Machine Learning on IoT Data: Predicting Battery Health Status and User Type

Mehdi Lebdi - Mirza Tauqeer Baig - Mehdi Nikkhah – Junaid Qazi

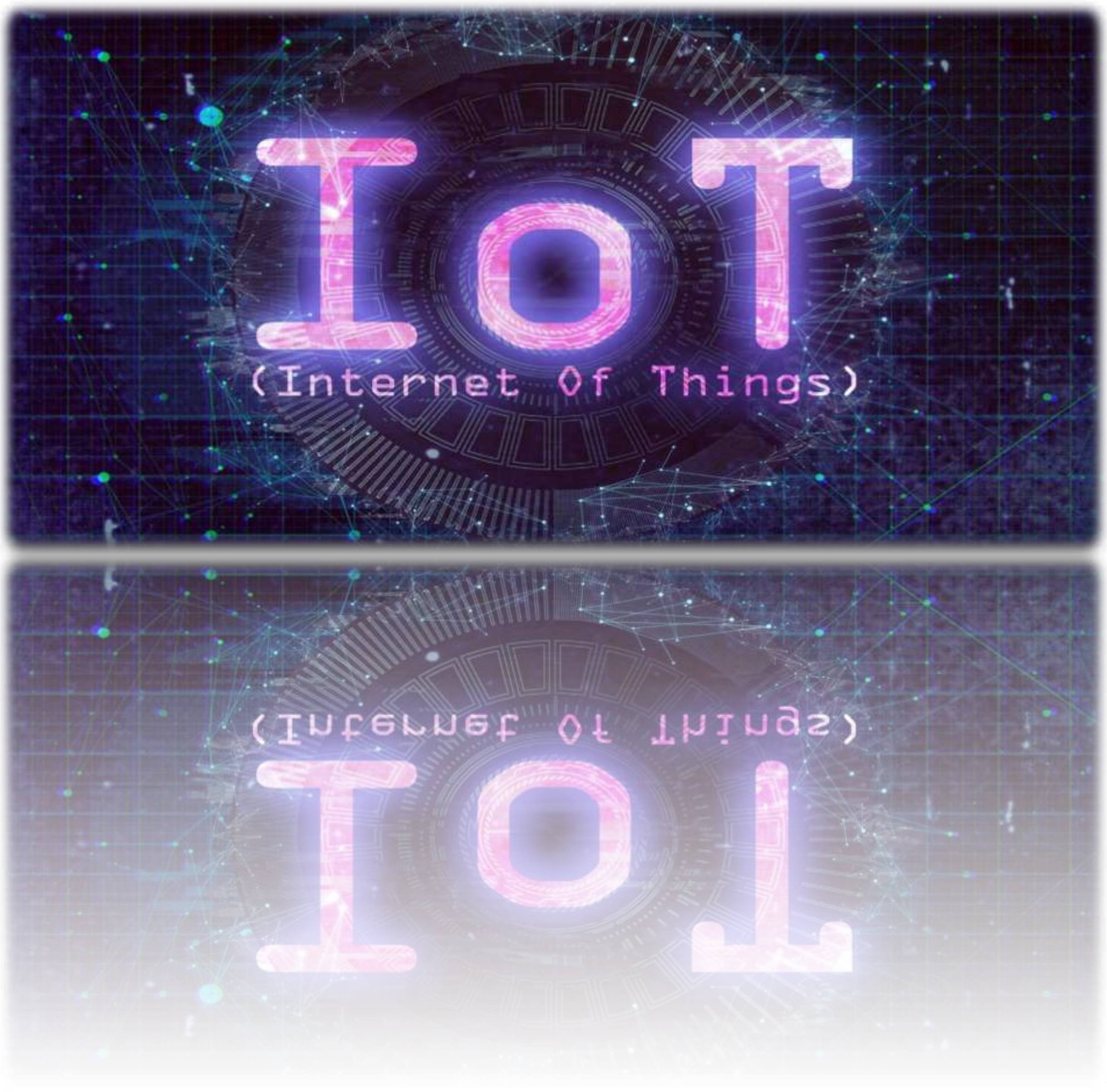


Table of Contents

1 Motivation and Background

With the rise of the IoT devices in the market, the cost incurred on maintenance will be consequential and vast. Manufacturers are adding sensors to the components of their products so that they can transmit back data about how they are performing. This can help companies detect when a component is likely to fail and to replace it before it causes damage. Companies can also use the data generated by these sensors to make their systems and their supply chains more efficient, because they will have much more accurate data about what is really going on.

