while catch22 achieved 278.9 [14]. This indicated a greater separation between the two classifications in the feature space of catch22 than tsfeatures, making it our package of choice.

Feature selection using ElasticNetCV [15] was initially tried yet ultimately discarded; for most classifiers, classification times were found to increase and accuracy decrease.

To perform the classification, we used the Random Forest algorithm, implemented as part of the Scikit-Learn Python package [15]. Although this algorithm took the longest classification time, it fulfilled the more important requirement of being the most accurate, with over 80% accuracy under cross-validation tests. Hyper-parameter tuning was performed on the number of estimators used in the random forest algorithm, with scikit-learn's GridSearchCV method [15].

To train the model and ensure reliability, we collected of eye-movements made by all members of the team, ensuring that a variety of training data trains the classifier, preventing over-fitting of the data to a single individual.

The project was highly multidisciplinary in nature – the interaction between physics and data science in this project can be described as a feedback loop – processes from both fields strongly interacted with, and influenced each other. All decisions were driven by physical intuition and reasoning, but backed up by data-driven analysis. Choosing the correct filter, the appropriate feature extraction strategy and the correct classifier further extended into sciences of game-making, software development, and quality assurance. Finally, Appendix 9.2 further provides an overview of the complexity of the final design.

7 Discussion

The team was able to implement a successful product which satisfied our goals, however, there were limitations present in the approaches undertaken to do so.

One such limitation was the buffer of two seconds selected. However, in order to make the game compatible with this, it had to be slowed down to prevent the user making multiple eye movements within two seconds, and having commands being missed. This reduces the user's experience as playability is decreased due to the lag in the game.

Another limitation was the controls chosen for the game. As the game is being played from the Snake's perspective, only left and right eye movements were selected. However, with the game having a vertical 2D display, this may not be clear to the user, who may try to control the Snake with up and down eye movements as well. Therefore, future projects could include a variety of intuitive controls to improve the game's playability.

A significant limitation present was also the detection of the selected eye movement's signals. Through PCA, we discovered that lefts and rights were often being grouped as the same class. Additionally, left wave signals looked very similar to blinks in their raw data, potentially causing some classifications. Hence, the classification could have been improved by selecting eye movements for which the signal drastically varies from others and is distinct from noise. One such movement could have been using a left and right wink to control the Snake instead as their signals have significantly high peaks. However, whilst this improves the accuracy, and potentially the playability of the game, it could still reduce user experience as some may struggle to perform a wink.

Furthermore, the initial collection and storing of the wave data also had a drawback. After initially finding our heuristic classifier to have poor accuracy, a premature decision was made to delve solely into ML classifiers. Therefore, during data collection, the wave signal was immediately converted and stored as a list of catch22 features instead of its raw signal. Whilst this made handling the data during the training process easier, the lack of raw data delayed our progress further down. As we further explored various detection strategies, we realised we were unable to test methods such as variance of the waves or zero crossings. This caused setbacks as data had to then be recollected to fit our needs.

8 Student Contribution

Abhay was responsible for evaluating the feature extraction strategy and tuning the final ML model. Ruoyang(Michael) was responsible for data visualizations, selecting classifiers, and confusion matrix part. Xinchen(Wayne) was responsible for coding classifier, analyze and visualization of ML strategies' comparison(including accuracy and speed, IQR, and etc.), and confusion matrix part. Freya was responsible for feature selection, testing times for classifiers, and executive summary and discussion of the report. Damian was responsible for coding the streaming interface for the SpikerBox (including the treadmill heuristic), coding a data-collection interface for creating training data, coding a polarity-checking tool, demonstrating the project, and primarily responsible for the results section of the report. He also contributed to coding the signal filtering and the signal filtering analysis. Karl was responsible for finding the optimal electrode positions, designing the signal filtering, and developing the interface with the Snake game.

References

[1] Anwesha Banerjee et al. "Classifying Electrooculogram to Detect Directional Eye Movements". In: *Procedia Technology* 10 (2013), pp. 67–75. DOI: 10.1016/j.protcy.2013.12.338.

[2] Niklas Donges. *A Complete Guide To The Random Forest Algorithm*. June 2019. URL: https://builtin.com/data-science/random-forest-algorithm.

[3] *free-python-games on Github*. URL: https://github.com/grantjenks/free-python-games.

[4] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. June 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[5] Onel Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Sept. 2018. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[6] Rob Hyndman et al. *tsfeatures: Time Series Feature Extraction*. R package version 1.0.2. 2020. URL: https://CRAN.R-project.org/package=tsfeatures.

[7] Alberto López et al. "High-Performance Analog Front-End (AFE) for EOG Systems". In: *Electronics* 9.6 (2020), p. 970. DOI: 10.3390/electronics9060970.

[8] Carl H. Lubba et al. "catch22: CAnonical Time-series CHaracteristics". In: *Data Mining and Knowledge Discovery* 33.6 (Nov. 2019), pp. 1821–1852. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00647-x. URL: https://doi.org/10.1007/s10618-019-00647-x.

[9] Gianluca Malato. *Feature selection in machine learning using Lasso regression*. URL: https://towardsdatascience.com/feature-selection-in-machine-learning-using-lasso-regression-7809c7c2771a.

