



CAPSTONE REPORT

FLAREDOWN APP

Sarah Gates | BrainStation | September 2022

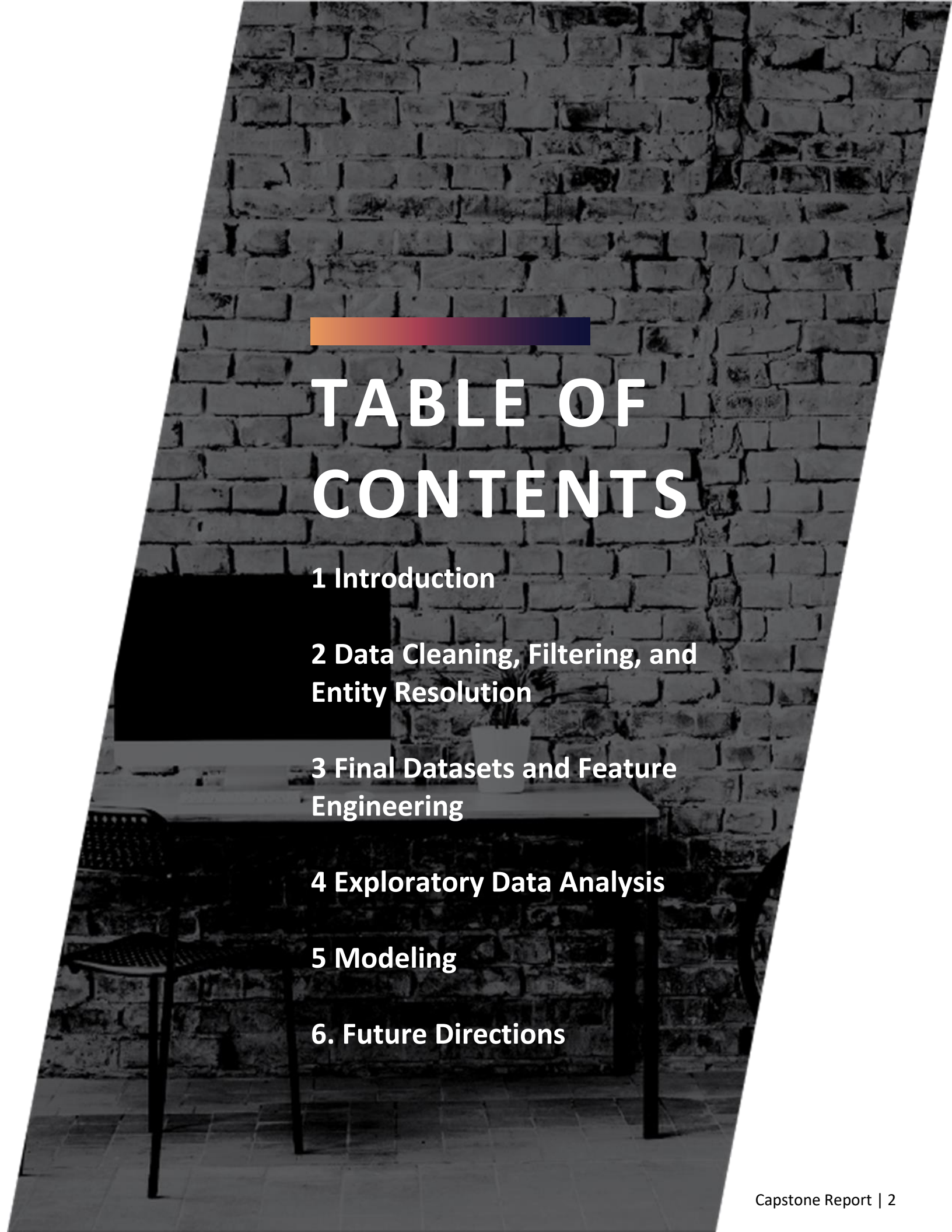


TABLE OF CONTENTS

1 Introduction

2 Data Cleaning, Filtering, and Entity Resolution

3 Final Datasets and Feature Engineering

4 Exploratory Data Analysis

5 Modeling

6. Future Directions



1 INTRODUCTION

CAPSTONE RATIONALE

The impetus for this project stems from my own personal experiences with autoimmune disease. Within the health care space, disease detection is a challenging domain. This is even more challenging for autoimmune diseases, for which the average time to diagnosis is 4.5 years. This time can be as high as 10 years however, which greatly impacts the treatment of disease and the quality of life of patients. Many patients with autoimmune disease receive at least one misdiagnosis, which slows the time to proper treatment and disease maintenance.