The benefits of the IoT for business depend on the particular implementation, but the key is that enterprises should have access to more data about their own products and their own internal systems, and a greater ability to make changes as a result.

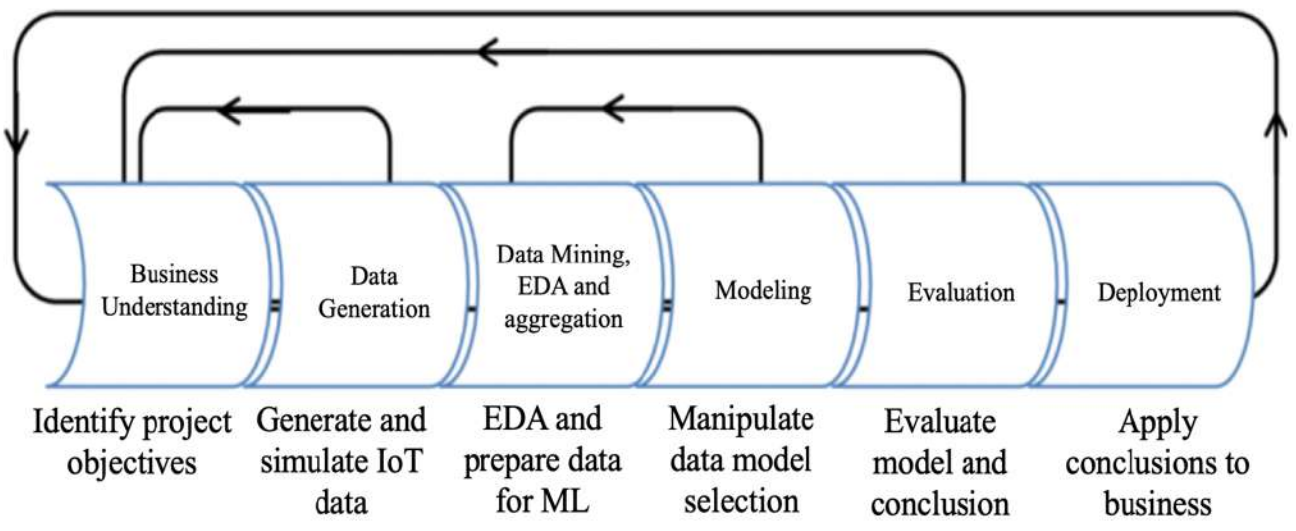
2 Problem Statement

For industrial equipment, more efficient maintenance translates into big savings. It's a combination of longer equipment life and avoiding expensive downtime, which can lead to costly liabilities such as device failure. The objective is to determine whether we can answer these questions using predictive analytics on IoT data:

- Can we build a predictive model that can warn us of those initial glitches or anomalies in IoT that would help a company detect a complication before it turns into a costly liability?
- Can we classify users into categories by analyzing their device activity?
- Can we predict the current health of a device?
- Can we predict the Remaining Useful Life (RUL) of a device?

Companies that want to bring their organization into the fourth industrial revolution need to start with state of-the-art sensors whose technologies can be linked to large amounts of data in almost real time. Sensors need to be able to monitor conditions with a high level of reliability in real time to provide meaningful data. Many sensors are limited by antiquated technology and need to be updated, thus making it hard to transmit large amounts of data.

3 Data Science Pipeline


Business Understanding
Data Generation
Data Mining, EDA and aggregation
Modeling
Evaluation
Deployment
Identify project objectives
Generate and simulate IoT data
EDA and prepare data for ML
Manipulate data model selection
Evaluate model and conclusion
Apply conclusions to business

3.1 Business Understanding

The project revolves around the IoT domain and the business of predictive maintenance. When predictive maintenance is in place, machines on the factory shop floor or appliances in a customer's home can evaluate their own performance. We can make use of algorithms based on big data to predict the type of users adopting these machines and the equipment life span. This will provide a deeper understanding about equipment optimization as well as if and when it needs to be serviced or replaced. For this project, we will use data from device batteries in accordance to the Samsung IoT data guidelines. Therefore, the objective is to determine whether we can categories users as well as predict the current health of a battery.

