**SELEC v0**

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

**User Story 1:**

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.

**Use Case: 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:**

Component 1: Train a supervised machine learning algorithm (i.e. KNN, Random forest decision trees, support vector regression method).

The input is a set of arrays with values from the battery dataset columns “cathode”, “electrolyte”, “cycle”, “temperature”, and “discharge-crate”. This set of arrays will be split between a training set and test set. The training set will be used to train the MLM and the test set will be used to verify how accurate the MLM can predict the output. The outputs are the predicted charge capacity, discharge capacity, charge energy, discharge energy, and coulombic efficiency (integers). The predicted values will be compared to the test values. A loss value will quantify this comparison, which will be a mean squared error, r2 score, accuracy score, and score.

Component 2: Hyper parameters

**User Story 2:**

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.

**Use Case: 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 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:**

Component 1: Graphical User Interface

A streamlit user interface will allow user to input their battery descriptor based on options available in drop down menus. A function will take these inputs and turn it into an array as an output for the predictor. The user will be able to click an evaluate button to run the predictor. The GUI will return an electrolyte selection and a plot with the results of all the predicted batteries. The user can pick which output parameter they want for the interactive plot through a dropdown menu.

Component 2: Predictor

The input is the battery description given by the use from the GUI. A data frame will be created from this first array which will include the electrolyte options. This array is the input for our machine learning model and output is the prediction for charge capacity, discharge capacity, charge energy, discharge energy, coulombic efficiency, and energy efficiency. The program will select and return the best battery electrolyte based on performance metrics (capacity, open-circuit voltage, and resistance, efficiency, etc).

Component 3: Results Generator

Input is results form machine learning model

Output is dataframe, anode, cathode, electrolyte, cycle, temperature, discharge c-rate

Component 4: Visualization Generator

The program will use Matplotlib to generate an interactive 3D plot to display battery performance.

Input is columns from the data frame from result generator

X = cycle Y= battery desciptor+anode Z= output indicated by GUI