Course Project Description

Julie Lee

STA 141A Project: Analzying the Brain's Role in Decision-Making through Neuronal Dynamics in Mice

ABSTRACT

This project delves into a subset of data sourced from an experiment conducted by Steinmetz et al., which investigates the neural encoding of vision, choice, action initiation, and behavioral engagement across diverse brain regions in mice. In this report, we perform a comprehensive analysis of recorded data reflecting the activity of neurons in the mice's visual cortex by exploring features of the datasets through visual aids, integrating the data by identifying homogeneity and heterogeneity across sessions and mice, and building a predictive model that accurately forecasts trial outcomes by considering the relationship between different levels of visual stimuli and neural activity information. Further exploration and analysis of this data enhances our understanding of the spatial distribution of neurons underlying behavioral processes, which hopefully will hold clinical value in medicine, neuroscience, and technology in the future.

SECTION 1 INTRODUCTION:

In this report, we will construct the most accurate model that predicts the feedback type (Success or Failure) of each trial depending on the neural activity and stimuli presented in front of 4 different mice: Cori, Frossman, Hence, and Lederberg. The neural activity data is characterized by spike trains of neurons that have been recorded across the brain as the mice perform various tasks during each trial. During each trial, the mice are presented with visual stimuli of various contrast that appears on only the left side, right side, both sides, or neither sides where a "Success" is characterized by whether the mice chooses the correct action, along with choosing to engage in a specific task. These are the following criteria:

When left contrast > right contrast, success (1) if turning the wheel to the right and failure (-1) otherwise. When right contrast > left contrast, success (1) if turning the wheel to the left and failure (-1) otherwise. When both left and right contrasts are zero, success (1) if holding the wheel still and failure (-1) otherwise. When left and right contrasts are equal but non-zero, left or right will be randomly chosen (50%) as the correct choice.

We utilize data from 18 sessions, each holding varying numbers of trials for 4 mice as indicated above. Each trial includes information regarding Feedback Type (1 for success and -1 for failure), contrast_left (contrast of the left stimulus), contrast_right (contrast of the right stimulus), time(centers of the time bins for spikes), spk (number of spikes of neurons in the visual cortex in time bins defined in time), and brain_area (area of the brain where each neuron lives).

To start with our analysis, we will print out the following code to list out the corresponding experiment dates and mouse name of each session (Session 1 to 18):

```
## [1] "Cori"
## [1] "2016-12-14"
## [1] "Cori"
## [1] "2016-12-17"
## [1] "Cori"
## [1] "2016-12-18"
## [1] "Forssmann"
## [1] "2017-11-01"
## [1] "Forssmann"
## [1] "2017-11-02"
## [1] "Forssmann"
## [1] "2017-11-04"
## [1] "Forssmann"
## [1] "2017-11-05"
## [1] "Hench"
## [1] "2017-06-15"
## [1] "Hench"
## [1] "2017-06-16"
## [1] "Hench"
## [1] "2017-06-17"
## [1] "Hench"
## [1] "2017-06-18"
## [1] "Lederberg"
## [1] "2017-12-05"
## [1] "Lederberg"
## [1] "2017-12-06"
## [1] "Lederberg"
## [1] "2017-12-07"
## [1] "Lederberg"
## [1] "2017-12-08"
## [1] "Lederberg"
## [1] "2017-12-09"
## [1] "Lederberg"
## [1] "2017-12-10"
## [1] "Lederberg"
## [1] "2017-12-11"
```

SECTION 2 EXPLORATORY ANALYSIS:

In this section, we will explore the features of the data sets to build prediction model.

(I) Data structures across sessions (number of neurons, number of trials, stimuli conditions, feedback types):

The table below summarizes the Experiment Data, corresponding Mouse Name, Number of Neurons, Number of Trials, unique levels of Stimuli Conditions of both the Left and Right Contrast, and the unique levels of feedback types for EACH session (1-18 sessions). The summary table aids in comprehending the distinct characteristics of each session when viewed collectively. For instance, Session #1 was conducted on 12/14/2016 on Cori, consisted of 114 trials, involved 734 neurons, had varying contrast_left and contrast_right leels at 0,0.25, 0.5, and 1, and had 2 unique feedback types, 1 and -1.

Summary Table

Session_Number	Experiment_Date	Mouse_Name	Number_Neurons	Number_Trials	Unique_Stimuli_Conditions	Unique_Feedback_Types
1	2016-12-14	Cori	734	114	Contrast_Left: 0, 0.5, 1, 0.25 Contrast_Right: 0.5, 0, 1, 0.25	1, -1
2	2016-12-17	Cori	1070	251	Contrast_Left: 1, 0.25, 0.5, 0 Contrast_Right: 1, 0, 0.5, 0.25	-1, 1
3	2016-12-18	Cori	619	228	Contrast_Left: 0.5, 1, 0, 0.25 Contrast_Right: 0, 1, 0.5, 0.25	1, -1
4	2017-11-01	Forssmann	1769	249	Contrast_Left: 0, 0.5, 0.25, 1 Contrast_Right: 1, 0.5, 0.25, 0	1, -1
5	2017-11-02	Forssmann	1077	254	Contrast_Left: 0, 0.25, 0.5, 1 Contrast_Right: 1, 0, 0.25, 0.5	1, -1
6	2017-11-04	Forssmann	1169	290	Contrast_Left: 0.25, 0, 1, 0.5 Contrast_Right: 0, 0.5, 1, 0.25	1, -1

Session_Number	Experiment_Date	Mouse_Name	Number_Neurons	Number_Trials	Unique_Stimuli_Conditions	Unique_Feedback_Types
7	2017-11-05	Forssmann	584	252	Contrast_Left: 1, 0, 0.25, 0.5 Contrast_Right: 0.5, 0, 1, 0.25	1, -1
8	2017-06-15	Hench	1157	250	Contrast_Left: 0, 1, 0.5, 0.25 Contrast_Right: 0.5, 0, 1, 0.25	1, -1
9	2017-06-16	Hench	788	372	Contrast_Left: 1, 0.5, 0, 0.25 Contrast_Right: 0.25, 0, 0.5, 1	1, -1
10	2017-06-17	Hench	1172	447	Contrast_Left: 0, 0.5, 1, 0.25 Contrast_Right: 1, 0.5, 0, 0.25	-1, 1
11	2017-06-18	Hench	857	342	Contrast_Left: 0.5, 0, 1, 0.25 Contrast_Right: 0.5, 1, 0, 0.25	1, -1
12	2017-12-05	Lederberg	698	340	Contrast_Left: 0, 1, 0.25, 0.5 Contrast_Right: 0, 0.5, 1, 0.25	1, -1
13	2017-12-06	Lederberg	983	300	Contrast_Left: 0, 0.5, 0.25, 1 Contrast_Right: 0, 1, 0.25, 0.5	-1, 1
14	2017-12-07	Lederberg	756	268	Contrast_Left: 1, 0, 0.25, 0.5 Contrast_Right: 0, 0.25, 1, 0.5	1, -1
15	2017-12-08	Lederberg	743	404	Contrast_Left: 0.5, 0, 0.25, 1 Contrast_Right: 0, 1, 0.25, 0.5	1, -1
16	2017-12-09	Lederberg	474	280	Contrast_Left: 1, 0, 0.5, 0.25 Contrast_Right: 0.25, 0, 1, 0.5	1, -1
17	2017-12-10	Lederberg	565	224	Contrast_Left: 0, 0.5, 0.25, 1 Contrast_Right: 0, 1, 0.5, 0.25	-1, 1
18	2017-12-11	Lederberg	1090	216	Contrast_Left: 0.5, 1, 0, 0.25 Contrast_Right: 0, 0.25, 0.5, 1	1, -1

The summary table below individually lists ALL TRIALS from sessions 1-18 and includes information corresponding to the levels of contrast_right and contrast_left, Feedback type (Success or Failure), and the number of time bins within each specific trial. It also provides session-specific information such as the number of unique brain areas and the number of neurons. This table was created to aid in constructing visualizations that showcase patterns and trends of different variables throughout every trial of a chosen session later in the report. Only the first and last 10 observations from this summary table are presented from the complete table, which consists of 5081 rows (total number of trials across all 18 sessions) and 10 columns (different variables characteristic of trials and sessions).

Combined Data for Each Session

Session_Number	Trial_Number	Mouse_Name	Date_Exp	Contrast_Left	Contrast_Right	Feedback_Type	Number_Neurons	Number_Time_Bins	Un
1	1	Cori	2016-12-14	0.0	0.50	Success	734	40	
1	2	Cori	2016-12-14	0.0	0.00	Success	734	40	
1	3	Cori	2016-12-14	0.5	1.00	Failure	734	40	
1	4	Cori	2016-12-14	0.0	0.00	Failure	734	40	
1	5	Cori	2016-12-14	0.0	0.00	Failure	734	40	

Session_Number	Trial_Number	Mouse_Name	Date_Exp	Contrast_Left	Contrast_Right	Feedback_Type	Number_Neurons	Number_Time_Bins	Un
1	6	Cori	2016-12-14	0.0	0.00	Success	734	40	
1	7	Cori	2016-12-14	1.0	0.50	Success	734	40	
1	8	Cori	2016-12-14	0.5	0.00	Success	734	40	
1	9	Cori	2016-12-14	0.0	0.00	Success	734	40	
1	10	Cori	2016-12-14	0.5	0.25	Success	734	40	

(II) Investigating Neural Activity

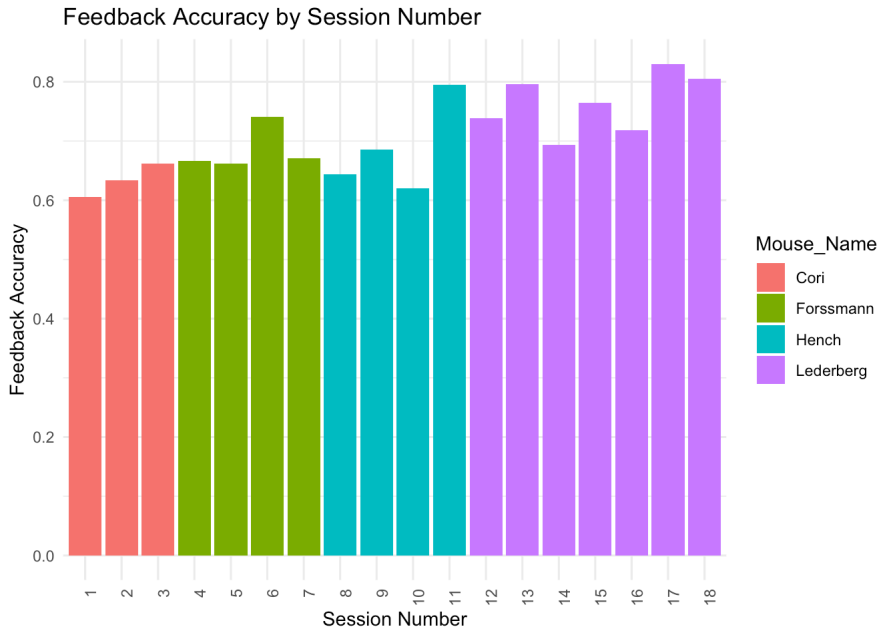
To assist in generating visualizations of neural activities for each trial of a particular session, I created data sets by constructing a function that iterates through every trial of a specific session (from session 1-18). This function also iterates through every neuron of each specific trial, displaying the corresponding neuron number, affiliated brain area, and a record of the number of spikes within each of the 40 time bins. I will employ this function to specify a session number and enhance the visualization of its neural activities across all its trials later in the project report. For example purposes, we call the function to obtain the data set for Session #17 which consists of 126,560 rows that correspond to each neuron of each trial within the session. After calling the function: load_session_data(17), we print the first 6 observations of the data set.

Top 5 Observations of Neural Activity for Session 17

	Trial	Neuron_Number	brain_area	#Spikes of Neurons in Time Bin 1	#Spikes of Neurons in Time Bin 2	#Spikes of Neurons in Time Bin 3	#Spikes of Neurons in Time Bin 4	#Spikes of Neurons in Time Bin 5	#Spikes of Neurons in Time Bin 6	#Spikes of Neurons in Time Bin 7	#Spikes of Neurons in Time Bin 8	#Spikes of Neurons in Time Bin 9	#Spik of Neuro in Tir Bin
Trial_ 1 *Neuron* 1	1	1	root	0	0	0	0	0	0	0	0	0	0
Trial_ 1 *Neuron* 2	1	2	root	0	0	0	0	0	0	0	0	0	0
Trial_ 1 *Neuron* 3	1	3	VPL	0	0	0	0	0	1	0	0	0	0
Trial_ 1 *Neuron* 4	1	4	root	0	0	0	0	0	0	0	0	0	0
Trial_ 1 *Neuron* 5	1	5	root	0	0	0	0	0	0	0	0	0	0

Feedback Accuracy Across Sessions and Mice

We begin by comparing the Feedback Type Accuracy across two dimensions: 1. all 18 sessions, and 2. across all 4 mice. This approach provides a better gauge on which session number or specific mouse warrants deeper study when examining neural activity. The session number or mouse exhibiting the highest Feedback Accuracy will serve as a basis for constructing our prediction model later in the report. Displayed below is a bar plot depicting each session (1-18) on the x-axis and Feedback Accuracy (proportion of successful feedback when feedback_type == 1) on the y-axis. Each bar, representing a specific session number, is color-coded based on the corresponding mouse name. Notably, Session #17 exhibits the highest Feedback Accuracy, while Session #1 shows the lowest. Additionally, Session #17 corresponds to the mouse named “Lederberg”. This information will guide our further investigation into neural activity and aid in constructing our prediction model in later analyses.



Investigating Neural Activities throughout each trial

Now that we have obtained the specific session and mouse name that yields the highest feedback accuracy, we perform deeper investigation and analysis of the neural activities existent within Session #17 (Lederberg).