3.2 Data Generation

We faced challenges in collection of the IoT dataset. Due to privacy concerns, the data used for this project was generated to simulate real world IoT data. One of the most important challenges in big data is finding and collecting good quality data from reliable sources. This project is no exception; The Samsung battery usage dataset was not publicly accessible anymore and the process of getting approval to share that data would take at least two months to get the required licenses. Hence, after meeting with Bandeep Singh, Data Scientist at Samsung Digital Canada, we understood the real data specifications and their requirements. Consequently, we developed a Data Generator based on the latest standards (this part is thoroughly described in the data_generator.ipynb notebook as well as further in the report).

3.3 Exploratory Data Analysis

Exploring Data Analysis on IoT data for user's device behavior.

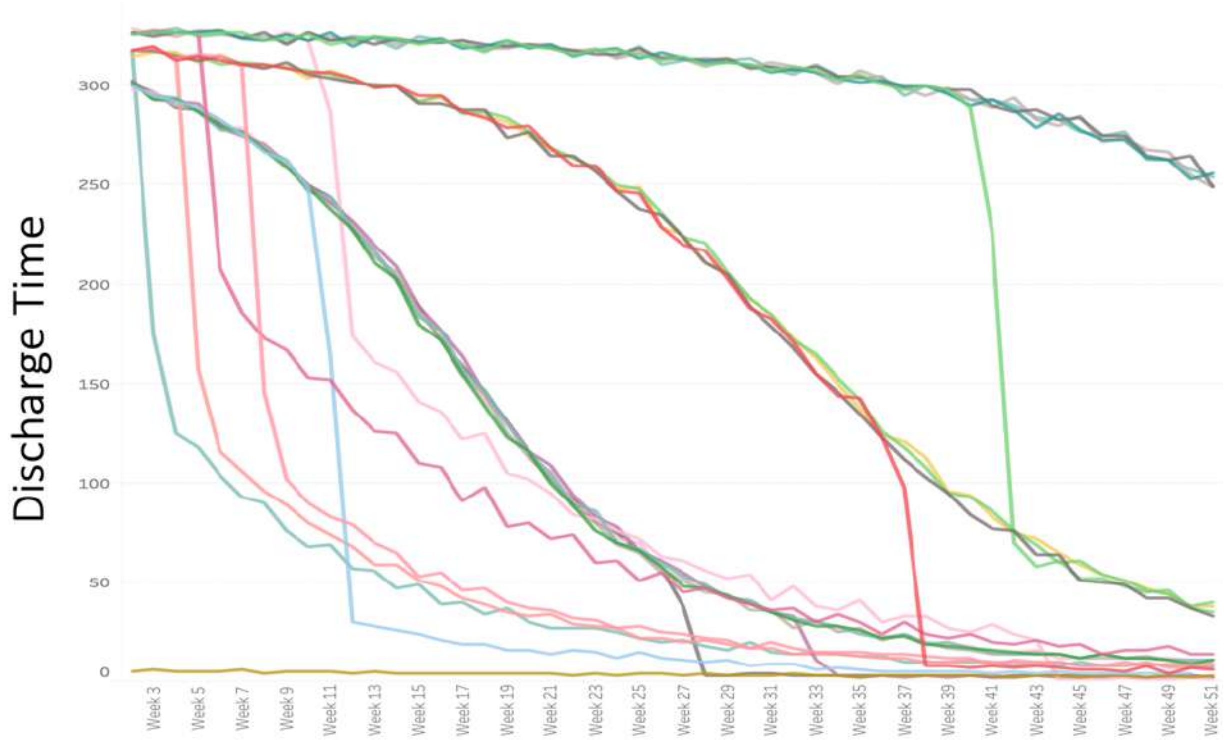


Figure 1: Battery discharging time over the period of 1 year. Plot shows trends of different types of users using same device battery. Sudden drops depict unusual behavior (anomalies/glitches) of the device