The goal of this project is therefore to apply machine learning modeling techniques to better predict autoimmune condition diagnosis in order to reduce the time to diagnosis, and limit misdiagnosis, thereby improving the health outcomes of patients living with these diseases.

From <https://creakyjoints.org/research/getting-diagnosed-with-autoimmune-disease/>

WHAT IS FLAREDOWN?

Flaredown is an app that helps patients with chronic illnesses track their symptoms over time, allowing them to avoid triggers and evaluating their treatments. Patients can track their symptom severity, treatments and doses, and any potential environmental triggers (foods, stress, allergens, etc.) they encounter.

DATA ACQUISITION, FORMAT, AND QUALITY

The data is freely available on the Flaredown App Kaggle page (found here: <https://www.kaggle.com/datasets/flaredown/flaredown-autoimmune-symptom-tracker>). The original csv file contains eight columns: four user level / biographical (user id, age, gender, country), and four tracking related variables: check-in date, trackable type (symptom, condition), trackable name (e.g., fever), trackable value (number indicating symptom intensity). The data contains 7,976,223 rows. Other important user level information, such as weight and height, is not included in the dataset as it is not recorded in the app.



Data Cleaning

- Removed duplicated rows
- Managed null values



Filtering

- Kept users with at least one condition and one symptom logged



Entity Resolution

- Manually resolved the trackable name column
- Reclassified symptom and condition types accordingly

2 DATA CLEANING, FILTERING, AND ENTITY RESOLUTION

DATA CLEANING

My data cleaning process primarily involved removed duplicate rows, and dealing with nulls on a case-by-case basis. The most impactful decision regarding nulls was cleaning the age column where I filled the missing values (3.9% of total) with the median age (34). This created a noticeable peak at 34 years old. I also performed substantial filtering and manual entity resolution, discussed below.

FILTERING

I created a custom filter function to only include users who had at least one symptom and one condition logged. This filter was used multi times throughout the project to ensure adequate data throughout the project.

ENTITY RESOLUTION

Manual entity resolution took up the bulk of the project time, but it was crucial for capturing enough data for proper modeling. As users were able to manually input their symptoms and conditions as strings, the trackable name column was nearly unusable. For example, in addition to spelling mistakes, conditions would be listed under different names (e.g., IBS, irritable bowel syndrome, irritable bowel, etc.). I chose names to resolve in an iterative process, examining the most common conditions and symptoms overall, and then adding to this list as I progressed through EDA and modeling.

I created a custom function to assist with entity resolution. This function, called `entity_resolver`, would take in two lists, a dataframe and a string. The first list was the list of the names matching a given substring to resolve, and the second a list of names to exclude from the resolution process (selected using manual inspection). The words-to-be-resolved were then replaced with a string the user provided, which were then written to the dataframe provided by the user. I then manually re-classified symptoms and conditions accordingly (e.g., headache as symptom).

2

Binary Classification Dataset

- Fibromyalgia classification
- Fibromyalgia (1), Other (0)



Multiclassification Dataset

- Classification of Postural Orthostatic Tachycardia Syndrome (POTS), Lupus, or Autoimmune Arthritis
- Arthritis category included rheumatoid, reactive, psoriatic, and enteropathic arthritis, as well as ankylosing spondylitis



Feature Engineering

- Biographical features: age, gender, country
- Symptom / Condition features: binary (has/ doesn't have), disease activity
- App usage features: total logs, logging rate, usage time

3 FINAL DATASETS AND FEATURE ENGINEERING

FINAL DATASET CREATION

Through the process of cleaning, filtering, and entity resolution, I examined the frequency of occurrence of various conditions in the dataset. The most practical approach appeared to be to select the most common occurring condition in order to construct a binary classification problem. This condition was fibromyalgia. The multiclassification dataset was selected from conditions that were of near equal size and didn't have too many overlaps between them (POTS, Lupus, Arthritis). The final datasets were created by examining overlapping conditions and symptoms between users with the target classes. I used a 3% cut off for symptoms and conditions to consider. This resulted in a small number of non-overlapping symptoms and conditions between users in the target classes.