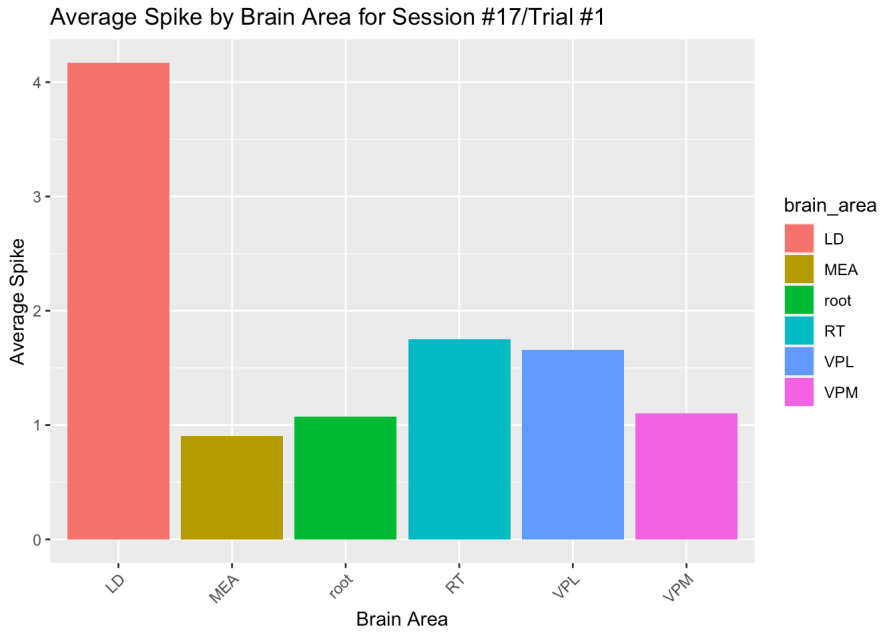
1. We first construct a table that lists out all the unique brain areas for each session (1-18). The total number of unique brain areas across all 18 sessions is: 62 unique brain areas. We see that Session #17 has 6 associated unique brain areas: root, VPL, VPM, RT, MEA, and LD.

Unique Brain Areas in Each Session

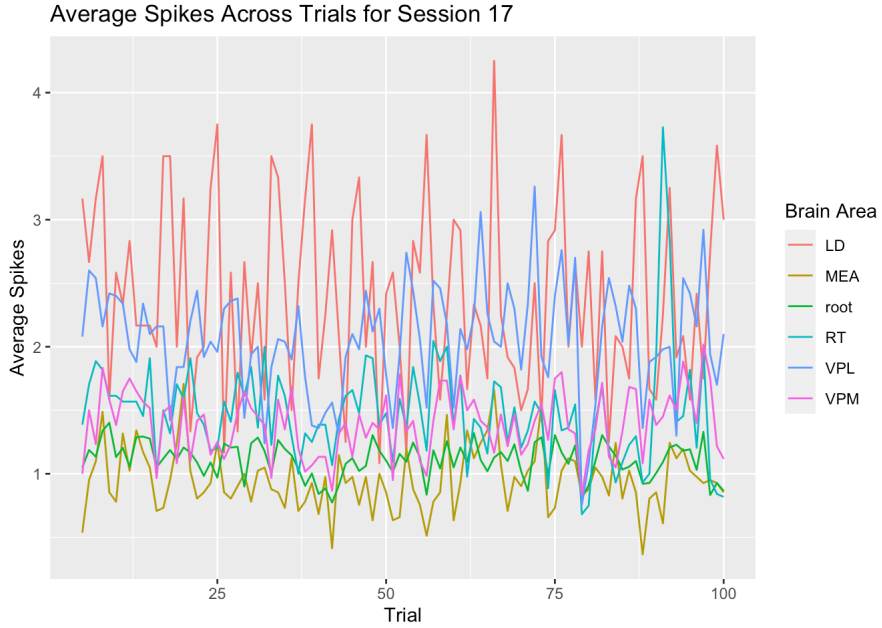
Session_Number	Unique_Brain_Areas	Total_Unique_Brain_Areas
1	ACA, MOs, LS, root, VISp, CA3, SUB, DG	8
2	CA1, VISl, root, VISpm, POST	5
3	DG, VISam, MG, CA1, SPF, root, LP, MRN, POST, NB, VISp	11
4	LGd, DG, TH, SUB, VPL, VISp, CA1, VISa, LSr, ACA, MOs	11
5	VISa, root, CA1, SUB, DG, OLF, ORB, ACA, PL, MOs	10
6	AUD, root, SSp, CA1, TH	5
7	VPL, root, CA3, LD, CP, EPd, SSp, PIR	8
8	ILA, TT, MOs, PL, LSr, root, LD, PO, CA3, VISa, CA1, LP, DG, VISp, SUB	15
9	TT, ORBm, PL, LSr, root, CA3, VISl, CA1, TH, VISam, VPL, LD	12
10	MB, VISp, SCm, SCsg, POST, DG, MRN, CA1, VISl, POL, root, GPe, VISrl	13
11	MOp, LSc, root, PT, CP, LSr	6
12	VISp, DG, SUB, LGd, PL, root, MOs, ACA, CA1, VISam, MD, LH	12
13	VISam, ZI, DG, CA1, LGd, MB, SCs, RN, MRN, SCm, ACA, PL, MS, root, MOs	15
14	ORB, MOs, root, MRN, SCm, SCs, VISp, RSP, CA1, PAG	10
15	BLA, GPe, root, VPM, LGd, ZI, MB, CA3	8
16	SSs, SSp, MB, TH, LGd, CA3	6
17	root, VPL, VPM, RT, MEA, LD	6
18	CP, ACB, OT, SI, SNr, LGd, ZI, CA3, root, TH	10

2. Secondly, we delve deeper into the spike information pertaining to a specific trial within Session #17. In this analysis, we focus on the average spike count, which is calculated by summing the total number of spike counts across all 40 time bins and dividing by the total number of neurons. Specifically, we examine the first trial of Session #17, grouping the spike counts by the respective brain areas. Notably, we observe a notable spike count average for the brain area LD, suggesting heightened neural activity within this region of mice when exposed to various stimuli, in contrast to other brain areas. This observation may imply a strong correlation between the LD area and neural activity during decision-making or action execution. However, since this observation is derived from a single trial within the session, drawing definitive conclusions regarding the significance of LD's activity during neural events is challenging. To address this, we extend our analysis to encompass the average spike count across all trials within Session #17 in the next part of our report.

```
##
##
## |brain_area | Total_Spikes| Total_Brain_Areas| Average_Spikes|
## |:----------|------------:|-----------------:|--------------:|
## |LD         |           50|                12|       4.166667|
## |MEA        |           37|                41|       0.902439|
## |RT         |           77|                44|       1.750000|
## |VPL        |           83|                50|       1.660000|
## |VPM        |           66|                60|       1.100000|
## |root       |          384|               358|       1.072626|
```



3. Thirdly, we proceed to visualize the average spike count, computed as the total number of spikes divided by the number of brain areas, across ALL trials of Session #17 grouped by brain_area. To enhance clarity, we start the graph at the 5th trial and conclude at the 100th trial, facilitating the identification of patterns. Notably, throughout the majority of trials, the average spike count exhibits its highest values for the Red line, corresponding to the brain area “LD”. This observation supports our earlier findings, where LD demonstrated the highest average spike count during the first trial of Session #17. Therefore, we can say that there is a strong and robust correlation between the LD brain area and neural activity during decision-making or action execution for the mouse Lederberg in Session #17.

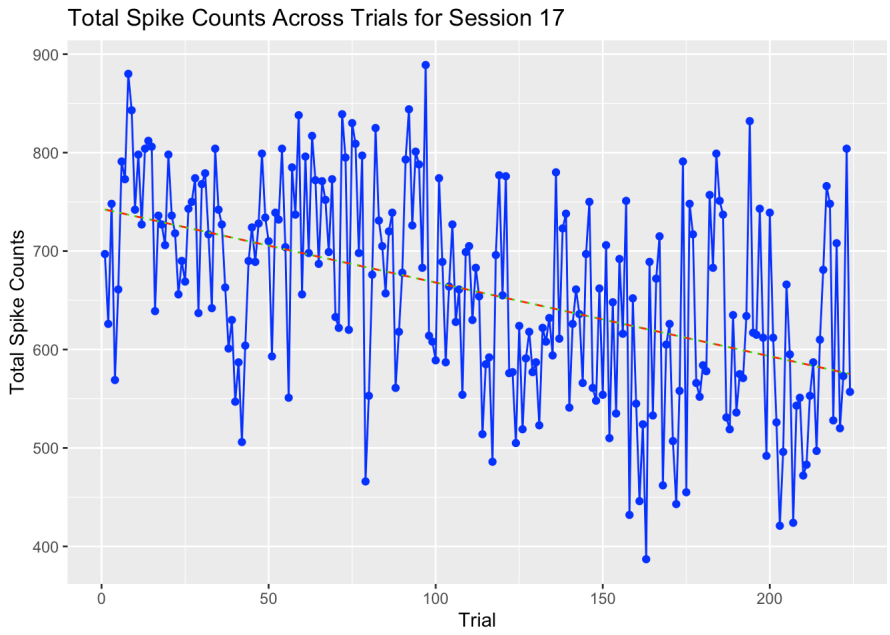


(III) Exploring Spk Changes

Total Number of Spikes across Trials

In our investigation of spike activity across all trials of Session #17, the plot below illustrates the total number of spike counts on the y-axis against each trial on the x-axis. A red trend line represents the trend as the number of trials increases. This trend line indicates a noticeable decrease in the total number of spike counts with an increasing trial number for Session #17. This observation suggests that the mice involved in

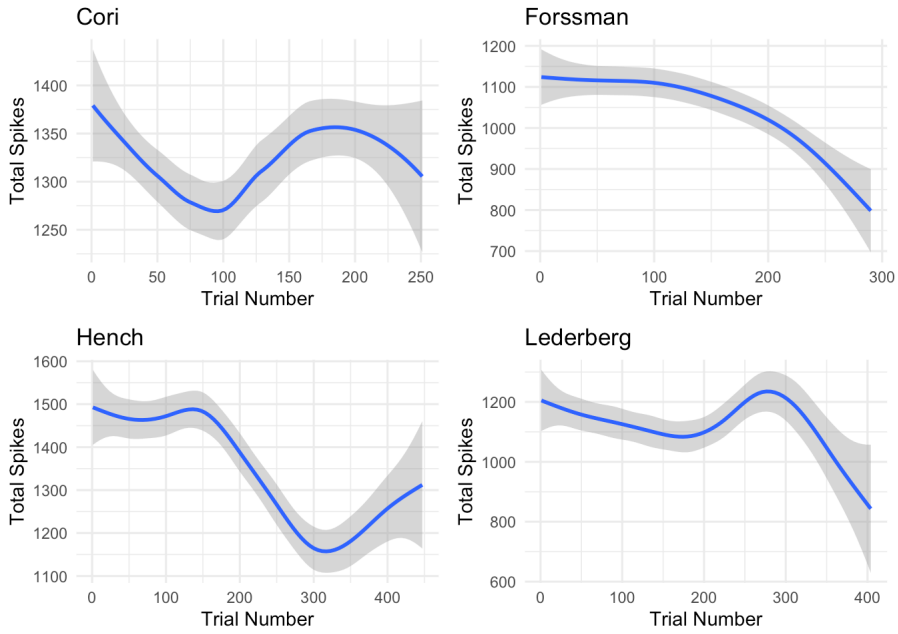
the experiment are becoming less responsive to the stimuli presented in each trial as they undergo successive trials. This could be due to various factors such as adaptation to different stimuli, boredom, or physiological changes in neural circuits as the mice are exposed to more stimuli throughout time.



Spike Information Across all Trials from all 18 sessions: grouped by mouse_name:

Additionally, we generate four visualizations, each corresponding to a different mouse, to illustrate the change in total spikes throughout each trial across all 18 sessions. This allows for a comprehensive comparison of neural activity trends across different experimental mice. Notably, across all four mice, there is a consistent downward trend in the total number of spikes throughout the trials across all sessions. Given our earlier deduction that Session 17, associated with the mouse Lederberg, exhibited the highest feedback accuracy, we closely examine its corresponding graph. We observe that the total spike count reaches its peak value around trial number 280 but experiences a sudden drop towards the end of the trials. This observation aligns with our previous finding, where there was a noticeable decrease in the total number of spike counts with an increasing trial number for Session #17. This trend suggests that the mouse Lederberg may be becoming less responsive to the stimuli presented in each trial as the experiment progresses across all sessions. Utilizing this information, we can infer that later trials may not convey accurate information, which can be critical when constructing predictive models in later sections of our analysis.

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

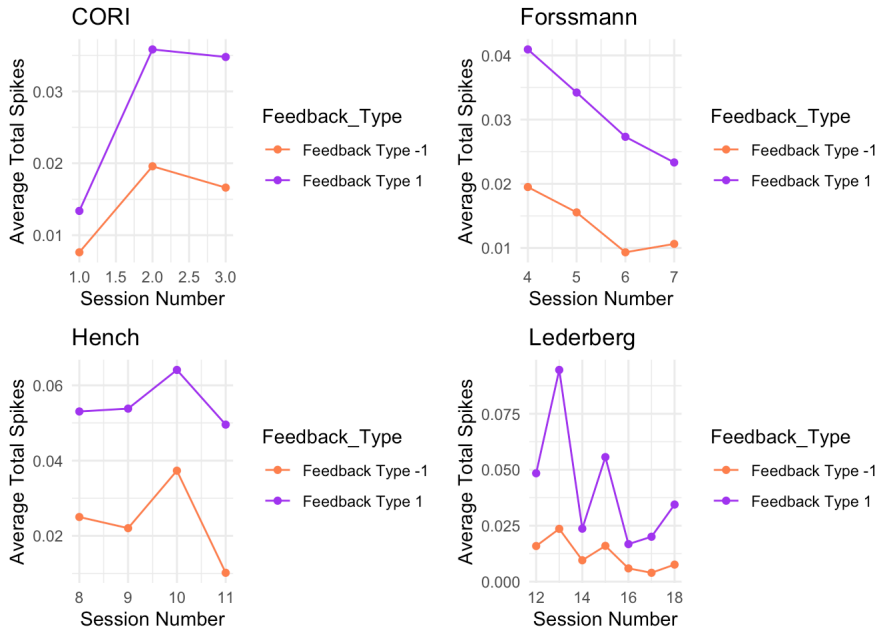


SECTION 3 DATA INTEGRATION

To enhance the predictive performance of our final model, we aim to extract additional information across sessions and mice. Our focus will be on exploring the impact of average spike rate, spike count, and specific brain areas on the task performance of our subjects. We intend to investigate how variations in neural activity levels—whether higher or lower—are associated with successful or unsuccessful feedback types. By delving deeper into these factors, we aim to uncover nuanced relationships between neural activity and task performance. This deeper understanding will enable us to build a more comprehensive and accurate predictive model.