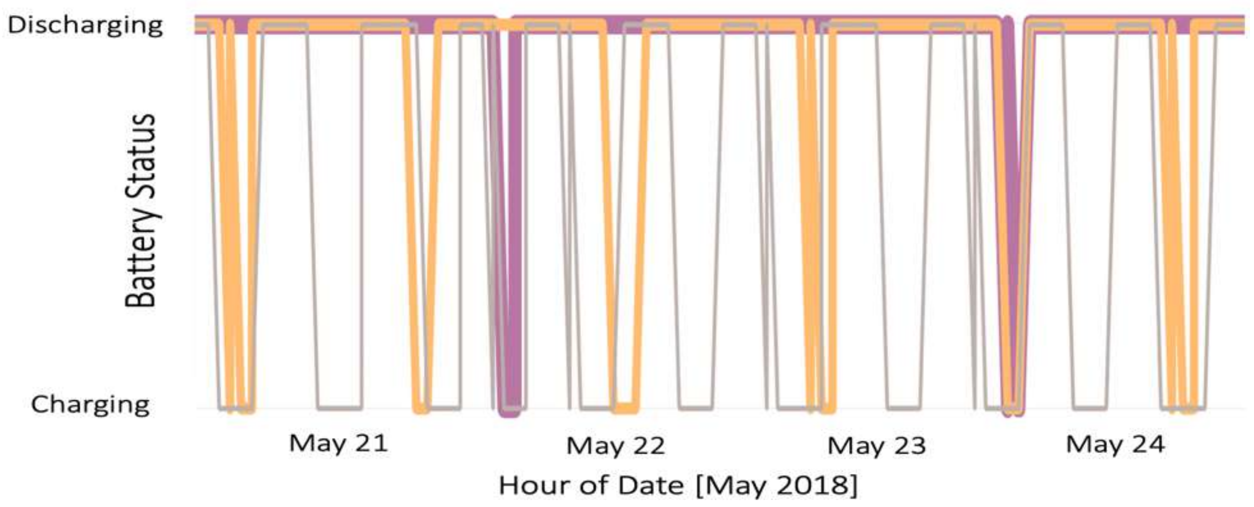


Figure 2: Plot depicts usage pattern for 3 distinctive types of users (low/medium/high activity). Low user (Purple) has experienced 2 charging/discharging cycles a lower frequency compared to other users with higher cycle frequencies

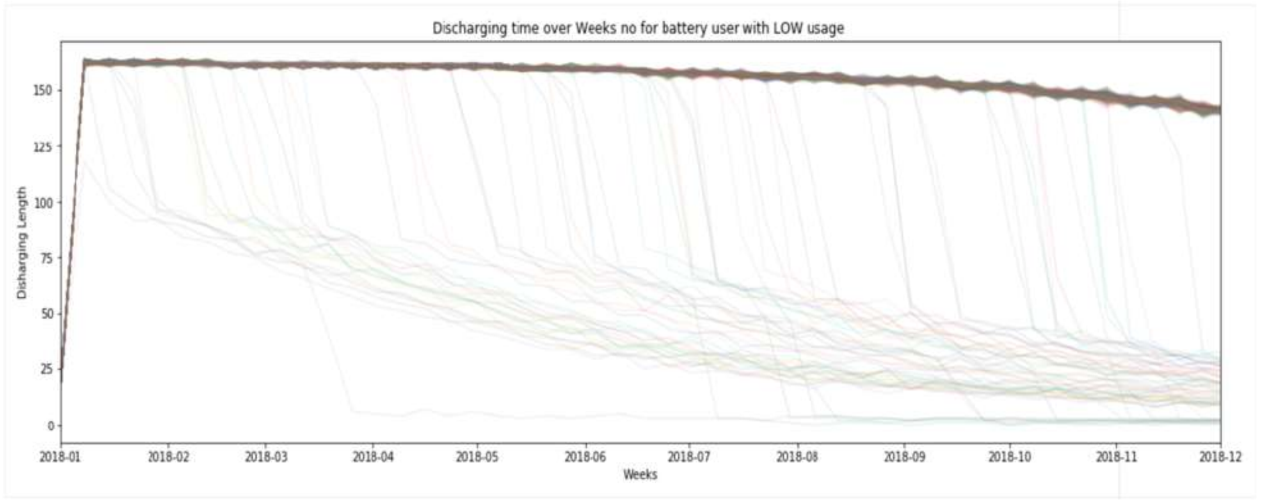


Figure 3: Plot shows a slow discharging rate over the period of 1 year, denoting the battery is still healthy for most Low users

