**Detailed Report on the NASA Battery Dataset Analysis**

**Introduction:**

The goal of this assignment is to analyse the NASA Battery Dataset, which contains measurements of lithium-ion (Li-ion) batteries subjected to different operational profiles (charge, discharge, and impedance) under varying temperatures. Specifically, we are tasked with visualizing the changes in two key battery parameters, **Battery Impedance** and **Charge Transfer Resistance**, as the battery ages through multiple charge/discharge cycles. These measurements are essential for understanding how the battery's internal parameters evolve as it approaches its end-of-life (EOL) during usage.

**Problem Statement:**

The dataset includes measurements related to the charge/discharge cycles of batteries and impedance data taken via Electrochemical Impedance Spectroscopy (EIS). The goal is to plot the following battery parameters as a function of the aging process:

1. **Battery Impedance (Re)**: Estimated electrolyte resistance (in ohms).
2. **Charge Transfer Resistance (Rct)**: Estimated resistance related to the charge transfer process (in ohms).

These parameters can give insights into how the battery's internal conditions change as it undergoes repeated cycles.

**Dataset Overview:**

The dataset consists of multiple CSV files, with each file containing data for one test cycle of a particular battery. The columns in the metadata.csv file include:

* **type**: The type of operation (charge, discharge, impedance).
* **start\_time**: The timestamp when the measurement started.
* **ambient\_temperature**: The temperature during the test.
* **battery\_id**: Identifier for the battery.
* **test\_id**: Unique identifier for each test.
* **filename**: The file containing the detailed measurements.
* **Capacity**: Battery capacity during the test.
* **Re**: Electrolyte resistance (battery impedance).
* **Rct**: Charge transfer resistance.

**Data Preprocessing and Cleaning:**

**1.** **Loading Data:**

* The dataset is read using pandas, where metadata.csv is loaded first, followed by inspecting the structure of the data.
* The data contains missing values in the **Re** and **Rct** columns. These missing values need to be handled for accurate analysis.

**2.** **Handling Missing Values:**

* The missing values in the **Re** and **Rct** columns are filled using the median value for each column. This approach is often used to minimize the effect of outliers while filling missing data.

**3.** **Data Exploration:**

* We analysed the distribution of **Re** and **Rct** using histograms. Prior to filling missing values, we could observe that both parameters had several missing entries.
* After filling missing values, we confirmed that no missing data remained in these columns.

**4.** **Filtering Data for Charge/Discharge Operations:**

* The data for charge and discharge operations is filtered, and we added a **cycle\_number** column using the **uid** column to represent the cycle number.
* The data is sorted by the **cycle\_number** to ensure correct visualization over time.

**5.** **Impedance Data Filtering:**

* A similar filtering process is applied to the impedance data, and the data is also sorted by cycle number.

**Data Visualization:**

Visualization is done using **Plotly**, a powerful library for creating interactive plots. The following visualizations were created to track the changes in **Re** and **Rct** over time:

1. **Histograms for Re and Rct (Before Filling Missing Values):**
   * The first step in the visualization process is plotting the distribution of **Re** and **Rct** to understand their ranges and the presence of any anomalies in the dataset.
2. **Line Plots for Charge/Discharge Cycles:**
   * Line plots are created for both **Re** and **Rct** over the charge and discharge cycles. Custom colours (blue for charge and red for discharge) and opacity are used for better clarity.
   * These plots help visualize how each parameter changes over the cycles and how the battery’s internal resistance evolves during charge and discharge.
3. **Line Plots for Impedance Cycles:**
   * Separate line plots are created for **Re** and **Rct** during impedance cycles (represented in green). These plots show how the parameters change during impedance measurements.

**Insights and Observations:**

**1. Charge and Discharge Data:**

* Both **Re** and **Rct** remain relatively constant during charge and discharge cycles, but small variations are observable as the battery ages. This suggests that while the battery might appear stable over the short term, there may be subtle changes in internal resistance due to repeated cycling.