Average Spike Rate across Sessions by Feedback Type

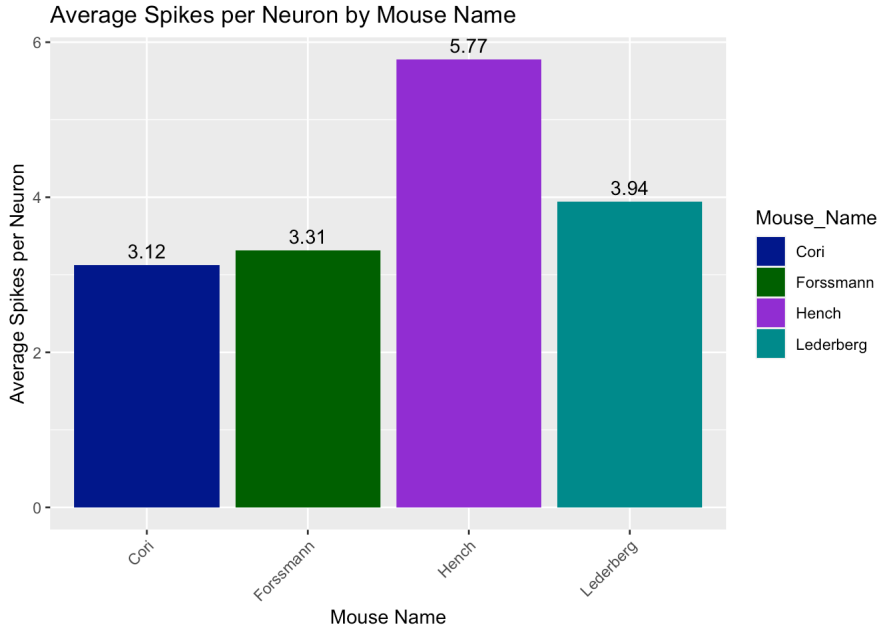
After plotting the average total spikes across all 18 sessions for the four mice (Cori, Forssmann, Hench, and Lederberg) and distinguishing them by feedback type (Success and Failure), a consistent pattern emerges. Across all mice and sessions, the average total spikes are notably higher for the "Success" feedback type compared to the "Failure" feedback type. This finding confirms the association between increased neural activity and improved task performance across all mice and sessions. It indicates that when mice make correct decisions (Success feedback type), there is a correlation with stronger and more pronounced neural activity. These observations provide evidence supporting the importance of including spike frequency in our final model for predicting the outcome of future trials. By incorporating spike frequency data, we capture an important aspect of neural activity that is closely linked to task performance. Including this information enhances the predictive power of our model, allowing for more accurate predictions of trial outcomes. Therefore, the inclusion of spike frequency data in our final model is crucial for capturing the nuanced relationship between neural activity and task performance, ultimately improving our future model's predictive accuracy.



Spike Information Grouped by Mouse Name: Total Spikes, Number of Trials, and Average Spikes:

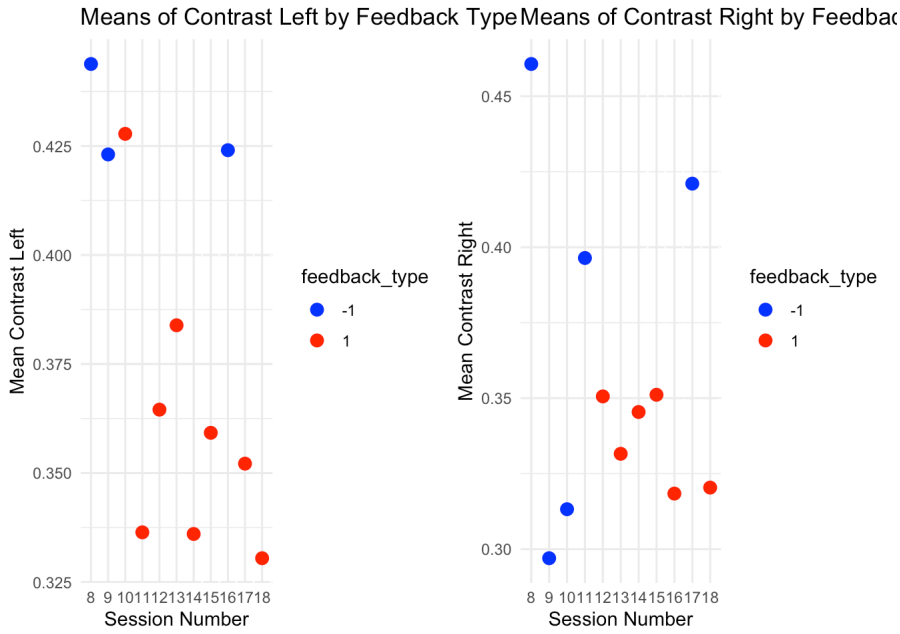
Based on the previous analysis, we can draw the following conclusions: sessions with higher average total spikes tend to be associated with greater feedback_type accuracy (Feedback Type == 1 or Success). To further explore this relationship and identify which mice and their corresponding sessions exhibit the highest average total spikes, we present a bar graph depicting the average spikes per neuron for each of the four distinct mouse names. Since mice may have varying numbers of sessions and therefore trials, we normalize the total number of spikes by dividing by the product of the number of trials and the number of neurons (brain_areas) present in each trial. This normalization allows us to compare the average spike activity across different mice accurately. Our analysis reveals that the mice with the highest average spike per neuron are Hench and Lederberg, which are associated with sessions 8-18. This suggests that sessions 8-18 exhibit high neural activity, potentially contributing to improved task performance. Therefore, it is advisable to include these trials in our final predictive model. Furthermore, our previous findings indicate that Lederberg and its affiliated sessions demonstrate high feedback_type accuracy. This further reinforces our observation from the bar plot below, emphasizing the importance of incorporating these sessions into our final model.

Mouse_Name	Total_Spikes	Total_Neurons	Average_Spikes
Cori	784022	251028	3.123245
Forssmann	1109059	334704	3.313552
Hench	1932830	334704	5.774744
Lederberg	2308017	585732	3.940398



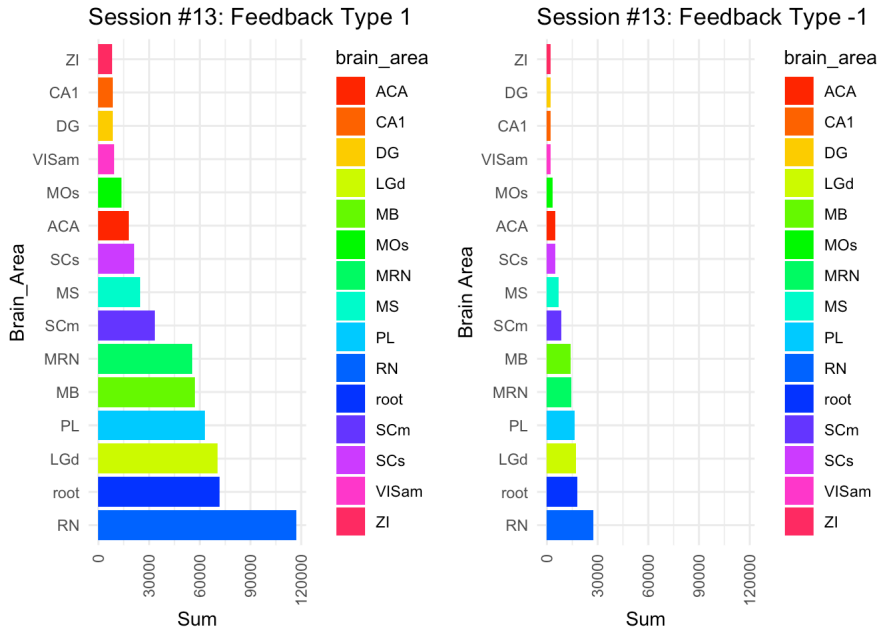
Success Rate versus Mean Contrast Difference (Sessions 8-18)

Next, we analyze the mean levels of both the left and right screens across all trials of each session from sessions 8 to 18 (the reason why we choose these specific sessions is stated above). The dots in the plot are color-coded based on whether they are associated with trials with a feedback_type of -1 (Failure) or 1 (Success). For each session, the visible colored dot on the graph represents the one with the higher mean contrast value between feedback_types. Notably, we observe a larger cluster of red dots for contrast_left, indicating that more sessions had higher mean left contrast values for trials associated with Success. This may suggest that there is a correlation between higher left contrast levels and task success. We can confirm that contrast_left is a valuable predictor variable in the construction of the final model because it may improve the accuracy in predicting task outcomes. However, a notable observation is the relatively smaller number of red dots for contrast_right compared to blue dots. This suggests that fewer sessions had higher mean right contrast values for trials linked with Success. This observation calls for further investigation into the potential implications of contrast_right on task performance and feedback_type accuracy. Therefore we choose to not include the contrast_right in our final model as it may decrease the accuracy in predicting task outcomes.



Total spikes fired for each brain area by feedback type: Session #13

In this section, we investigate the correlation between the number of spikes fired in specific brain areas and the accuracy of feedback. We focused on Session #13 for its inclusion in our final predictive model, covering sessions 8-18, and due to its richness in unique brain areas (15 areas). Our analysis revealed that across all 15 brain areas, trials with a “Success” feedback type (feedback type == 1) exhibited higher neural activity compared to those with a “Failure” feedback type (feedback type == -1). This observation strengthens our prior findings suggesting a robust association between increased neural activity and improved task performance across mice. Furthermore, our analysis suggests that the number of spikes fired in each trial may be more closely correlated with feedback accuracy rather than the specific brain areas involved. Across all 15 brain areas in Session #13, we observed a decrease in brain activity for trials resulting in failure (Feedback Type = -1) compared to success (Feedback Type = 1).These insights highlight the importance of including the number of spikes per trial, regardless of the specific brain area, in our final predictive model, highlighting the potential predictive value of neural activity levels in determining task outcomes.



FINAL DATASET OF DATA INTEGRATION

Based on our previous observations and the information extracted across all sessions and mice in previous sections, we are now ready to construct our final data set for predictive model building. Here are the modifications we have made:

1. Addition of New Variable: Average Spike Counts: We have introduced a new variable, Average Spike Counts, which summarizes information regarding the total number of spikes divided by the number of neurons or brain areas. Through our exploration of average spike rates across sessions by feedback type and the total spikes fired for each brain area by feedback type, we deduced that higher spike counts and neural activity were associated with the Success feedback type. As a result, we included sessions 8-18, involving mice Hench and Lederberg, in our final data set due to their high average spikes per neuron. This addition ensures that sessions with records of high neural activity are included, leading to more accurate outcomes in predicting feedback type accuracy.
2. Exclusion of “brain_area” Variable: Upon investigation of neural activity for each brain area in Session #13 by feedback type, we found that the number of spikes fired in each trial was more closely correlated with feedback accuracy than the specific brain area where the spikes occurred. Consequently, we decided not to include the “brain_area” variable in our dataset.
3. Removal of “contrast_right” Variable: Analysis of mean levels for both left and right screens across all trials of each session by feedback type revealed that fewer sessions had higher mean right contrast values for trials associated with Success. As a result, we concluded that there wasn’t sufficient evidence to associate higher mean right contrast levels with successful feedback type outcomes. Therefore, we excluded the “contrast_right” variable from our analysis.
4. Deletion of Trials from Session #16: Upon observing the average spike rate across sessions and mice by Feedback Type, we found that the average total spikes were the lowest for the “Success” feedback type in Session #14, particularly in the mouse “Lederberg” graph between Sessions 8-18. Given our earlier deduction that higher average spike rates are linked to higher accuracy in feedback type, we decided to remove trials from this session to improve the accuracy of our prediction model.

These refinements aim to enhance the robustness and predictive performance of our final dataset, ensuring that it captures relevant information while eliminating potential sources of noise or bias. The first couple of observations in the “within_sessions_combined_data” data set are printed below. The final data set contains 3163 rows.

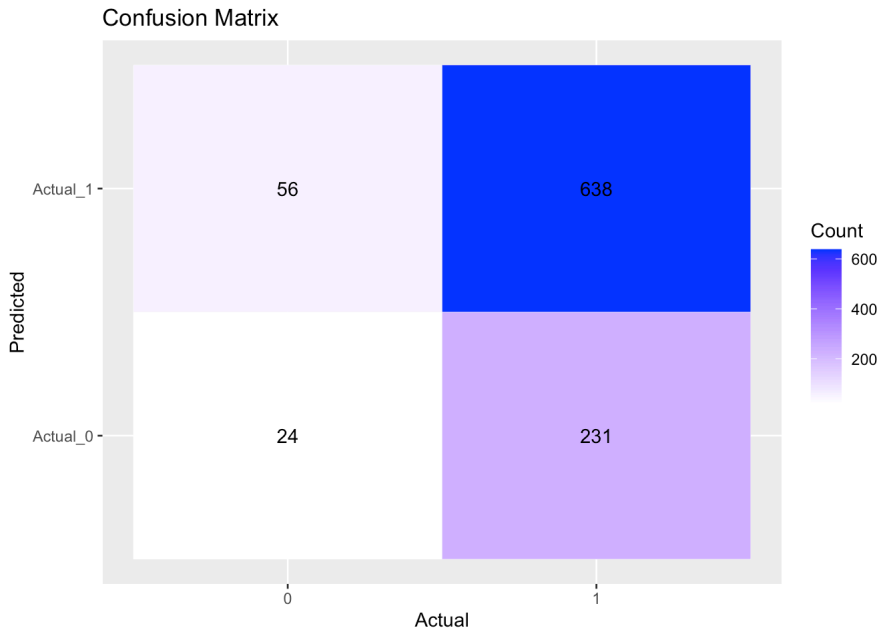
Session_Number	Trial_Number	Mouse_Name	Contrast_Left	Feedback_Type	Number_Neurons	Average_Spikes
8	1	Hench	0.0	1	1157	1.038029
8	2	Hench	1.0	0	1157	1.196197
8	3	Hench	1.0	1	1157	1.178911
8	4	Hench	0.5	1	1157	2.103716
8	5	Hench	0.0	0	1157	1.318928
8	6	Hench	0.0	1	1157	1.353500

SECTION 4 PREDICTIVE MODELING:

First Model: Logistic Regression Model Before constructing our models, I partitioned the final dataset into training (70%) and testing sets (30%) to prevent overfitting. My initial model is a logistic regression model, which is suitable for handling a dependent variable such as “feedback type” that is categorical and binary. The predictor variables are: Contrast_Left + Mouse_Name + Average_Spikes. This regression model aims to estimate the probability that a given trial or observation belongs to either the “Feedback Type Success” or “Feedback Type Failure” category. Upon assessing the accuracy of the Logistic Regression Classification Method, we examine the confusion matrix. The model accurately predicts 24 trials as having “feedback_type == 0” (Failure). However, it misclassifies 231 trials that actually have “feedback_type == 0” as having

"feedback == 1", and it incorrectly identifies 56 trials with "feedback type == 0" when they actually have "feedback == 1". On the other hand, the model correctly predicts that 638 trials belong to "feedback type == 1". The sensitivity rate (True Positive Rate), measuring the proportion of actual positive cases (Feedback type == 1) correctly identified by the model, is 9.412%, indicating a relatively low rate. Conversely, the specificity rate (True Negative Rate), representing the proportion of actual negative cases correctly identified by the model, is 91.93%, indicating a relatively high rate.

Overall, the model's accuracy stands at approximately 69.76%, prompting us to explore additional models using our final dataset.