FEATURE ENGINEERING

As I wanted to attempt a static (i.e., non-temporal) classification problem, I needed to collapse (in some cases) years-long data into a single row in the dataframe such that each row contained each users' data. As such, I wanted to maintain some of the time-component in the data where possible. To accomplish this task, I created three classes of features:

- ❖ Biographical information:
 - Age
 - Country
 - Gender (female, male, other)
- ❖ Disease Information:
 - Condition / Symptom Binary (Yes/No)
 - Disease Activity (median trackable value, 0-4)
- ❖ App / Time Usage: Total and per Condition
 - Total logs
 - Unique dates
 - Logging rate (total logs / dates)

The app / time usage features were constructed globally (i.e., aggregates of all user information), and also calculated for each symptom and condition in the dataset. This resulted in five features per symptom or condition indicated by the user: Binary, Activity, Total Logs, Unique Dates, and Logging Rate.

4 EXPLORATORY DATA ANALYSIS

For the exploratory analysis, I examined variables in each classification dataset—binary fibromyalgia classification, and autoimmune multiclass—separately, split across the target classes. This was done so that I could better understand the potential impact of each variable (e.g., back pain) in the context of the newly created features (e.g., back pain total logs). I discovered that some of the expected variables appear to be predictive of fibromyalgia in the binary classification dataset, such as IBS, and brain fog (see FIGURE 1). Within the multiclass EDA, I discovered that several variables appear to be highly predictive of POTS, while lupus and autoimmune arthritis conditions variables appear quite similar across variables (see FIGURE 2). This suggested to me that distinguishing between lupus and arthritis would be a challenge for the model, which it was (see 5 Modeling).

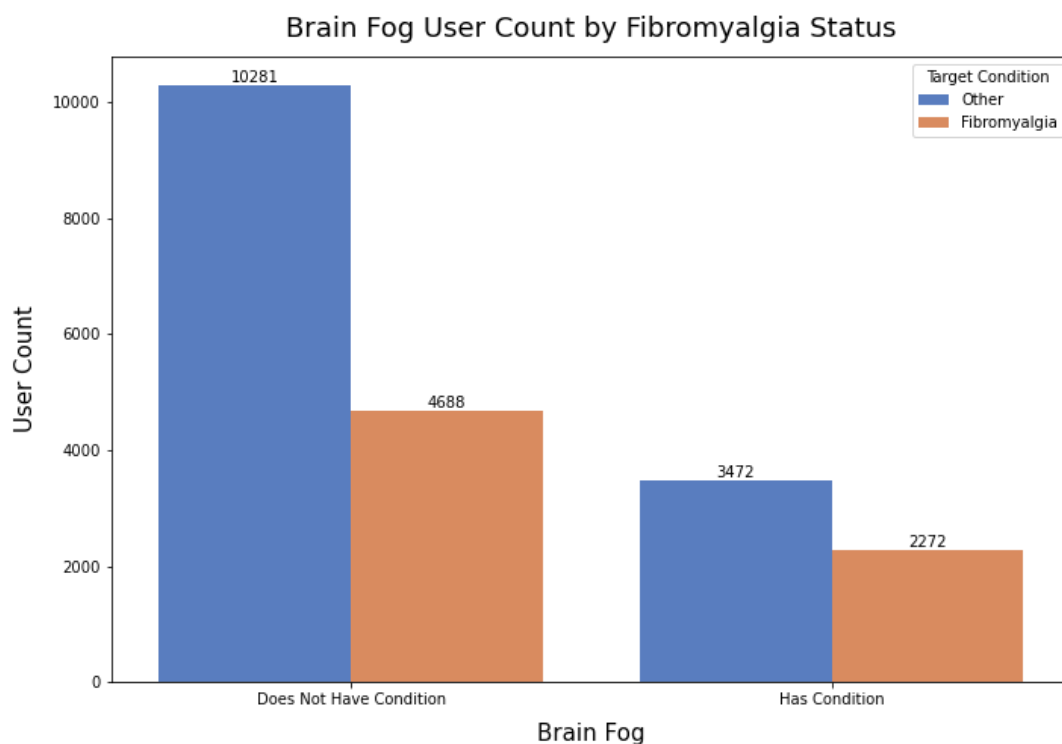


Figure 1. Brain Fog Binary by Fibromyalgia Status.

