**SELEC v0**

Team: Bella Wu, Karen Li, Mark Bertolami, Rose Lee, Shuyan Zhao

**User Story 1:**

The battery electrochemist working in an R&D lab on the bench scale, for example, the senior scientist in a national lab or the severely underpaid graduate student in a university lab, wants to know which electrolyte will have the best performance for the battery system they are working with. The user, who may be unfamiliar with programming languages, will select their battery cathode, anode, and C-rate from a drop-down menu on a user-friendly interface. The user may also want to know the predicted performance of alternative choices, so both the predicted performance of the best electrolyte and the alternative choices will be *beautifully* visualized on the UI.

**User Story 2:**

The SELEC developers want to create a machine learning model that can predict the effect of a battery’s performance based on the electrolyte. The developers have experience with python and want to train a machine learning algorithm to predict the performance of a battery based on a dataset with experimental results. The developers also want to include an option to add new data into the dataset as more data is collected by scientists and include an option to add more parameters to describe the batteries.

**Use Case 1: Train machine learning algorithm**

**Description:** The user, a developer with experience in python, wants to train a machine learning model (MLM) to predict battery performance.

**Inputs:** The user provides the model with a battery dataset that includes components and performance results.

**Outputs:** A machine learning algorithm that can predict how well a battery (with a component combination not seen by the MLA previously) will perform depending on the electrolyte used.

**Components:** A supervised machine learning algorithm (i.e. KNN, Random forest decision trees, support vector regression method) will be trained with a dataset containing battery components (anode, cathode). The algorithm will also calculate a metric of error (i.e. mean squared error, r2 score) for each test sample.

**Use Case 2: Add New Data Set**

**Description:** The user, the developer or scientist, wants to provide new battery data into the training set.

**Inputs:** The user uploads a new dataset (file type TBD) formatted in a specific way with specific headers (i.e. Anode Name, Cathode Name, Voltage, etc. Details are TBD).

**Outputs:** The program will add the new data to the existing dataset. A new machine learning model that has been trained with the new dataset.

**Components:** The program will check the data to see if the cathode/anode/electrolyte name exists within the set, and categorizes the data accordingly. If it does not exist, it appends the data into the existing dataset. The model will then be retrained to include the new data.

**Use Case 3: Battery Descriptor**

**Description:** The user, the developer or scientist, wants to include another battery component (i.e. charge rate, discharge rate) or battery environment parameter (i.e. temperature or pressure) to the battery description which includes an anode and cathode.

**Inputs:** The input is the battery dataset. The new input can be included as a new column to the dataset

**Outputs:** An updated interface, an updated battery descriptor, and an updated machine learning model

**Components:** The dataset can include multiple input parameters. Inputs that are not numbers will be classified by a number (i.e. 0,1,2,...) depending on how many types of that input is in the dataset. For example, if there are three anode materials, graphite, Li metal, and Na metal, will be classified by 0,1, and 2 respectively. For inputs that are numbers, its class will be the value of the number. The battery descriptor will be an empty list that will append the class of the input gained from the drop down menus in the user interface. If a new input is to be considered, the machine learning algorithm will be trained with the new input and the battery descriptor list will append the new value as well.

**Use Case 4: Battery Performance Predictor**

**Description:** The user, a battery electrochemist working in an R&D lab on the bench scale with limited python experience, is interested in selecting an electrolyte for a battery with certain components and how that battery will perform.

**Inputs:** The user provides a description of the desired battery(an array) that consists of an anode material, a cathode material, and a charge rate and runs the program.

**Outputs:** MLA returns an electrolyte that gives the best performance and the performance of batteries with alternative electrolytes in a user- friendly interface. An error message will be returned if the MLM can only give poor predictions.

**Components:**

A user interface for the user to input their battery descriptor based on options available in drop down menus. The input is a battery description which will be an array with classifiers of the anode, cathode, c-rate, etc. The user will be able to click an evaluate button to run the MLM. Program will use a visualization package (i.e. Seaborn, Matplotlib, etc) to display the battery performance. The display will return an error message if only poor predictions can be made based on an error metric (i.e. a high MSE). Otherwise the program will select and return the best battery electrolyte based on performance metrics (capacity, open-circuit voltage, and resistance, efficiency, etc). The program will also generate a table/dataframe of alternative electrolytes and their performance results .