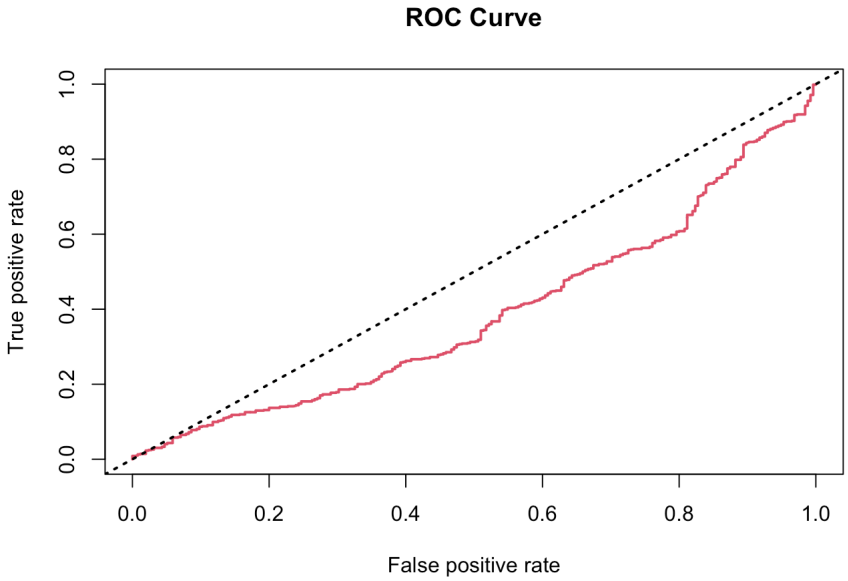

```
## Confusion Matrix and Statistics
##
##           Actual
## Prediction   0   1
##          0  24  56
##          1 231 638
##
##                Accuracy : 0.6976
##                  95% CI : (0.6672, 0.7267)
##     No Information Rate : 0.7313
##     P-Value [Acc > NIR] : 0.9908
##
##                   Kappa : 0.0171
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.09412
##             Specificity : 0.91931
##          Pos Pred Value : 0.30000
##          Neg Pred Value : 0.73418
##              Prevalence : 0.26870
##          Detection Rate : 0.02529
##    Detection Prevalence : 0.08430
##       Balanced Accuracy : 0.50671
##
##        'Positive' Class : 0
##
```

Measuring Machine Learning Metrics at Different Cut-off Probabilities: Logistic Regression Model:

In addition, we will evaluate the logistic regression model's performance across varying threshold values and identify the trade-offs between different metrics such as Sensitivity, Specificity, Accuracy, and Kappa values. The table below showcases the different cutoff probabilities of thresholds at which there will be different accuracy levels. We observe that the accuracy increases to above 70 percent (0.7197050 to be exact) when the cutoff probability becomes 0.55 or 55 percent. Therefore, when we run our test data on Sessions 1 and 18 later, we are aware to set the cutoff probability at 55 percent or lower.

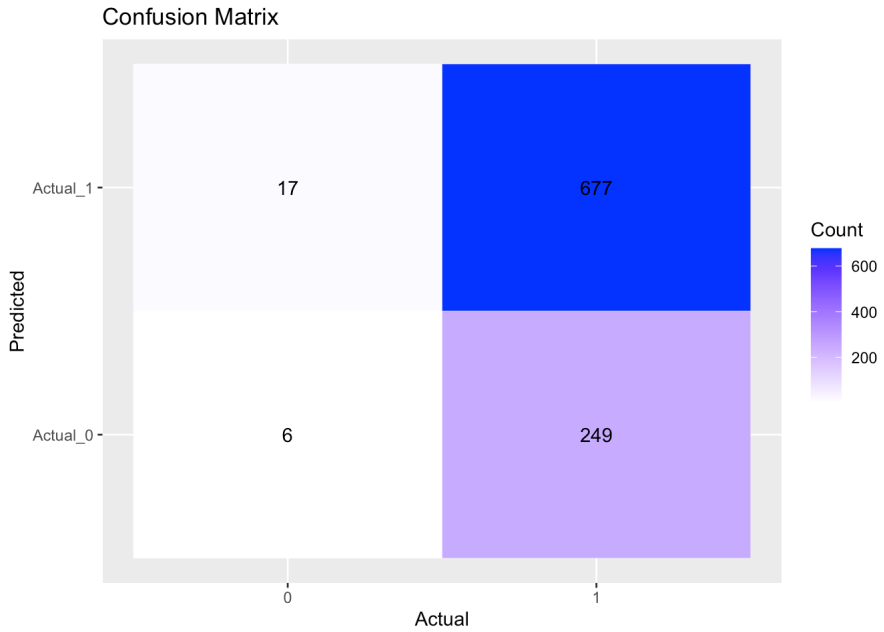
Additionally, we plot the ROC (Receiver Operating Characteristic) curve, which is a graphical representation of the true positive rate (sensitivity) against the false positive rate (specificity) for different threshold values. By plotting this curve, we can assess the performance of a binary classification model throughout different decision threshold values. We observe that the red curve is relatively close to the black diagonal line (which represents a random classifier where the true positive rate equals the false positive rate). Because it is relatively close, we can say that the model is random and has relatively smaller predictive ability. The area under the curve is 0.3869667, which is low, indicating that the model discriminates between positive and negative instances poorly. We must improve this model!

```
##
##
## | cutoff|  Accuracy| Senstivity| Specificity|      kappa|
## |------:|---------:|----------:|-----------:|----------:|
## |   0.10| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.15| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.20| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.25| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.30| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.35| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.40| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.45| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.50| 0.7312961|  1.0000000|   0.0000000|  0.0000000|
## |   0.55| 0.7197050|  0.9755043|   0.0235294| -0.0013566|
## |   0.60| 0.6975764|  0.9193084|   0.0941176|  0.0171481|
## |   0.65| 0.6828240|  0.8270893|   0.2901961|  0.1268855|
## |   0.70| 0.6354057|  0.6873199|   0.4941176|  0.1635827|
## |   0.75| 0.5205479|  0.4337176|   0.7568627|  0.1351109|
## |   0.80| 0.3951528|  0.2161383|   0.8823529|  0.0601464|
## |   0.85| 0.3234984|  0.0864553|   0.9686275|  0.0310074|
## |   0.90| 0.2792413|  0.0158501|   0.9960784|  0.0064622|
```



We lower our threshold value from 60 percent to 55 percent to see if the model accuracy increases. As expected, the accuracy of the model when we decrease the threshold value increases to 71.9 percent. The confusion matrix is presented below as well:

```
## Confusion Matrix and Statistics
##
##          Actual
## Prediction   0   1
##          0   6  17
##          1 249 677
##
##                Accuracy : 0.7197
##                  95% CI : (0.69, 0.7481)
##     No Information Rate : 0.7313
##     P-Value [Acc > NIR] : 0.8006
##
##                   Kappa : -0.0014
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.023529
##             Specificity : 0.975504
##          Pos Pred Value : 0.260870
##          Neg Pred Value : 0.731102
##              Prevalence : 0.268704
##          Detection Rate : 0.006322
##    Detection Prevalence : 0.024236
##       Balanced Accuracy : 0.499517
##
##        'Positive' Class : 0
##
```

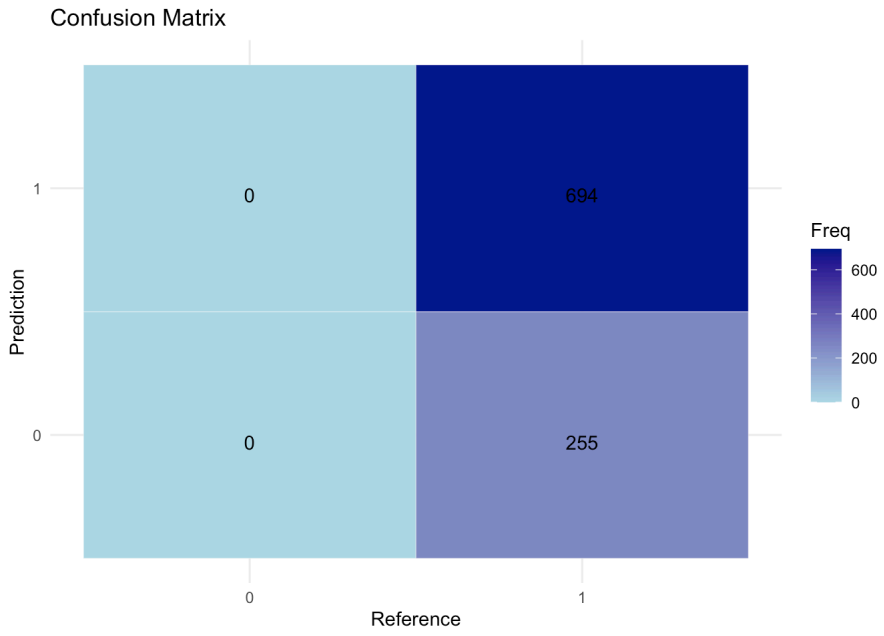


SECOND MODEL: Random Forest Modeling We constructed our second model, the Random Forest model, which is adept at classification tasks by aggregating decision trees and mitigating over fitting. The decision to utilize Random Forest stemmed from its robustness to noise and its ability to generalize effectively to new, unseen data. Furthermore, Random Forest provides valuable insights into feature importance, allowing us to discern which predictor variables are pivotal in predicting feedback type.

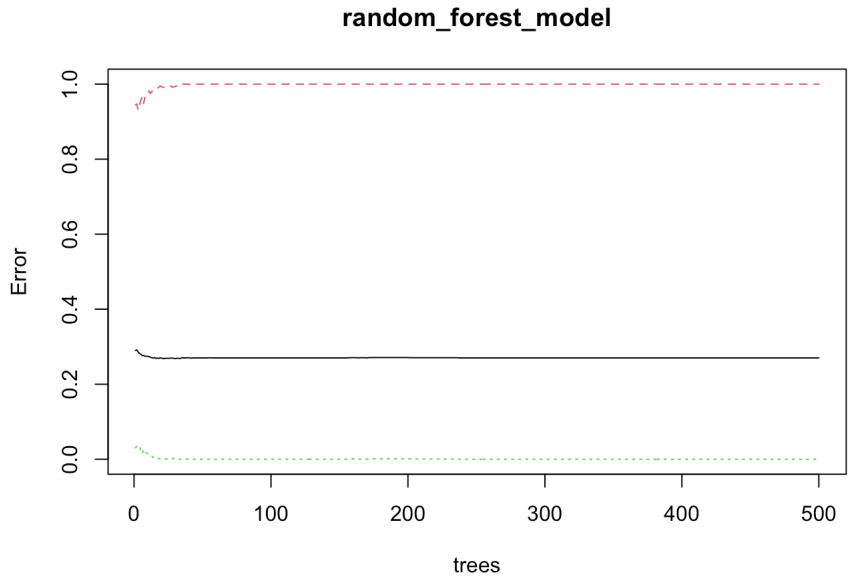
Upon evaluating the Random Forest model, we observed an increase in accuracy compared to the logistic regression model (which had an accuracy of 71.9 percent.) The Random Forest model achieved an accuracy rate of 73.313 percent. The out-of-bag (OOB) error, estimated at around 27.01 percent, serves as an approximation of the model's error rate when applied to unseen data, with lower error rates being preferable. When we look at the confusion matrix, we notice that the model performs poorly in identifying trials belonging to class 0 (Feedback type is a failure). There is a bias towards predicting class 1 (feedback type is a Success). For instance, the confusion matrix shows that the model correctly predicts 1616 trials to be in class 1 while it falsely predicts 598 trials to be class 0 although they belong to class 1. Therefore, we will tune the parameters of the random forest model to see if we can increase the accuracy.

```
##
## Call:
##  randomForest(formula = Feedback_Type ~ Contrast_Left + Mouse_Name +      Average_Spikes, data = dataTrain)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 27.01%
## Confusion matrix:
##   0    1 class.error
## 0 0  598           1
## 1 0 1616           0
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   0   1
##          0   0 255
##          1   0 694
##
##                Accuracy : 0.7313
##                  95% CI : (0.7019, 0.7593)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity :     NA
##             Specificity : 0.7313
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 0.0000
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.2687
##       Balanced Accuracy :     NA
##
##        'Positive' Class : 0
##
```

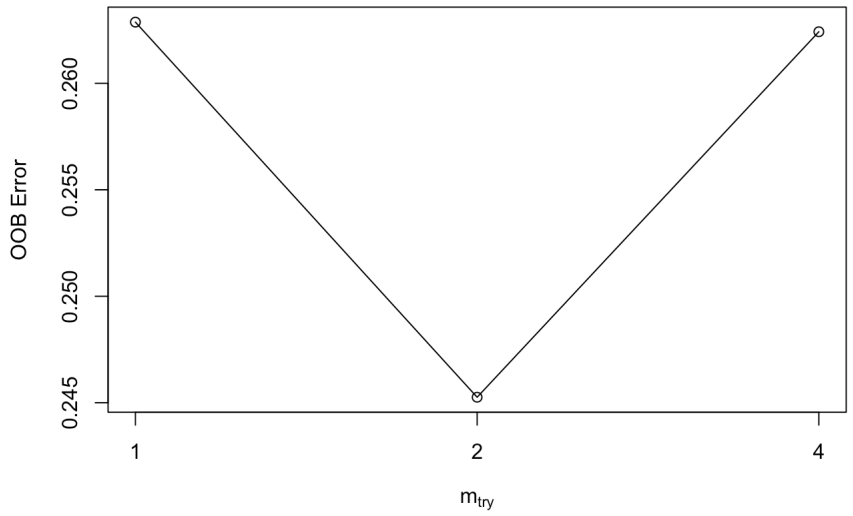


We want to see if we can refine this random forest model. We go about this by plotting the error rates against the number of trees in the random forest model. This will allow us to determine the optimal number of trees that minimizes the error rate. Based on the plot, we can visually identify the point where the error rate stabilizes. We observe that the error rate stops improving significantly after around 200 trees, so we specify the number of trees to use in the model as 200 for better performance.

