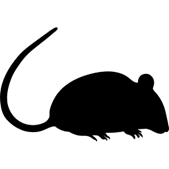
Machine Learning Approach to Identifying Neural Features That Predict Rodent Behavior

**Solution Approach**

Washington State University - Department of Psychology



Nunnerson Computing



Aidan Nunn, Charlie Nickerson

10/5/22

**TABLE OF CONTENTS**

**I. Introduction**

The introduction begins by stating the purpose of the document. Explain the purpose for

providing this design document and specify the intended audience for it. If this is a revision of

an earlier document, please make sure to summarize what changes have been made during the

revision (keep this discussion brief). Then provide a brief description of your project and state

your project goal.

In the document, the Nunnerson Computing team will present the solution approach for building a program that can predict when rodents addicted to alcohol are in the decision making process of consuming alcohol using the lasso machine learning model. The purpose of writing this is to help future researchers understand the basic architecture and structure of our code.

We will first go over the architecture and design of the program and give a top-level design view of the software architecture and describe each component and its responsibility. In the next section we’ll provide the subsystems of our program and describe the rationale for the decomposition of each subsystem. Next we will talk about the type of data structures and databases required for our program. After that we will end the paper by describing the user interface design.

**II. System Overview**

The system overview contains a general description of the functionality and design of the

project. The overview will only briefly describe the overall design considerations and the

comprehensive explanations will be done in the sections to follow. The overview should serve

as an introduction to these sections.

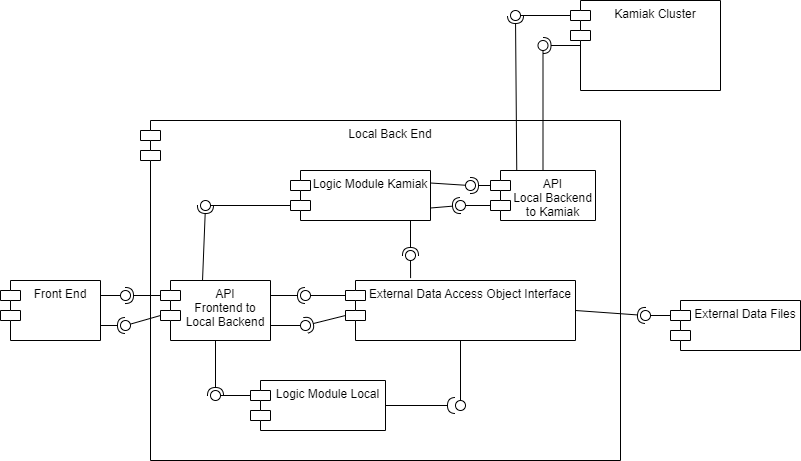
This program will take in data which contains the LFP (local field potential) data collected from the rodents brain and preprocess the data. In order to preprocess the data, it is broken up into 5 second periods. Using the data found within these 5 second periods, the cohesive and power values are calculated. After these values are calculated, we then move onto the machine learning step. Using a machine learning model the user will pass in the preprocessed data. During this step, the program should output the 5 second periods of data where the rodents were in the decision making process of drinking alcohol. The user should be able to take this outputted data and download it to their computer as a csv file.

**III. Architecture Design**

**III.1. Overview**

Nunnerson Computing has decided on a Client-Server model for this project. We considered using a Model-View-Controller architecture, but since there isn’t much need to store data we opted to stick with the Client-Server model instead. The Client-Server model also more accurately reflects the relationship any local program would have with Kamiak, since we want a desktop application that sends requests to the Kamiak Cluster to run jobs and return the results to the user. We will design a frontend desktop application with a GUI for the researchers to use. Previously they have been using a terminal to use the code, so we want to make the machine learning process more accessible to the researchers. The desktop app will also have a local backend that stores the logic for the program and acts as an intermediary between the frontend and Kamiak Cluster. This local backend will also run less intensive computations that we don’t need Kamiak for.