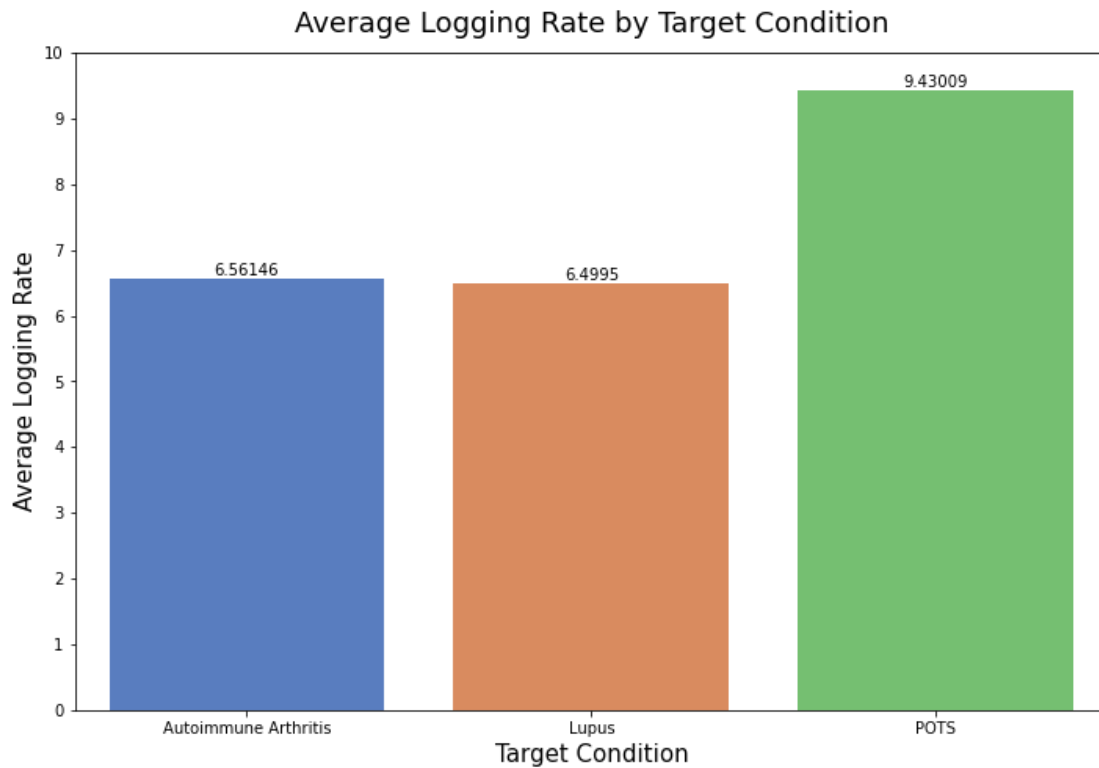


Figure 2. Average Logging Rate by Multiclass Target.



5 MODELING

The approach to modeling was to examine each of the variable subsets—binary, total logs, unique dates, logging rate and activity—separately so that their impact on classification could be evaluated. I used three types of models—decision tree, logistic regression, and KNN—on each variable set, and then selected the model and set that had the best performance for further optimization via grid search and application to the remainder-test split. For both classification problems, a logistic regression model provided the best fit. Low recall scores were consistently an issue (see FIGURE 3) in both datasets. As predicted from the EDA, POTS was easy for the model to distinguish, while lupus and autoimmune arthritis were more difficult and often confused for one another (see FIGURE 4).