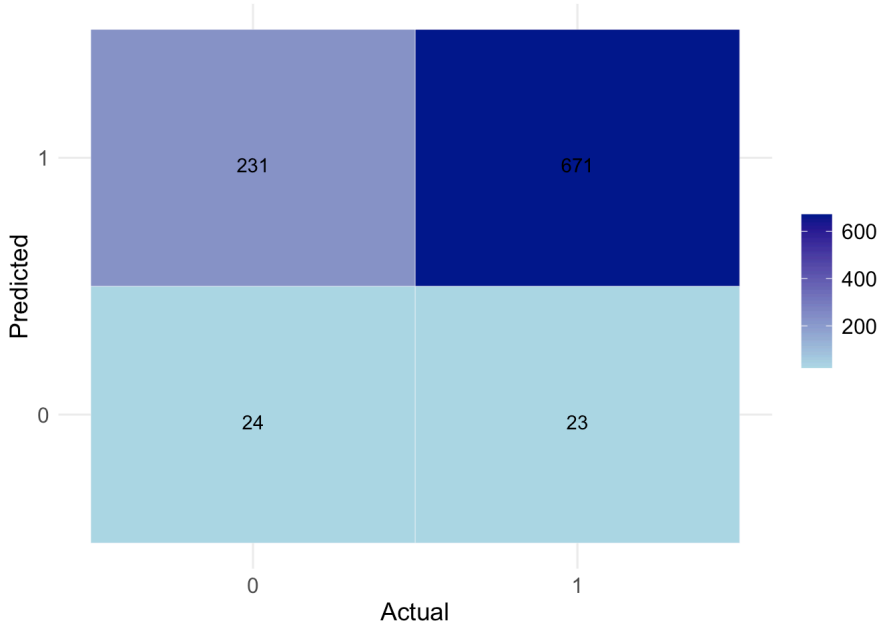


We tune the random forest model by running a function that tests different values of the “mtry” parameters which represents the number of variables randomly samples as candidates at each split to find the best value for the model. The plot below graphs the parameters and the corresponding OOB error rate. We observe that the optimal parameter value is 2 for mtry because it corresponds with the lowest OOB error rate of 0.2425474. We refine our previous random forest model to contain 200 trees (as justified in the previous section) and a mtry value of 2. We get an accuracy rate of 0.7323 which is GREATER than that of the previous random forest model before tuning.

```
## mtry = 2  OOB error = 24.53%
## Searching left ...
## mtry = 4     OOB error = 26.24%
## -0.06998158 0.05
## Searching right ...
## mtry = 1     OOB error = 26.29%
## -0.0718232 0.05
```

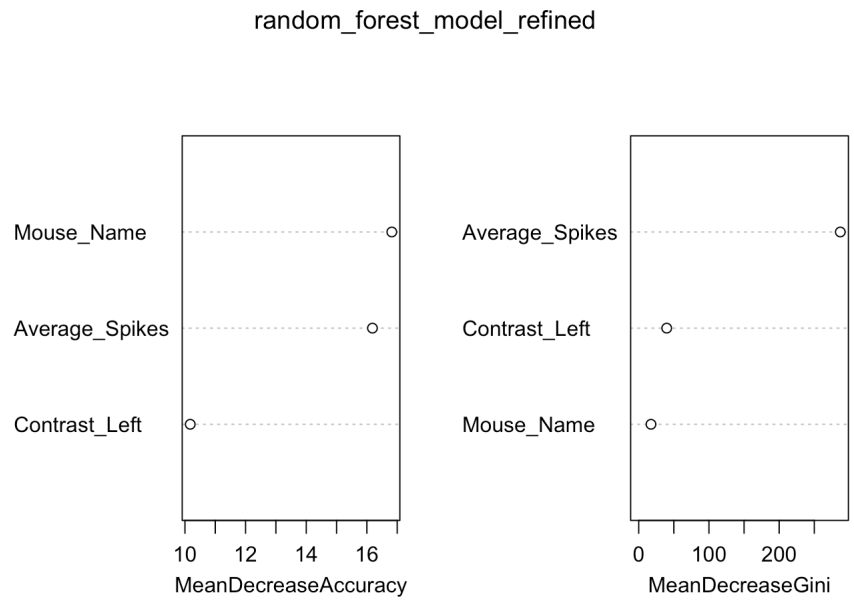


```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   0   1
##          0  24  23
##          1 231 671
##
##                Accuracy : 0.7323
##                  95% CI : (0.703, 0.7603)
##     No Information Rate : 0.7313
##     P-Value [Acc > NIR] : 0.4876
##
##                   Kappa : 0.0822
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.09412
##             Specificity : 0.96686
##          Pos Pred Value : 0.51064
##          Neg Pred Value : 0.74390
##              Prevalence : 0.26870
##          Detection Rate : 0.02529
##    Detection Prevalence : 0.04953
##       Balanced Accuracy : 0.53049
##
##        'Positive' Class : 0
##
```



Using our new random forest model, we create a Variable Important plot (mean decrease accuracy and mean decrease gini). These plots showcase the importance of each predictor variable in classifying the testing data with our random forest model. The Mean Decrease Accuracy plot showcases how much accuracy the model will lose if we take out a specific variable. For instance, we notice that there is a high mean decrease accuracy for average spikes, indicating that the variable is important for accurate classification of feedback type. However, we notice that the Mouse-name is relatively low in mean decrease accuracy meaning that it is not significantly important in predicting the outcomes of our trials. The mean decrease in the Gini coefficient is a measure of how much each predictor variable contributes to the homogeneity of the nodes where a higher value in this score means higher importance. The results of both plots coincide with each other where the average_spikes has the highest mean decrease accuracy and mean decrease in the gini coefficient (287.09245).

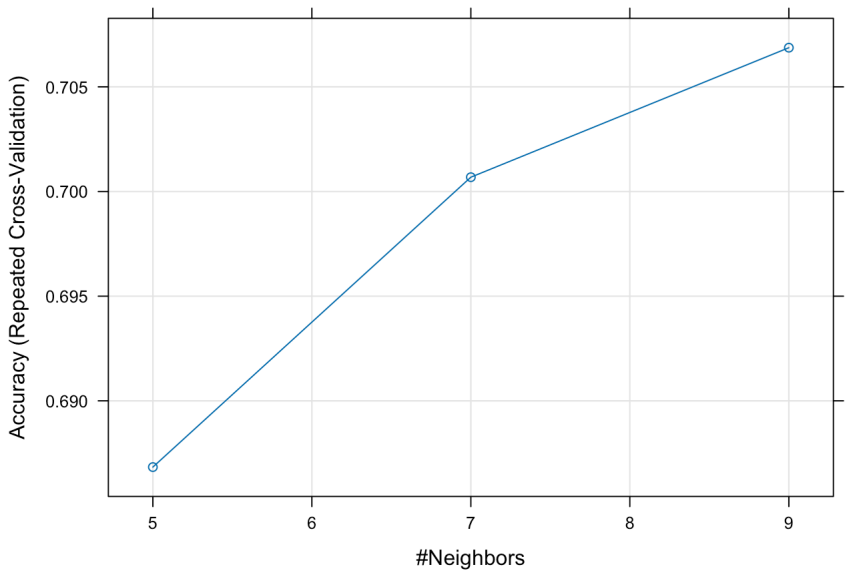
```
##                        0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Contrast_Left   6.399718  8.209594            10.17464         39.83185
## Mouse_Name     16.382396  9.567547            16.82214         17.44391
## Average_Spikes  8.646812 14.875213            16.18394        287.09245
```



THIRD MODEL: KNN Modeling

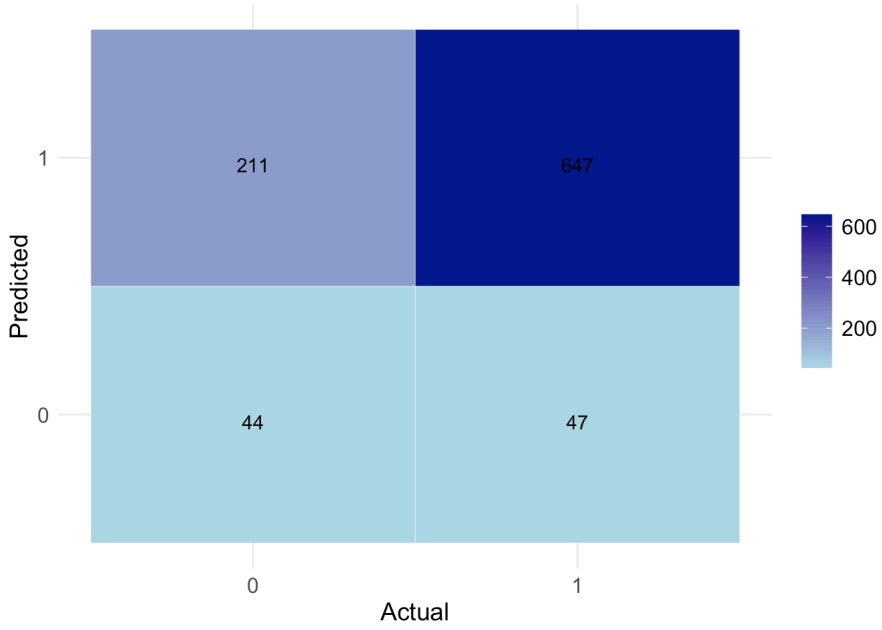
First, we determine the optimal number of clusters to include in our KNN model by plotting an elbow chart by using an algorithm that utilizes 10-fold cross validation. We are able to see that when we use 9 (neighbors k), the accuracy of the model is around 0.7068704 for the training data set.

```
## k-Nearest Neighbors
##
## 2214 samples
##    3 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1992, 1992, 1992, 1993, 1992, 1994, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6868337  0.06593778
##   7  0.7006835  0.06869128
##   9  0.7068704  0.05993460
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```



Next, we run the KNN model on the testing data with k = 9 to observe the confusion matrix and the accuracy rate. We see that the accuracy rate is 0.7281 which is less than that of the random forest model before and after tuning and of that of the logistic regression model after decreasing the cut off probability (accuracy of 0.719).

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  44  47
##          1 211 647
##
##                Accuracy : 0.7281
##                  95% CI : (0.6986, 0.7562)
##     No Information Rate : 0.7313
##     P-Value [Acc > NIR] : 0.6032
##
##                   Kappa : 0.1316
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.17255
##             Specificity : 0.93228
##          Pos Pred Value : 0.48352
##          Neg Pred Value : 0.75408
##              Prevalence : 0.26870
##          Detection Rate : 0.04636
##    Detection Prevalence : 0.09589
##       Balanced Accuracy : 0.55241
##
##        'Positive' Class : 0
##
```



SECTION 5 PREDICTION PERFORMANCE ON THE TEST SETS:

Finally, we will build a prediction model to predict the outcome (i.e., feedback types). The performance will be evaluated on two test sets of 100 trials randomly selected from Session 1 and Session 18, respectively. Our best prediction model was the Random Forest model after tuning it with an accuracy of 73.323 percent. Therefore, we will use this model on our test data:

```
## [1] "Cori"
## [1] "2016-12-14"
## [1] "Lederberg"
## [1] "2017-12-11"
```

Constructing the Data Table for the test data for better visualization:

Combined Data for Each Session with TEST DATA

Session_Number	Trial_Number	Mouse_Name	Date_Exp	Contrast_Left	Contrast_Right	Feedback_Type	Number_Neurons	Number_Time_Bins	Un

Session_Number	Trial_Number	Mouse_Name	Date_Exp	Contrast_Left	Contrast_Right	Feedback_Type	Number_Neurons	Number_Time_Bins	Un
1	1	Cori	2016-12-14	0.25	0.25	Failure	734	40	
1	2	Cori	2016-12-14	0.00	0.00	Success	734	40	
1	3	Cori	2016-12-14	1.00	0.50	Success	734	40	
1	4	Cori	2016-12-14	0.00	0.50	Success	734	40	
1	5	Cori	2016-12-14	1.00	0.50	Success	734	40	
1	6	Cori	2016-12-14	0.50	0.25	Success	734	40	
1	7	Cori	2016-12-14	0.00	1.00	Success	734	40	
1	8	Cori	2016-12-14	1.00	0.00	Failure	734	40	
1	9	Cori	2016-12-14	0.25	0.50	Failure	734	40	
1	10	Cori	2016-12-14	0.00	0.50	Success	734	40	

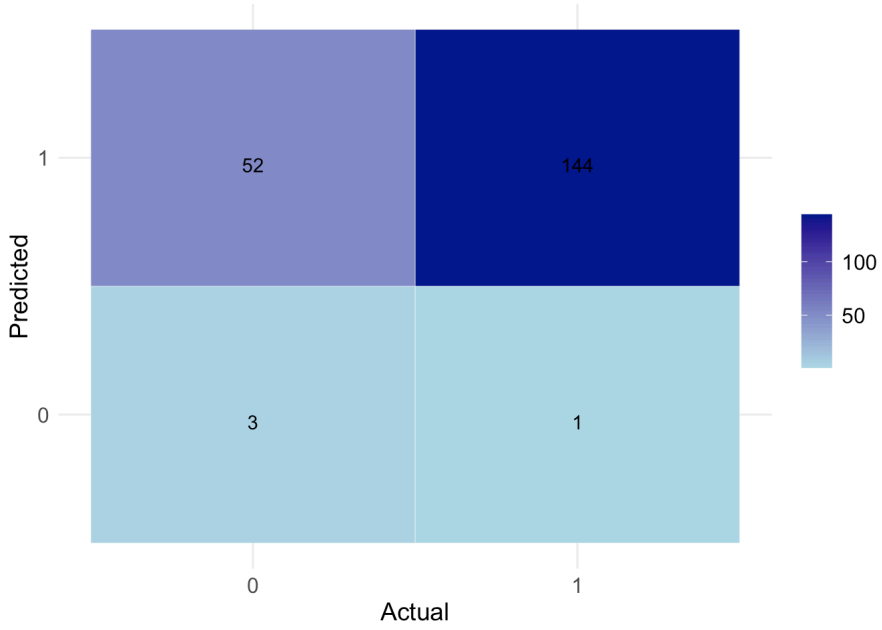
As we did with the sessions data, we will make modifications to the final data set for the test data (from Sessions 1 and 2) to include our variables of interest. We exclude the variables: contrast_right, brain_area, date experiment, number of time bins, and number of unique brain areas. A variable that we choose to add is Average Spikes which is characterized by the total number of spikes fired for each trial divided by the number of unique neurons present. Our final data set should have the variables: Session_Number, Trial_Number, Mouse_Name, Contrast_left, Contrast_Right, Feedback_Type, and Number_Neurons for the testing data. In addition, it should have 200 observations/rows. The first few observations of the data table are listed below:

Session_Number	Trial_Number	Mouse_Name	Contrast_Left	Feedback_Type	Number_Neurons	Average_Spikes
1	1	Cori	0.25	0	734	1.433242
1	2	Cori	0.00	1	734	1.555858
1	3	Cori	1.00	1	734	1.884196
1	4	Cori	0.00	1	734	1.831063
1	5	Cori	1.00	1	734	1.587193
1	6	Cori	0.50	1	734	1.369210

Best Model: Refined Random Forest

We will now fit the refined random forest model to our testing data. Note that the refined random forest model included 3 predictor variables: contrast_left, mouse_name, and average_spikes. It utilized 200 trees and a mtry value of 2 (which was proven to be optimal value). The accuracy rate comes out to be 0.735 or 73.5 percent indicating that our chosen model does a fairly good job in classifying trials as class 0 (Feedback type == -1) or class 1 (Feedback type == 1).