The entry point for the user is the Front End, which is the Client half of the Client-Server model. The Front End contains the GUI interface where the user will input commands and view results of their requests. We have split the Server part of the Client-Server model into two parts: the Local Back End and the Kamiak Cluster. We don’t want to heavily couple the GUI and logic components, so the Local Back End interprets requests from the Front End and sends job protocols to Kamiak. The Front End communicates with the Local Back End via the API Frontend to Local Backend module (API F/B). The API F/B branches to the External Data Access Object Interface, which preprocesses data from the files inputted by the user. Then, depending on the job requested by the user, we move to either the Logic Module Local or the Logic Module Kamiak. The Logic Module Local stores algorithms for less computationally intensive processes, like testing a machine learning model against sample data or using the model to predict rodent behaviors. If we discover that this is too intensive for a local machine, we will remove this module and instead have all jobs run on Kamiak. If an intensive job is requested, like building a new machine learning model, Logic Module Kamiak uses the API Local Backend to Kamiak module to interface with Kamiak and send a job request. Kamiak will run the job and return the results.



**III.2. Subsystem Decomposition**

**I.1.1. API Frontend to Local Backend**

1. **Description**

The API Frontend to Local Backend (API F/B) is the connection between our frontend GUI and the backend of our software. It handles job requests from the frontend and uses the appropriate logic module to perform those jobs, as well as returning the results of those jobs to the frontend.

1. **Concepts and Algorithms Generated**
2. **Interface Description**

Services Provided:

Services Required:

**I.1.2. External Data Access Object Interface**

1. **Description**

The External Data Access Object Interface preprocesses data from inputted user data files for use in the logic modules.

1. **Concepts and Algorithms Generated**

The External Data Access Object Interface

1. **Interface Description**

Services Provided:

Services Required:

**I.1.3. Logic Module Local**

1. **Description**

The Logic Module Local holds algorithms for computations that can be run locally. Since it takes time to reserve Kamiak’s computing time, we intend on using the local backend to run smaller jobs that don’t need Kamiak, instead of relying on Kamiak for all computations. Local computations include testing the accuracy of a machine learning model against sample data and using the model to make predictions on input data. If these jobs are too intensive to be run locally, we will remove the Logic Module Local and allow Kamiak to perform all computations instead.

1. **Concepts and Algorithms Generated**
2. **Interface Description**

Services Provided:

Services Required:

**I.1.4. Logic Module Kamiak**

1. **Description**

The Logic Module Kamiak holds protocols and algorithms for jobs that need to be run on the Kamiak Cluster. When the API Frontend to Local Backend asks for a job to be run that needs Kamiak’s computing power, the Logic Module Kamiak will send the appropriate protocol and algorithm to the API Local Backend to Kamiak. We will be basing the protocols used by our software on those invented by the previous capstone team who worked on this project.

1. **Concepts and Algorithms Generated**
2. **Interface Description**

Services Provided:

Services Required:

**I.1.5. API Local Backend to Kamiak**

1. **Description**

The API Local Backend to Kamiak connects our Local Backend to the Kamiak Cluster. It accepts a protocol and algorithm from the Logic Module Kamiak and sends them to the Kamiak Cluster. Once the job is completed on Kamiak, the results will be returned via this API.

1. **Concepts and Algorithms Generated**
2. **Interface Description**

Services Provided:

Services Required:

**IV. Data Design**

[You may skip this section if your project doesn’t require any data manipulation or storage]

Describe all data structures (including the internal and temporary data structures), and the

database(s) created as part of the application. This information is important from the design

point of view as it will help the team in properly understanding all the data structures and

databases which will be required for the coding.

**IV. User Interface Design**

[You may skip this section if your project doesn’t have a GUI component] – but! If the tools is

ever to be used by humans (even just starting and stopping it), there’s some form of user

interface design. It can be very simple, but it does exist. Make sure you document how you

expect people to use your product, even if it’s just:

● Installation

● Configuration file edits

● Launch daemon by running command [x]

Provide a detailed description of user interface. The information in this section should be

accompanied with proper images showing how exactly you vision the interface to be like (for

example mock-ups). Make sure to mention which use cases in your “Requirements

Specification” document will utilize these interfaces for user interaction.

**IV. Glossary**