**SELEC v0**

Karen Li, Bella Wu, Rose Lee, Shuyan Zhao

Use Cases and Component Specifications

**User Story 1:**

The SELEC developers want to create a machine learning model (MLM) 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 1: Data Management**

**Description:** The user, a developer with experience in python, wants to format a battery data set that includes and cycling data and experiment conditions to

**Inputs:** CSV files that has cycling data and experiment conditions for multiple batteries.

**Outputs:** One CVS file that includes all necessary data needed to train a machine learning regression model.

**Components:**

Component 1: Assemble dataset

Publicly assessable battery cycling data of different commercial batteries provided by Sandia National Laboratory at the Battery Archive repository. The meta data and cycling csv files need to be downloaded. A data frame will be constructed in jupyter notebook to include the information in the metadata file and the battery performance metrics (charge capacity, discharge capacity, charge energy, discharge energy, coulombic efficiency, and energy efficiency) for each battery in cycles 50 to 500, in increments of 50 cycles. The resulting data fame will be exported to a csv file.

**Use Case 2: Train machine learning algorithm**

**Description:** The user, a developer with experience in python, wants to train a 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 will perform depending on the electrolyte used.

**Components:**

Component 1: Train supervised machine learning regressors.

Multiple machine learning regressors (K-NN, random forest, decision trees, support vector regression method, gaussian process) will be tested in their ability to predict the performance of a battery. The input is an encoded battery dataset columns (anode, cathode, electrolyte, cycle, temperature, discharge c-rate, charge capacity, discharge capacity, charge energy, discharge energy, and coulombic efficiency). This set of arrays will be split between a training set and test set. The training set will be used to train MLM’s and the test set will be used to verify how accurate the MLM can predict the output. The outputs are the predicted battery metrics (charge capacity, discharge capacity, charge energy, discharge energy, coulombic efficiency, and energy efficiency). Each of these metrics will train a k-nn regressor separately. The comparison of the predicted values to the test values will be quantified with a loss value, the root mean squared error

Component 2: Hyper parameters

Hyper parameters of the chosen regression model, k-nearest neighbor regression, will be optimized using the sklearn model GridSearchCV. The inputs will be options (range and lists) for the hyperparameters: n\_neighbors, weights, and algorithm. Other inputs will also include the training set as described in the previous component. The output will be a list of the best hyperparameters for each output.

**User Story 2:**

A 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 visualized on the UI.

**Use Case 1: 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:** MLL that returns the performance of batteries with different electrolytes in a user- friendly interface.

**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 interactive plots with the results of all the predicted batteries.

Component 2: Predictor

The input is the battery description list given by the user from the GUI. A data frame will be created from this description list which will include al cycle and electrolyte options. This data frame is the input for our trained k-nn regressor and output is the predictions for each battery metrics in a data series format.

Component 3: Results Generator

The input is a data frame describing the user’s battery and data series of the results of the predictor. The output is a data frame that contains columns describing the battery description (anode, cathode, electrolyte, cycle, temperature, discharge c-rate) and a column for each metric.

Component 4: Visualization Generator

The program will use plotly to generate interactive 3D plots and 2D plots to display predicted battery performance. The x, y, and z axis will take an array created from values in the appropriate data frame column from the Result generator. The x-axis is the electrolyte options, the y axis is the cycles available, and the z axis will be the predicted battery metrics.