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   3   1
##          1  52 144
##
##                Accuracy : 0.735
##                  95% CI : (0.6681, 0.7948)
##     No Information Rate : 0.725
##     P-Value [Acc > NIR] : 0.4105
##
##                   Kappa : 0.0669
##
##  Mcnemar's Test P-Value : 6.51e-12
##
##             Sensitivity : 0.05455
##             Specificity : 0.99310
##          Pos Pred Value : 0.75000
##          Neg Pred Value : 0.73469
##              Prevalence : 0.27500
##          Detection Rate : 0.01500
##    Detection Prevalence : 0.02000
##       Balanced Accuracy : 0.52382
##
##        'Positive' Class : 0
##
```



SECTION 6 DISCUSSION:

The intricate process of data exploration on a subset derived from an experiment conducted by Steinmetz et al. revealed specific features within the dataset. These features aided in determining the predictor variables to be included in our final model. A detailed examination of graphs illustrating the average spikes across trials for Session #17, alongside line graphs displaying the average spike rate across sessions by feedback type, confirmed a robust association between increased neural activity and improved task performance across all 18 sessions and 4 mice. These findings prompted the integration of information regarding average spike count into our final dataset for predictive modeling. Additionally, we decided to include only sessions 8-18, corresponding to the mice Hench and Lederberg, as they exhibited the highest levels of average spikes per neuron. Another crucial decision was to exclude the variable Contrast_Right from our final dataset. This decision was based on our observation of the means of Contrast_Left and Contrast_Right by feedback type, indicating that fewer sessions had higher mean right contrast values for trials linked with success, thus justifying the inclusion of only Contrast_Left.The last modification involved excluding Session #16 due to its low average total spike values, as depicted by the four line graphs representing average spike rates across sessions by feedback type. This exclusion aimed to mitigate potential inaccuracies in predicting the outcome of trials in our predictive model.

With our final dataset, we conducted predictive modeling using three different models, all incorporating the same predictor variables: Contrast_Left, Mouse_Name, and Average_Spikes. The first model utilized logistic regression and achieved an accuracy rate of 0.6976. Subsequently, by evaluating accuracy rates at different cutoff probabilities using machine learning metrics, we refined the logistic regression model. Lowering the threshold value from 60 percent to 55 percent, we achieved an accuracy rate of 0.7197. The second model employed a random forest algorithm and attained an accuracy rate of 0.7313. Through multiple tests and adjustments, including changing the number of trees to 2 and identifying the optimal value for mtry by examining the out-of-bag (OOB) error rates, we refined the original random forest model.

The refined random forest model achieved an accuracy rate of 0.7323, surpassing that of the original. Lastly, we constructed a KNN model. By determining the optimal number of clusters through 10-fold cross-validation, we identified 9 clusters and ran a KNN model, resulting in an accuracy rate of 0.719.

While considering informative factors such as Sensitivity, Specificity, and Kappa values is crucial in evaluating the best model, our primary objective is to optimize the accuracy level of our final predictive model to fit our testing data. Consequently, we concluded that the random forest model, after tuning, was the best predictive model, boasting the highest accuracy rate of 0.7323. When fitting this best model to our testing data, comprising 100 random trials from 2 sessions, it yielded an accuracy rate of 0.735. This result suggests a fairly robust model in classifying trials as either having a success or failure feedback type.

Reference

Steinmetz, N.A., Zatka-Haas, P., Carandini, M. et al. Distributed coding of choice, action and engagement across the mouse brain. Nature 576, 266–273 (2019). https://doi.org/10.1038/s41586-019-1787-x (https://doi.org/10.1038/s41586-019-1787-x)

#Acknowledgements: 1. Afra Raza (classmate in STA 141A) 2. Ehsaan Mohammed (classmate in STA 141A) 3. CHAT GPT: I have included all the links of the history below: https://chat.openai.com/share/cdc1e0be-7c41-4a30-9264-8b117390f3b2 (https://chat.openai.com/share/cdc1e0be-7c41-4a30-9264-8b117390f3b2) https://chat.openai.com/share/6433cf57-8468-4192-bfec-5a1e7a45c345 (https://chat.openai.com/share/6433cf57-8468-4192-bfec-5a1e7a45c345) https://chat.openai.com/share/94b5525f-fcef-4e22-9cdd-6fc446d00918 (https://chat.openai.com/share/94b5525f-fcef-4e22-9cdd-6fc446d00918) https://chat.openai.com/share/c80ad9c5-c6dd-497d-b7b7-958d45104ca8 (https://chat.openai.com/share/c80ad9c5-c6dd-497d-b7b7-958d45104ca8) https://chat.openai.com/share/11dd0460-c53b-4311-8524-57268c9001c5 (https://chat.openai.com/share/11dd0460-c53b-4311-8524-57268c9001c5) https://chat.openai.com/share/546d8e08-31a2-4868-8fa5-482e6c992b83 (https://chat.openai.com/share/546d8e08-31a2-4868-8fa5-482e6c992b83) https://chat.openai.com/share/a9a812ab-a4d1-459b-a0cd-bff885e310bb (https://chat.openai.com/share/a9a812ab-a4d1-459b-a0cd-bff885e310bb) https://chat.openai.com/share/5d8f535f-b612-4a92-9c8d-f20d91d82569 (https://chat.openai.com/share/5d8f535f-b612-4a92-9c8d-f20d91d82569) https://chat.openai.com/share/34f2538f-d292-4456-96c7-47afd10afbec (https://chat.openai.com/share/34f2538f-d292-4456-96c7-47afd10afbec) https://chat.openai.com/share/79a1edf0-e256-44ab-b13e-123c0f3c86b5 (https://chat.openai.com/share/79a1edf0-e256-44ab-b13e-123c0f3c86b5) https://chat.openai.com/share/81b66bef-de47-4d21-b71f-1dba13ec2a1c (https://chat.openai.com/share/81b66bef-de47-4d21-b71f-1dba13ec2a1c) https://chat.openai.com/share/18ff3fb0-939a-4c6d-9fee-a61d93f442a6 (https://chat.openai.com/share/18ff3fb0-939a-4c6d-9fee-a61d93f442a6) https://chat.openai.com/share/7b1cba2e-4669-4df4-90d2-558da8d15b8b (https://chat.openai.com/share/7b1cba2e-4669-4df4-90d2-558da8d15b8b)

4. https://rpubs.com/SameerMathur/LR_GLM_CCDefault (https://rpubs.com/SameerMathur/LR_GLM_CCDefault)
5. https://www.youtube.com/watch?v=dJclNIN-TPo (https://www.youtube.com/watch?v=dJclNIN-TPo)
6. https://www.datacamp.com/tutorial/k-nearest-neighbors-knn-classification-with-r-tutorial (https://www.datacamp.com/tutorial/k-nearest-neighbors-knn-classification-with-r-tutorial)
7. https://www.geeksforgeeks.org/k-nn-classifier-in-r-programming/ (https://www.geeksforgeeks.org/k-nn-classifier-in-r-programming/)
8. https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/ (https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/)
9. https://www.geeksforgeeks.org/variable-importance-plot-using-random-forest-package-in-r/ (https://www.geeksforgeeks.org/variable-importance-plot-using-random-forest-package-in-r/)
10. https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html (https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html)
11. https://plos.figshare.com/articles/figure/Variable_importance_plot_mean_decrease_accuracy_and_mean_decrease_Gini_/12060105 (https://plos.figshare.com/articles/figure/Variable_importance_plot_mean_decrease_accuracy_and_mean_decrease_Gini_/12060105)
12. https://rpubs.com/pmtam/knn (https://rpubs.com/pmtam/knn)
13. TA DEMOS on CANVAS

Appendix

```
library(dplyr)
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(knitr)
library(kableExtra)
library(gt)
library(grid)
library(lattice)
library(caret)
library(ggplot2)
library(reshape2)
library(caret)
library(dplyr)
library(randomForest)
library(class)
session=list()
for(i in 1:18){
  session[[i]]=readRDS(paste('datasets/session',i,'.rds',sep=''))
  print(session[[i]]$mouse_name)
  print(session[[i]]$date_exp)
}
# Create an empty data frame
summary_table <- data.frame(
  Session_Number = numeric(),
  Experiment_Date = character(),
  Mouse_Name = character(),
  Number_Neurons = numeric(),
  Number_Trials = numeric(),
  Unique_Stimuli_Conditions = character(),
  Unique_Feedback_Types = character(),
  stringsAsFactors = FALSE
)

# Loop through each session
for (i in 1:18) {
  session_data <- session[[i]]  # Access the data for the current session
  session_number <- i
  experiment_date <- unique(session_data$date_exp)
  mouse_name <- unique(session_data$mouse_name)
  number_neurons <- length(session_data$brain_area)
  number_trials <- length(session_data$feedback)
  unique_stimuli_conditions <- paste("Contrast_Left:", paste(unique(session_data$contrast_left), collapse = ",
"),
                                     "Contrast_Right:", paste(unique(session_data$contrast_right), collapse = ",
"))
  unique_feedback_types <- paste(unique(session_data$feedback_type), collapse = ", ")

  # Append data to the summary table
  summary_table <- rbind(summary_table, data.frame(
    Session_Number = session_number,
    Experiment_Date = experiment_date,
    Mouse_Name = mouse_name,
    Number_Neurons = number_neurons,
    Number_Trials = number_trials,
    Unique_Stimuli_Conditions = unique_stimuli_conditions,
    Unique_Feedback_Types = unique_feedback_types
  ))
}

# Print summary table with kable and kableExtra formatting
kable(summary_table, format = "html", align = "c", caption = "Summary Table") %>%
  kable_styling(full_width = FALSE) %>%
  row_spec(0, bold = TRUE)  # Bold header row
library(knitr)

 within_session_combined_data <- data.frame(
  Session_number = numeric(),
  Trial_Number = numeric(),
  Mouse_Name = character(),
  Date_Exp = character(),
  Contrast_Left = numeric(),
  Contrast_Right = numeric(),
  Feedback_Type = numeric(),
  Number_Neurons = numeric(),
  Number_Time_Bins = numeric(),
```

```
  Unique_Brain_Areas = numeric(),
  stringsAsFactors = FALSE
)

for (i in 1:18) {
  session_data <- session[[i]]
  feedback_label <- ifelse(session_data$feedback_type == 1, "Success", "Failure")

  within_session_combined_data <- rbind(
    within_session_combined_data,
    data.frame(
      Session_Number = i,
      Trial_Number = seq_along(session_data$contrast_left),
      Mouse_Name = session_data$mouse_name,
      Date_Exp = session_data$date_exp,
      Contrast_Left = session_data$contrast_left,
      Contrast_Right = session_data$contrast_right,
      Feedback_Type =  feedback_label,
      Number_Neurons = length(session_data$brain_area),
      Number_Time_Bins = ncol(session_data$spks[[1]]),
      Unique_Brain_Areas = length(unique(session_data$brain_area))
    )
  )
}
rownames(within_session_combined_data) <- NULL

kable(head(within_session_combined_data, 10), caption = "Combined Data for Each Session")

load_session_data <- function(session_number) {
  neural_activity_session <- data.frame()

  for (i in 1:length(session[[session_number]]$spks)) {
    spks_matrix <- session[[session_number]]$spks[[i]]
    spks_df <- as.data.frame(spks_matrix)

    rownames(spks_df) <- paste("Trial_", i, "_Neuron_", 1:nrow(spks_matrix))
    colnames(spks_df) <- paste("#Spikes of Neurons in Time Bin ", 1:ncol(spks_matrix), sep = "")

    trial_number <- rep(i, nrow(spks_df))
    spks_df$Trial <- trial_number

    neuron_number <- 1:nrow(spks_matrix)
    spks_df$Neuron_Number <- neuron_number

    brain_area <- session[[session_number]]$brain_area
    spks_df$brain_area <- brain_area

    if (nrow(neural_activity_session) == 0) {
      neural_activity_session <- spks_df
    } else {
      neural_activity_session <- rbind(neural_activity_session, spks_df)
    }
  }

  neural_activity_session <- neural_activity_session[, c("Trial", "Neuron_Number", "brain_area", setdiff(names(ne
ural_activity_session), c("Trial", "Neuron_Number", "brain_area")))]

  return(neural_activity_session)
}

neural_activity_session_17 <- load_session_data(17)
neural_activity_head<- head(neural_activity_session_17, 5)

# Create a HTML table with kableExtra
neural_activity_table <- kable(neural_activity_head, format = "html", align = "c", caption = "Top 5 Observations
of Neural Activity for Session 17") %>%
  kable_styling(full_width = FALSE)  # Apply styling to the table

# Print the table
neural_activity_table
accuracy <- numeric(length = 18)
mouse_names <- character(length = 18)

for (i in 1:18) {
  session_data <- session[[i]]
  mouse_names[i] <- unique(session_data$mouse_name)
}

session_mouse_table <- data.frame(Session_Number = 1:18, Mouse_Name = mouse_names)
```