**2.** **Impedance Data:**

* The **Re** and **Rct** values in impedance cycles show fluctuations as the impedance measurement is performed. These measurements provide insights into the internal chemical and electrochemical processes occurring as the battery ages.

**3.** **Trends Over Time:**

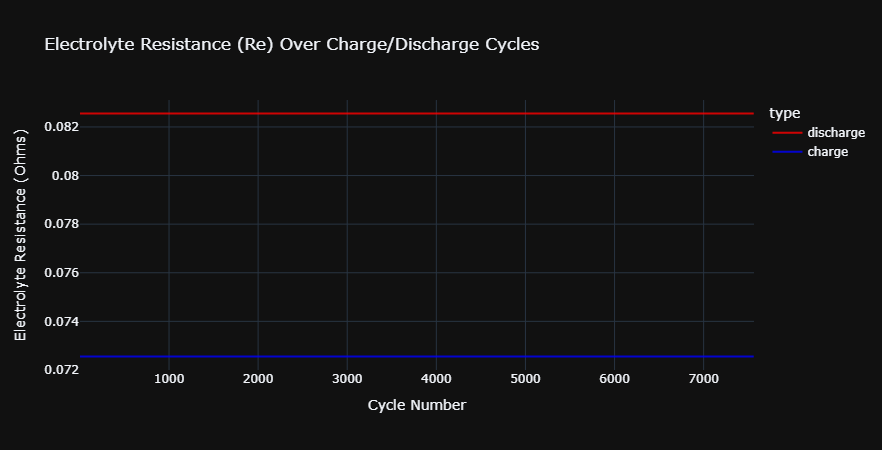
* A consistent upward trend in **Re** and **Rct** would indicate aging and deterioration of the battery over time. If such a trend is not observed in the visualizations, it could imply that the battery is still operating efficiently within the given cycles.

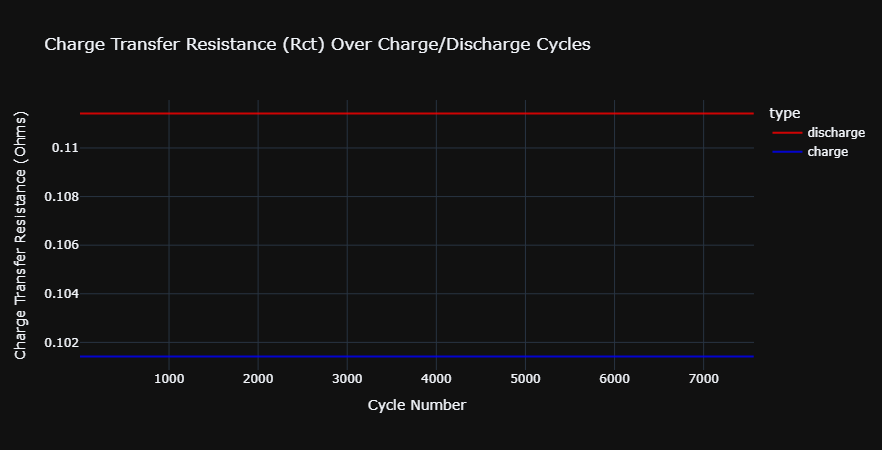
**Conclusion:**

By examining their impedance and resistance, this assignment illustrates how Li-ion batteries can be processed, cleaned, and visualized as they age. By examining the Battery Impedance (Re) and Charge Transfer Resistance (Rct) over time, it is possible to gain insight into the internal degradation of batteries as they age with frequent charge/discharge cycles. Through the use of Plotly, interactive visualizations enable users to track these changes in a fun way, making it simpler to analyse and extract relevant insights from the dataset.

Moreover, by applying this form of data analysis and visualization to other battery datasets, one can obtain more information about battery age mechanisms, which is essential for optimizing battery life in practical applications.

**Graph Over Charge/Discharge Cycles using ploty:**





**Graph Over Impedance Cycles using ploty:**