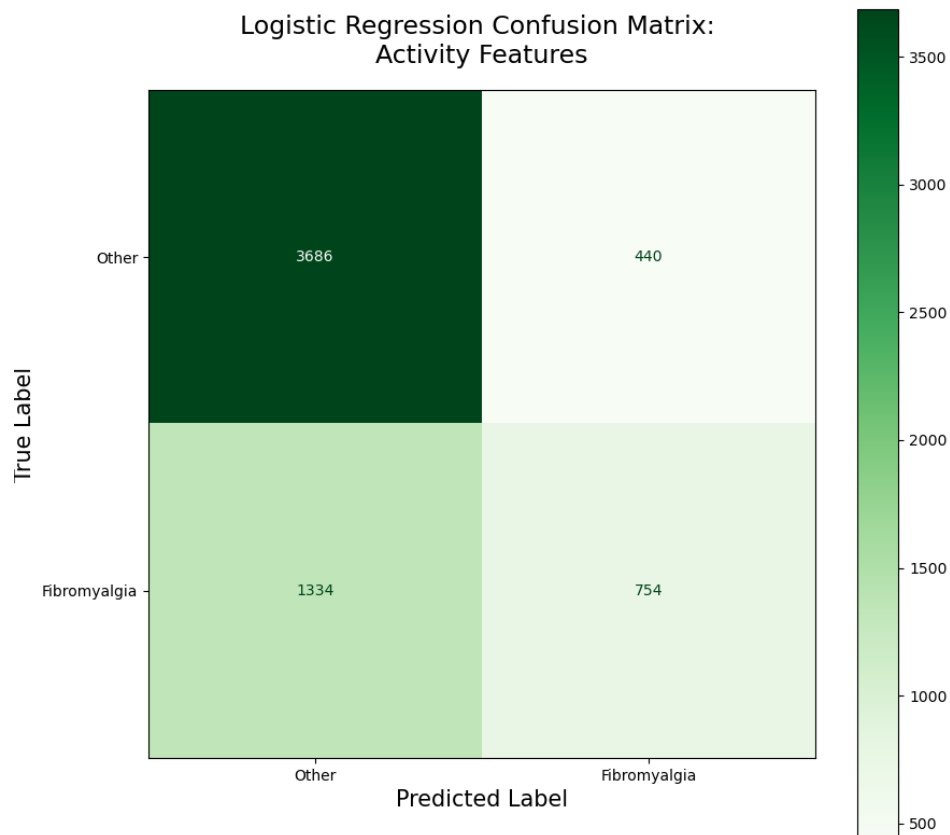


Figure 3. Binary Classification Confusion Matrix.

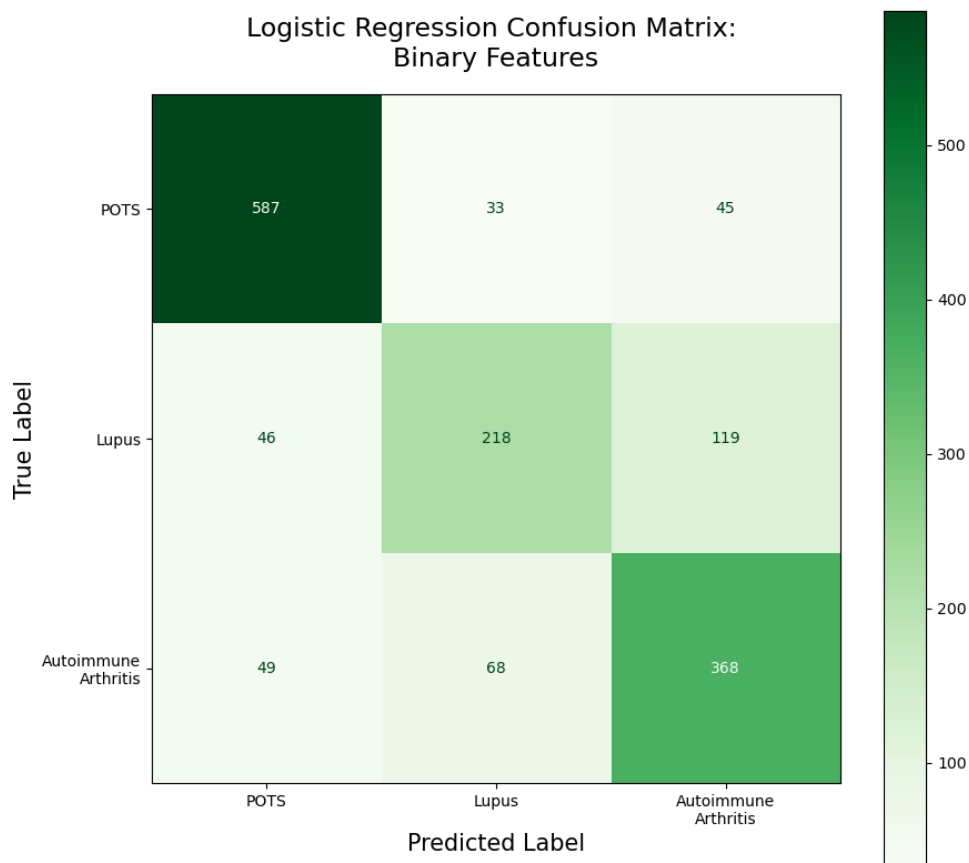


Figure 4. Multiclassification Confusion Matrix.



6 CONCLUSIONS AND FUTURE DIRECTIONS

The most interesting take aways from this project is the finding that all of the models had poor recall scores, which emulates the real-world difficulty in autoimmune diagnosis. In particular, it mimics the fact that most patients with an autoimmune condition receive at least one misdiagnosis in their lifetimes. This demonstrates that this domain is difficult for both human doctors and machine learning algorithms.

The next step for the multiclassificaton modeling is to expand out to overlapping classes (multilabel classification) such that target conditions could be inclusive (as many patients have multiple, overlapping conditions). Future steps for this project also include developing a web app (e.g., streamlit) for application of the findings to support diagnostics. The primary area of application for these models is to support health care workers in assessing and considering diagnosis options for conditions that are often missed. If implemented, this could help to reduce the time to diagnosis of autoimmune diseases.



BrainStation[®]