```
accuracy <- numeric(length = 18)

for (i in 1:18) {
  session_data <- session[[i]]
  feedback_type <- session_data$feedback_type

  accuracy[i] <- mean(feedback_type == 1)
}

summary_table <- cbind(session_mouse_table, Feedback_Accuracy = accuracy)
summary_table <- summary_table[order(summary_table$Feedback_Accuracy), ]
ggplot(summary_table, aes(x = factor(Session_Number), y = Feedback_Accuracy, fill = Mouse_Name)) +
  geom_bar(stat = "identity") +
  labs(x = "Session Number", y = "Feedback Accuracy") +
  ggtitle("Feedback Accuracy by Session Number") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
unique_brain_areas_table <- data.frame(Session_Number = 1:18, Unique_Brain_Areas = NA, Total_Unique_Brain_Areas =
NA)
for (i in 1:18) {
    unique_areas <- unique(session[[i]]$brain_area)
    unique_areas_string <- paste(unique_areas, collapse = ", ")
    unique_brain_areas_table[i, "Unique_Brain_Areas"] <- unique_areas_string
    total_unique_areas <- length(unique_areas)
    unique_brain_areas_table[i, "Total_Unique_Brain_Areas"] <- total_unique_areas
}
kable(unique_brain_areas_table, caption = "Unique Brain Areas in Each Session")
calculate_average_spikes <- function(session_number, trial_number) {
  # Load data for the specified session
  neural_activity_session <- load_session_data(session_number)

  trial_spike <- neural_activity_session %>%
    filter(Trial == trial_number) %>%
    group_by(brain_area) %>%
    summarise(Total_Spikes = sum(across(starts_with("#Spikes of Neurons in Time Bin"))))

  trial_brain_area <- neural_activity_session %>%
    filter(Trial == trial_number) %>%
    group_by(brain_area) %>%
    count(name = "Total_Brain_Areas")
  trial_average <- inner_join(trial_spike, trial_brain_area, by = "brain_area") %>%
    mutate(Average_Spikes = Total_Spikes / Total_Brain_Areas)

  return(trial_average)
}

session_trial_spike <- calculate_average_spikes(session_number = 17, trial_number = 1) #Session number = 1 and tr
ial number = 1.
print(kable(session_trial_spike))

ggplot(session_trial_spike, aes(x = brain_area, y = Average_Spikes, fill = brain_area)) +
  geom_bar(stat = "identity") +
  labs(x = "Brain Area", y = "Average Spike", title = "Average Spike by Brain Area for Session #17/Trial #1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability

plot_average_spikes_zoom <- function(session_number, start_trial, end_trial) {
  # Load data for the specified session
  neural_activity_session <- load_session_data(session_number)
  combined_data <- data.frame()
  for (trial_number in start_trial:end_trial) {
    trial_spike <- neural_activity_session %>%
      filter(Trial == trial_number) %>%
      group_by(brain_area) %>%
      summarise(Total_Spikes = sum(across(starts_with("#Spikes of Neurons in Time Bin"))))
    trial_brain_area <- neural_activity_session %>%
      filter(Trial == trial_number) %>%
      group_by(brain_area) %>%
      count(name = "Total_Brain_Areas")

    trial_average <- inner_join(trial_spike, trial_brain_area, by = "brain_area") %>%
      mutate(Average_Spikes = Total_Spikes / Total_Brain_Areas) %>%
      mutate(Trial = trial_number)  # Add trial number to the data

    combined_data <- rbind(combined_data, trial_average)
  }

  ggplot(combined_data, aes(x = Trial, y = Average_Spikes, color = brain_area)) +
    geom_line() +
    labs(x = "Trial", y = "Average Spikes", title = paste("Average Spikes Across Trials for Session", session_num
```

```
ber)) +
    scale_color_discrete(name = "Brain Area") +
    xlim(start_trial, end_trial)  # Set the x-axis limits
}

session_number <- 17
start_trial <- 5
end_trial <- 100
plot_average_spikes_zoom(session_number, start_trial, end_trial)
plot_average_spikes_zoom <- function(session_number) {
  # Load data for the specified session
  neural_activity_session <- load_session_data(session_number)

  trial_numbers <- c()
  total_spikes <- c()

  for (trial_number in unique(neural_activity_session$Trial)) {
    trial_spike <- neural_activity_session %>%
      filter(Trial == trial_number) %>%
      summarise(Total_Spikes = sum(across(starts_with("#Spikes of Neurons in Time Bin"))))

    trial_numbers <- c(trial_numbers, trial_number)
    total_spikes <- c(total_spikes, trial_spike$Total_Spikes)
  }

  trial_data <- data.frame(Trial = trial_numbers, Total_Spikes = total_spikes)

  # Fit linear regression line
  trend_model <- lm(Total_Spikes ~ Trial, data = trial_data)
  trend_data <- data.frame(Trial = trial_data$Trial, Trend = predict(trend_model))

  # Calculate slope line
  slope_intercept <- coef(trend_model)
  slope_line <- data.frame(Trial = c(min(trial_data$Trial), max(trial_data$Trial)),
                           Trend = c(min(trial_data$Trial), max(trial_data$Trial)) * slope_intercept[2] + slope_
intercept[1])

  ggplot(trial_data, aes(x = Trial, y = Total_Spikes, color = factor(Trial))) +
    geom_line(color = "blue") +
    geom_point(color = "blue") +
    geom_line(data = trend_data, aes(x = Trial, y = Trend), linetype = "dashed", color = "red") +  # Trend line
    geom_line(data = slope_line, aes(x = Trial, y = Trend), linetype = "dotted", color = "green") +  # Slope line
    labs(x = "Trial", y = "Total Spike Counts", title = paste("Total Spike Counts Across Trials for Session", ses
sion_number)) +
    scale_color_discrete(name = "Trial")
}

session_number <- 17
plot_average_spikes_zoom(session_number)
process_sessions_and_plot <- function(session_range, mouse_name) {
  trial_numbers_all <- numeric()
  total_spikes_per_trial_all <- numeric()

  for (session_index in session_range) {
    trial_numbers <- numeric()
    total_spikes_per_trial <- numeric()

    for (i in 1:length(session[[session_index]]$spks)) {
      total_spikes <- sum(session[[session_index]]$spks[[i]])
      trial_numbers <- c(trial_numbers, i)
      total_spikes_per_trial <- c(total_spikes_per_trial, total_spikes)
    }

    trial_numbers_all <- c(trial_numbers_all, trial_numbers)
    total_spikes_per_trial_all <- c(total_spikes_per_trial_all, total_spikes_per_trial)
  }

  trial_data_all <- data.frame(Trial = trial_numbers_all, Total_Spikes = total_spikes_per_trial_all)

  ggplot(trial_data_all, aes(x = Trial, y = Total_Spikes)) +
    geom_smooth() +
    labs(x = "Trial Number", y = "Total Spikes", title = paste(mouse_name)) +
    theme_minimal()
}
plot_cori <- process_sessions_and_plot(1:3, "Cori")
plot_forssman <- process_sessions_and_plot(4:7, "Forssman")
plot_hench <- process_sessions_and_plot(8:11, "Hench")
plot_lederberg <- process_sessions_and_plot(12:18, "Lederberg")
grid.arrange(plot_cori, plot_forssman, plot_hench, plot_lederberg, ncol = 2)
```

```
# Function to process mouse data and create plots
process_mouse_data <- function(session_range, mouse_name) {
  # Initialize vectors to store total spike counts for each session (positive and negative feedback types)
  total_spikes_all_sessions_pos <- numeric(length(session_range))
  total_spikes_all_sessions_neg <- numeric(length(session_range))

  # Iterate through sessions in the specified range
  for (i in session_range) {
    # Filter feedback type 1 and -1
    spikes_pos <- sum(unlist(session[[i]]$spk[session[[i]]$feedback_type == 1]))
    spikes_neg <- sum(unlist(session[[i]]$spk[session[[i]]$feedback_type == -1]))

    # Store total spike counts
    total_spikes_all_sessions_pos[i - min(session_range) + 1] <- spikes_pos
    total_spikes_all_sessions_neg[i - min(session_range) + 1] <- spikes_neg
  }

  # Divide total spike counts by the number of rows
  total_spikes_all_sessions_pos <- total_spikes_all_sessions_pos / 6133928
  total_spikes_all_sessions_neg <- total_spikes_all_sessions_neg / 6133928

  # Create data frames for positive and negative feedback types
  data_pos <- data.frame(Session = session_range, Average_Total_Spikes = total_spikes_all_sessions_pos, Feedback_
Type = "Feedback Type 1")
  data_neg <- data.frame(Session = session_range, Average_Total_Spikes = total_spikes_all_sessions_neg, Feedback_
Type = "Feedback Type -1")

  # Combine data frames
  data_combined <- rbind(data_pos, data_neg)

  # Plot
  ggplot(data_combined, aes(x = Session, y = Average_Total_Spikes, color = Feedback_Type)) +
    geom_line() +
    geom_point() +
    scale_color_manual(values = c("coral", "purple")) +
    labs(x = "Session Number", y = "Average Total Spikes", title = paste(mouse_name)) +
    theme_minimal()
}

# Plot for mouse CORI
plot_cori <- process_mouse_data(1:3, "CORI")
# Plot for mouse Forssmann
plot_forssman <- process_mouse_data(4:7, "Forssmann")
# Plot for mouse Hench
plot_hench <- process_mouse_data(8:11, "Hench")
# Plot for mouse Lederberg
plot_lederberg <- process_mouse_data(12:18, "Lederberg")
# Arrange plots together
grid.arrange(plot_cori, plot_forssman, plot_hench, plot_lederberg, ncol = 2)

total_spikes_table <- data.frame(Mouse_Name = character(), Total_Spikes = numeric(), Total_Neurons = numeric(), A
verage_Spikes = numeric())

for (session_number in 1:18) {
  mouse_name <- session[[session_number]]$mouse_name
  total_spikes <- 0
  for (i in 1:length(session[[session_number]]$spks)) {
    total_spikes <- total_spikes + sum(session[[session_number]]$spks[[i]])
  }
  total_neurons <- sum(length(session[[1]]$spks) * length(session[[1]]$brain_area))
  session_data <- data.frame(Mouse_Name = mouse_name, Total_Spikes = total_spikes, Total_Neurons = total_neurons)
  total_spikes_table <- rbind(total_spikes_table, session_data)
}

total_spikes_summary <- total_spikes_table %>%
  group_by(Mouse_Name) %>%
  summarise(Total_Spikes = sum(Total_Spikes),
            Total_Neurons = sum(Total_Neurons),
            Average_Spikes = Total_Spikes / Total_Neurons)

kable(total_spikes_summary)

ggplot(total_spikes_summary, aes(x = Mouse_Name, y = Average_Spikes, fill = Mouse_Name)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(Average_Spikes, 2)), vjust = -0.5) +  # Add labels for average spikes
  labs(x = "Mouse Name", y = "Average Spikes per Neuron", title = "Average Spikes per Neuron by Mouse Name") +
  scale_fill_manual(values = c("darkblue", "darkgreen", "darkviolet", "darkcyan", "darkmagenta")) +  # Set less b
right colors
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability
```

```
# Create an empty list to store the filtered data frames
filtered_data <- list()

# Loop through sessions 8 to 18
for (session_number in 8:18) {
  # Calculate the means for the current session
  mean_contrast_left_feedback_1 <- mean(session[[session_number]]$contrast_left[session[[session_number]]$feedbac
k_type == 1])
  mean_contrast_left_feedback_minus_1 <- mean(session[[session_number]]$contrast_left[session[[session_number]]$f
eedback_type == -1])

  # Check which mean contrast is greater
  if (mean_contrast_left_feedback_1 > mean_contrast_left_feedback_minus_1) {
    # If mean contrast for feedback type 1 is greater, store the data for feedback type 1
    filtered_data[[session_number - 7]] <- data.frame(
      session_number = session_number,
      feedback_type = factor(1),
      mean_contrast_left = mean_contrast_left_feedback_1
    )
  } else {
    # If mean contrast for feedback type -1 is greater or equal, store the data for feedback type -1
    filtered_data[[session_number - 7]] <- data.frame(
      session_number = session_number,
      feedback_type = factor(-1),
      mean_contrast_left = mean_contrast_left_feedback_minus_1
    )
  }
}

# Combine the filtered data frames into a single data frame
combined_data <- do.call(rbind, filtered_data)

# Plot the points where the mean contrast is greater than the mean contrast of the other feedback type
p <- ggplot(combined_data, aes(x = as.factor(session_number), y = mean_contrast_left, color = feedback_type)) +
  geom_point(size = 3) +
  geom_vline(xintercept = 8:18 - 0.5, linetype = "dotted", color = "white") + # Add vertical lines between each s
ession number
  labs(x = "Session Number", y = "Mean Contrast Left", title = "Means of Contrast Left by Feedback Type") +
  scale_color_manual(values = c("blue", "red")) +  # Color for feedback type 1 and -1 respectively
  theme_minimal()

# Create an empty list to store the filtered data frames for contrast_right
filtered_data_right <- list()

# Loop through sessions 8 to 18
for (session_number in 8:18) {
  # Calculate the means for the current session for contrast_right
  mean_contrast_right_feedback_1 <- mean(session[[session_number]]$contrast_right[session[[session_number]]$feedb
ack_type == 1])
  mean_contrast_right_feedback_minus_1 <- mean(session[[session_number]]$contrast_right[session[[session_number]]
$feedback_type == -1])

  # Check which mean contrast is greater for contrast_right
  if (mean_contrast_right_feedback_1 > mean_contrast_right_feedback_minus_1) {
    # If mean contrast for feedback type 1 is greater for contrast_right, store the data for feedback type 1
    filtered_data_right[[session_number - 7]] <- data.frame(
      session_number = session_number,
      feedback_type = factor(1),
      mean_contrast_right = mean_contrast_right_feedback_1
    )
  } else {
    # If mean contrast for feedback type -1 is greater or equal for contrast_right, store the data for feedback t
ype -1
    filtered_data_right[[session_number - 7]] <- data.frame(
      session_number = session_number,
      feedback_type = factor(-1),
      mean_contrast_right = mean_contrast_right_feedback_minus_1
    )
  }
}

# Combine the filtered data frames into a single data frame for contrast_right
combined_data_right <- do.call(rbind, filtered_data_right)

# Plot the points where the mean contrast is greater than the mean contrast of the other feedback type for contra
st_right
p_right <- ggplot(combined_data_right, aes(x = as.factor(session_number), y = mean_contrast_right, color = feedba
ck_type)) +
  geom_point(size = 3) +
```

```
  geom_vline(xintercept = 8:18 - 0.5, linetype = "dotted", color = "white") + # Add vertical lines between each s
ession number
  labs(x = "Session Number", y = "Mean Contrast Right", title = "Means of Contrast Right by Feedback Type") +
  scale_color_manual(values = c("blue", "red")) +  # Color for feedback type 1 and -1 respectively
  theme_minimal()

combined_plot <- grid.arrange(p, p_right, ncol = 2)

sums_1 <- numeric(983)# Initialize a vector to store the row sums for feedback_type == -1
sums_minus_1 <- numeric(983)

# Loop through each element in session[[8]]$spks
for (i in seq_along(session[[13]]$spks)) {
  # Check feedback_type for the current trial
  feedback_type <- session[[13]]$feedback_type[i]

  # Add each row to the sums vector based on feedback_type
  if (feedback_type == 1) {
    sums_1 <- sums_1 + rowSums(session[[13]]$spks[[i]])
  } else if (feedback_type == -1) {
    sums_minus_1 <- sums_minus_1 + rowSums(session[[13]]$spks[[i]])
  }
}

# Convert sums to data frames
result_df_1 <- as.data.frame(t(sums_1))
result_df_minus_1 <- as.data.frame(t(sums_minus_1))

# Add row names as a column
result_df_1$row <- rownames(result_df_1)
result_df_minus_1$row <- rownames(result_df_minus_1)

# Convert to long format
result_long_1 <- pivot_longer(result_df_1, -row, names_to = "Row", values_to = "Sum")
result_long_minus_1 <- pivot_longer(result_df_minus_1, -row, names_to = "Row", values_to = "Sum")

# Extract brain_area from session[[13]]
brain_area <- session[[13]]$brain_area

# Add brain_area column to result_long
result_long_1$brain_area <- brain_area
result_long_minus_1$brain_area <- brain_area