[10] *OcculoAnnelid on Github. Branch 'master'. 'src' folder*. URL: https://github.sydney.edu.au/dlin6906/OcculoAnnelid/tree/master.

[11] *OcculoAnnelid on Github. Branch 'ml-algo-result. accuracy.py'*. URL: https://github.sydney.edu.au/dlin6906/OcculoAnnelid/tree/ml-algo-result.

[12] *OcculoAnnelid on Github. Branch 'ml-algo-result. tuningRF.py'*. URL: https://github.sydney.edu.au/dlin6906/OcculoAnnelid/tree/ml-algo-result.

[13] *OcculoAnnelid on Github. Branch heuristic-tester. heuristic$_t$ester.py'*.. URL: https://github.sydney.edu.au/dlin6906/OcculoAnnelid/tree/heuristic-tester.

[14] *OcculoAnnelid on Github. Branch master. src/pca.py'*. URL: https://github.sydney.edu.au/dlin6906/OcculoAnnelid/tree/master.

[15] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

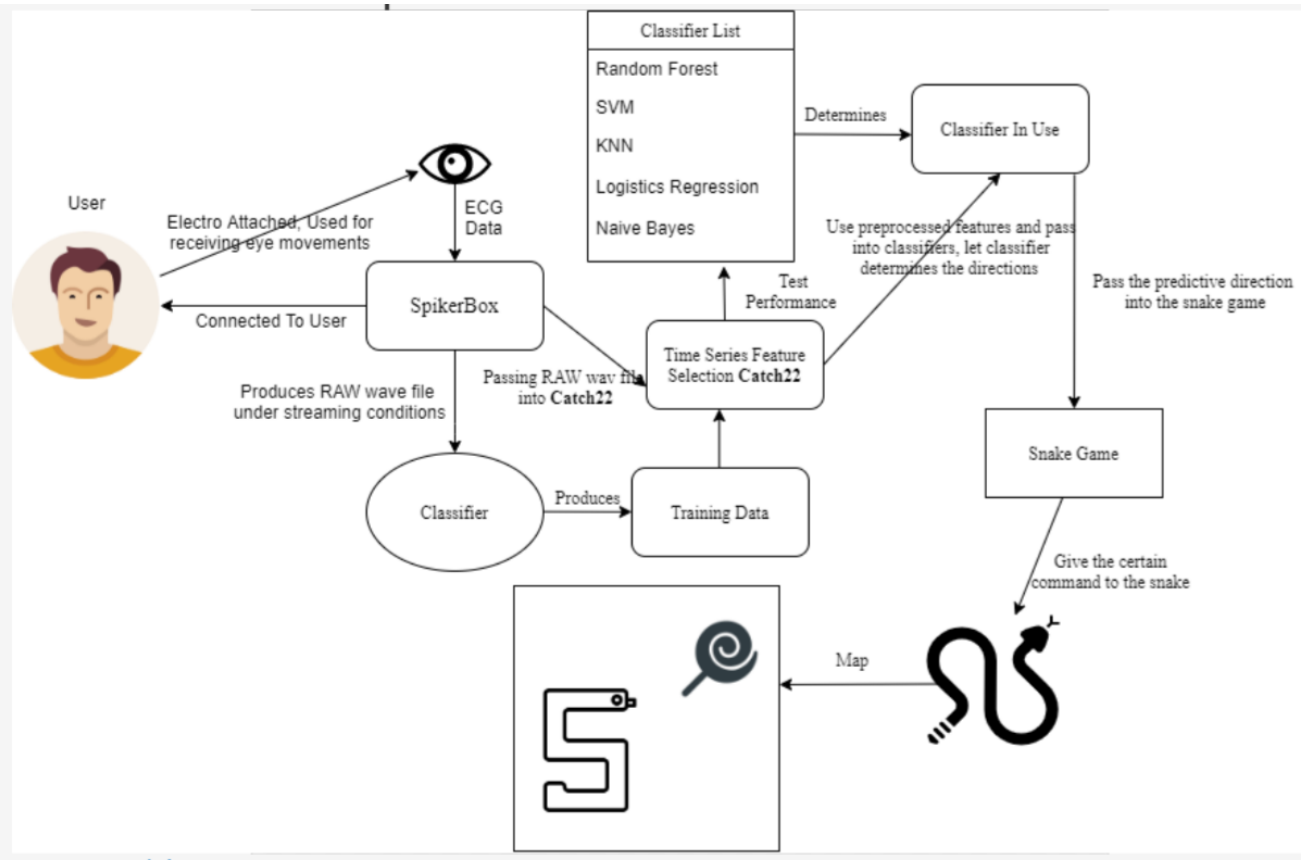
[16] Sunil Ray. *6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R*. Sept. 2017. URL: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[17] Saishruthi Swaminathan. *Logistic Regression — Detailed Overview*. Mar. 2018. URL: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

9 Appendix

9.1 Multidisciplinary Approach

The diagram below summarises the complexity of the project, outlining the multidisciplinary aspects.



9.2 Further Accuracy Analysis of Streaming Classifier

The table below summarises sensitivity, Specificity and Positive/Negative Parity of the Random Forest Classifier under streaming conditions.

	Sens	Spec	PPV	NPV
A(Left)	0.75	0.81	0.75	0.83
B(Right)	0.11	0.42	0.11	0.90
C(Still)	0.27	0.71	0.27	0.90