# Sum the entries from the "Sum" column grouped by "brain_area" for feedback_type == 1
sum_by_brain_area_1 <- result_long_1 %>%
  group_by(brain_area) %>%
  summarize(total_sum = sum(Sum))

# Sum the entries from the "Sum" column grouped by "brain_area" for feedback_type == -1
sum_by_brain_area_minus_1 <- result_long_minus_1 %>%
  group_by(brain_area) %>%
  summarize(total_sum_minus_1 = sum(Sum))

# Combine data frames
merged_data <- merge(sum_by_brain_area_1, sum_by_brain_area_minus_1, by = "brain_area", all = TRUE)

# Replace missing values with 0
merged_data[is.na(merged_data)] <- 0

# Set the maximum value for y-axis
max_y <- max(max(merged_data$total_sum), max(merged_data$total_sum_minus_1))

# Create bar plot for feedback_type == 1 with rotated 90 degrees
bar_plot_1 <- ggplot(merged_data, aes(x = reorder(brain_area, -total_sum), y = total_sum, fill = brain_area)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  ylim(0, max_y) +
  labs(title = "Session #13: Feedback Type 1",
       y = "Sum",
       x = "Brain_Area") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_fill_manual(values = rainbow(length(unique(merged_data$brain_area))))

# Create bar plot for feedback_type == -1 with rotated 90 degrees
bar_plot_minus_1 <- ggplot(merged_data, aes(x = reorder(brain_area, -total_sum_minus_1), y = total_sum_minus_1, f
ill = brain_area)) +
  geom_bar(stat = "identity") +
  coord_flip() +
```

```
  ylim(0, max_y) +
  labs(title = "Session #13: Feedback Type -1",
       y = "Sum",
       x = "Brain Area") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_fill_manual(values = rainbow(length(unique(merged_data$brain_area))))

# Arrange plots in a single window
combined_plot <- grid.arrange(bar_plot_1, bar_plot_minus_1, nrow = 1)
average_spikes_per_trial <- list()
for (i in 1:18) {
  session_average_spikes <- numeric()
  for (j in 1:length(session[[i]]$spks)) {
    trial_average_spikes <- sum(session[[i]]$spks[[j]]) / length(session[[i]]$brain_area)
    session_average_spikes <- c(session_average_spikes, trial_average_spikes)
  }
  average_spikes_per_trial[[i]] <- session_average_spikes
}
within_session_combined_data$Average_Spikes <- unlist(average_spikes_per_trial)
within_session_combined_data$Feedback_Type <- ifelse(within_session_combined_data$Feedback_Type =="Success", 1,
0)
within_session_combined_data$Feedback_Type <- as.factor(within_session_combined_data$Feedback_Type)
# Filter for sessions 8 to 18
within_session_combined_data <- within_session_combined_data %>%
  filter(Session_Number >= 8 & Session_Number <= 18)
within_session_combined_data$Date_Exp <- NULL
within_session_combined_data$Contrast_Right <- NULL
within_session_combined_data$Number_Time_Bins <- NULL
within_session_combined_data$Unique_Brain_Areas <- NULL
within_session_combined_data <- within_session_combined_data %>%
  filter(Session_Number != 16)

kable(head(within_session_combined_data), format = "html") %>%
  kable_styling(full_width = FALSE)  # Adjust styling as needed
set.seed(23434)
#splitting the data between training and testing sets to avoid overfitting.
total_rows <- nrow(within_session_combined_data)
train_rows <- round(0.7 * total_rows)
test_rows <- total_rows - train_rows

trainIndex <- sample(total_rows, train_rows)
dataTrain <- within_session_combined_data[trainIndex,]
dataTest <- within_session_combined_data[-trainIndex,]

#Running a Logistic Regression Model on the Training Data
logistic_regression_model <- glm(Feedback_Type ~ Contrast_Left + Mouse_Name + Average_Spikes,
              data = dataTrain ,
              family = binomial)

set.seed(23434)

#Predicting Test Set Results: (REPEAT FOR THE TESTING DATA)
predictions_logistic <- predict(logistic_regression_model, newdata = dataTest, type = "response")

#Confusion Matrix at 60 percent Cut-Off Probability:
classify60 <- ifelse(predictions_logistic > 0.60, 1, 0)
conf_matrix <- table(Prediction = classify60, Actual = dataTest$Feedback_Type)

# Convert confusion matrix to data frame
conf_matrix_df <- as.data.frame.matrix(conf_matrix)

# Rename columns for better visualization
colnames(conf_matrix_df) <- c("Actual_0", "Actual_1")

# Add a new column for the predicted classes
conf_matrix_df$Prediction <- rownames(conf_matrix_df)

# Melt the data for visualization
conf_matrix_melted <- melt(conf_matrix_df, id.vars = "Prediction")

# Plot the confusion matrix
ggplot(conf_matrix_melted, aes(x = Prediction, y = variable, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = value)) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrix",
       x = "Actual",
       y = "Predicted",
```

```
       fill = "Count")

confusionMatrix(conf_matrix)

# function to print confusion matrices for diffrent cut-off levels of probability
CmFn <- function(cutoff) {

       # predicting the test set results
         predictions_logistic <- predict(logistic_regression_model, dataTest, type = "response")
         C1 <- ifelse(predictions_logistic > cutoff, "1", "0")
         C2 <- dataTest$Feedback_Type
         predY   <- as.factor(C1)
         actualY <- as.factor(C2)

      predY <- ordered(predY, levels = c("1", "0"))
      actualY <- ordered(actualY, levels = c("1", "0"))

        # use the confusionMatrix from the caret package
        cm1 <-confusionMatrix(table(predY,actualY))
        # extracting accuracy
        Accuracy <- cm1$overall[1]
        # extracting sensitivity
          Sensitivity <- cm1$byClass[1]
        # extracting specificity
          Specificity <- cm1$byClass[2]
      # extracting value of kappa
          Kappa <- cm1$overall[2]

        # combined table
          tab <- cbind(Accuracy,Sensitivity,Specificity,Kappa)
        return(tab)}
     # making sequence of cut-off probabilities
        cutoff1 <- seq( .1, .9, by = .05 )
     # loop using "lapply"
        tab2    <- lapply(cutoff1, CmFn)
     # extra coding for saving table as desired format
        tab3 <- rbind(tab2[[1]],tab2[[2]],tab2[[3]],tab2[[4]],tab2[[5]],tab2[[6]],tab2[[7]],
                 tab2[[8]],tab2[[9]],tab2[[10]],tab2[[11]],tab2[[12]],tab2[[13]],tab2[[14]],
                 tab2[[15]],tab2[[16]],tab2[[17]])
        # printing the table
        tab4 <- as.data.frame(tab3)
        tab5 <- cbind(cutoff1,tab4$Accuracy,tab4$Sensitivity,tab4$Specificity,tab4$Kappa)
        tab6 <- as.data.frame(tab5)
        tab7 <- rename(tab6,cutoff =  cutoff1, Accuracy = V2 ,
                       Senstivity = V3 ,Specificity =  V4 ,kappa = V5)
      print(kable(tab7))
#Plot a ROC Curve:
# loading the package
library(ROCR)

PredLR <- predict(logistic_regression_model, dataTest,type = "response")
lgPredObj <- prediction((1-PredLR),dataTest$Feedback_Type)
lgPerfObj <- performance(lgPredObj, "tpr","fpr")
# plotting ROC curve
plot(lgPerfObj,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")

#AUC (Area Under the Curve)
library(ROCR)
# area under curve:
aucLR <- performance(lgPredObj, measure = "auc")
aucLR <- aucLR@y.values[[1]]  #0.3869667

predictions_logistic <- predict(logistic_regression_model, newdata = dataTest, type = "response")

#Confusion Matrix at 60 percent Cut-Off Probability:
classify55 <- ifelse(predictions_logistic > 0.55, 1, 0)
conf_matrix <- table(Prediction = classify55, Actual = dataTest$Feedback_Type)

confusionMatrix(conf_matrix)
# Convert confusion matrix to data frame
conf_matrix_df <- as.data.frame.matrix(conf_matrix)

# Rename columns for better visualization
colnames(conf_matrix_df) <- c("Actual_0", "Actual_1")

# Add a new column for the predicted classes
conf_matrix_df$Prediction <- rownames(conf_matrix_df)
```

```
# Melt the data for visualization
conf_matrix_melted <- melt(conf_matrix_df, id.vars = "Prediction")

# Plot the confusion matrix
ggplot(conf_matrix_melted, aes(x = Prediction, y = variable, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = value)) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrix",
       x = "Actual",
       y = "Predicted",
       fill = "Count")
set.seed(222)
random_forest_model<- randomForest(Feedback_Type ~ Contrast_Left + Mouse_Name + Average_Spikes, data = dataTrain)
random_forest_model

prediction_random_forest <- predict(random_forest_model, dataTest)
dataTest$prediction_random_forest = prediction_random_forest

confusion_matrix <- confusionMatrix(dataTest$Feedback_Type,prediction_random_forest)
confusion_matrix

# Extract confusion matrix values
cm_values <- as.data.frame(as.matrix(confusion_matrix$table))

# Plot confusion matrix as a heatmap
ggplot(cm_values, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  theme_minimal() +
  labs(title = "Confusion Matrix",
       x = "Reference",
       y = "Prediction")
plot(random_forest_model) #Can't improve the error after around 200 trees.
# Define specific columns you want to include in the model
selected_columns <- dataTrain[, -which(names(dataTrain) == "Feedback_Type")]

t <- tuneRF(selected_columns, dataTrain$Feedback_Type,
       stepFactor = 0.5,
       plot = TRUE, ntreeTry = 200, trace = TRUE, improve = 0.05) #got from the graph


random_forest_model_refined<- randomForest(Feedback_Type ~ Contrast_Left + Mouse_Name + Average_Spikes, data = da
taTrain, ntree = 200, mtry = 2, importance = TRUE, proximity = TRUE)

prediction_random_forest_refined <- predict(random_forest_model_refined, dataTest)
confusionMatrix(prediction_random_forest_refined,dataTest$Feedback_Type)

# Generate confusion matrix
conf_mat <- confusionMatrix(prediction_random_forest_refined, dataTest$Feedback_Type)

# Convert confusion matrix to data frame
conf_mat_df <- as.data.frame(conf_mat$table)
# Plot confusion matrix with blue gradient using ggplot2
ggplot(data = conf_mat_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  labs(x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        legend.title = element_blank(),
        legend.text = element_text(size = 12))

#Evaluate variable importance
importance(random_forest_model_refined)
varImpPlot(random_forest_model_refined)
#Determine number of classes to use:

# Run algorithms using 10-fold cross validation
trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"

set.seed(10)
fit.knn <- train(Feedback_Type ~ Contrast_Left + Mouse_Name + Average_Spikes, data=dataTrain, method="knn",
                 metric=metric ,trControl=trainControl)
knn.k1 <- fit.knn$bestTune # keep this Initial k for testing with knn() function in next section
```

```
print(fit.knn)
plot(fit.knn)

set.seed(7)
prediction <- predict(fit.knn, newdata = dataTest)
cf <- confusionMatrix(prediction, dataTest$Feedback_Type)
print(cf)
# Convert confusion matrix to data frame
conf_mat_df <- as.data.frame(cf$table)
# Plot confusion matrix with blue gradient using ggplot2
ggplot(data = conf_mat_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  labs(x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        legend.title = element_blank(),
        legend.text = element_text(size = 12))
test=list()
for(i in 1:2){
  test[[i]]=readRDS(paste('testdata/test',i,'.rds',sep=''))
  print(test[[i]]$mouse_name)
  print(test[[i]]$date_exp)
}
within_test_combined_data <- data.frame(
  Session_number = numeric(),
  Trial_Number = numeric(),
  Mouse_Name = character(),
  Date_Exp = character(),
  Contrast_Left = numeric(),
  Contrast_Right = numeric(),
  Feedback_Type = numeric(),
  Number_Neurons = numeric(),
  Number_Time_Bins = numeric(),
  Unique_Brain_Areas = numeric(),
  stringsAsFactors = FALSE
)

for (i in 1:2) {
  test_data <- test[[i]]
  feedback_label <- ifelse(test_data$feedback_type == 1, "Success", "Failure")

  within_test_combined_data <- rbind(
    within_test_combined_data,
    data.frame(
      Session_Number = i,
      Trial_Number = seq_along(test_data$contrast_left),
      Mouse_Name = test_data$mouse_name,
      Date_Exp = test_data$date_exp,
      Contrast_Left = test_data$contrast_left,
      Contrast_Right = test_data$contrast_right,
      Feedback_Type =  feedback_label,
      Number_Neurons = length(test_data$brain_area),
      Number_Time_Bins = ncol(test_data$spks[[1]]),
      Unique_Brain_Areas = length(unique(test_data$brain_area))
    )
  )
}
rownames(within_test_combined_data) <- NULL
kable(head(within_test_combined_data, 10), caption = "Combined Data for Each Session with TEST DATA")
within_test_combined_data$Feedback_Type <- ifelse(within_test_combined_data$Feedback_Type =="Success", 1, 0)
within_test_combined_data$Feedback_Type <- as.factor(within_test_combined_data$Feedback_Type)
within_test_combined_data$Date_Exp <- NULL
within_test_combined_data$Contrast_Right <- NULL
within_test_combined_data$Number_Time_Bins <- NULL
within_test_combined_data$Unique_Brain_Areas <- NULL



average_spikes_per_trial <- list()
for (i in 1:2) {
  test_average_spikes <- numeric()
  for (j in 1:length(test[[i]]$spks)) {
    trial_average_spikes <- sum(test[[i]]$spks[[j]]) / length(test[[i]]$brain_area)
    test_average_spikes <- c(test_average_spikes, trial_average_spikes)
  }
  average_spikes_per_trial[[i]] <- test_average_spikes
```

```
}

within_test_combined_data$Average_Spikes <- unlist(average_spikes_per_trial)

kable(head(within_test_combined_data), format = "html") %>%
  kable_styling(full_width = FALSE)  # Adjust styling as needed
#random_forest_model_refined<- randomForest(Feedback_Type ~ Contrast_Left + Mouse_Name + Average_Spikes, data = d
ataTrain, ntree = 200, mtry = 2, importance = TRUE, proximity = TRUE)

prediction_test <- predict(random_forest_model_refined, within_test_combined_data)
confusionMatrix(prediction_test,within_test_combined_data$Feedback_Type)

# Generate confusion matrix
conf_mat <- confusionMatrix(prediction_test, within_test_combined_data$Feedback_Type)

# Convert confusion matrix to data frame
conf_mat_df <- as.data.frame(conf_mat$table)
# Plot confusion matrix with blue gradient using ggplot2
ggplot(data = conf_mat_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  labs(x = "Actual", y = "Predicted") +
  theme_minimal() +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        legend.title = element_blank(),
        legend.text = element_text(size = 12))
```