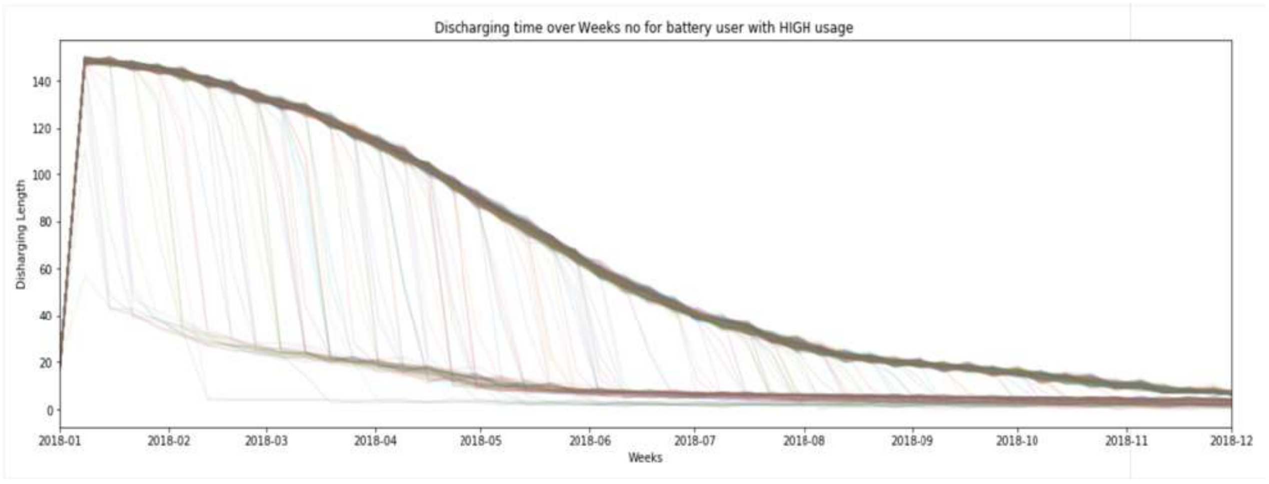


Figure 4: Plot shows a fast discharging rate over the period of 1 year, denoting that the battery health is deteriorating faster for HIGH users

3.4 Modeling

Based on the EDA performed on device activity, the models are applied to the main goal of predicting the user type and the battery health status. The unsupervised and supervised Machine Learning methods, such as **K-Means Clustering**, and **K-Nearest Neighbors (KNN)** are a clustering algorithm and a classification algorithm respectively. These models are implemented to classify users into categories as well as label the health status of batteries.

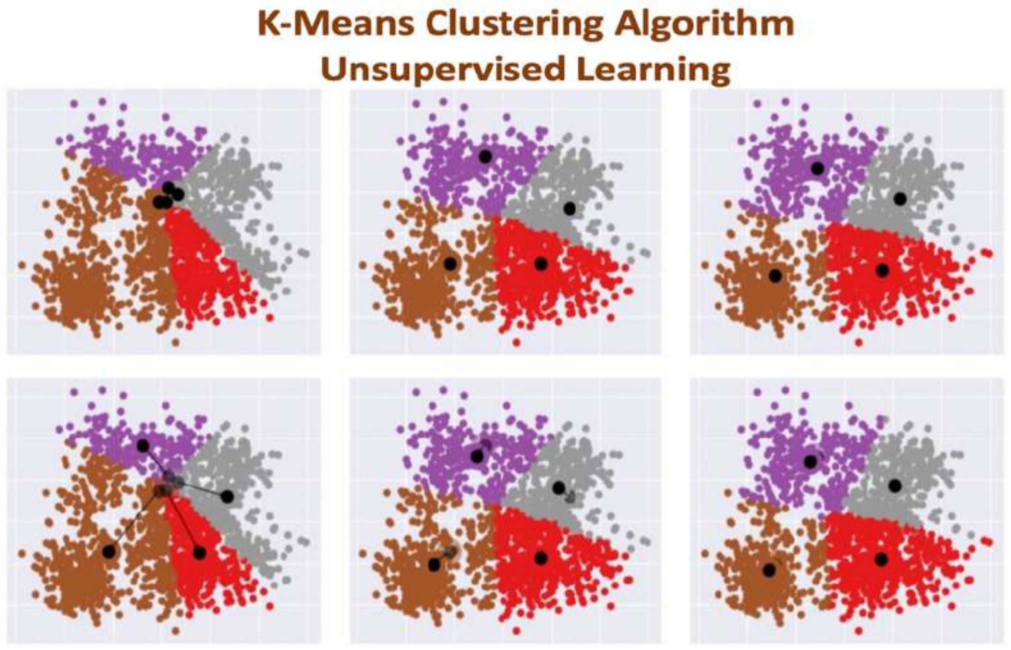


Figure 5: Centroids are initiated in the beginning leading to new cluster centroids in each iteration until converged. Cluster centroids are shown as points in black.

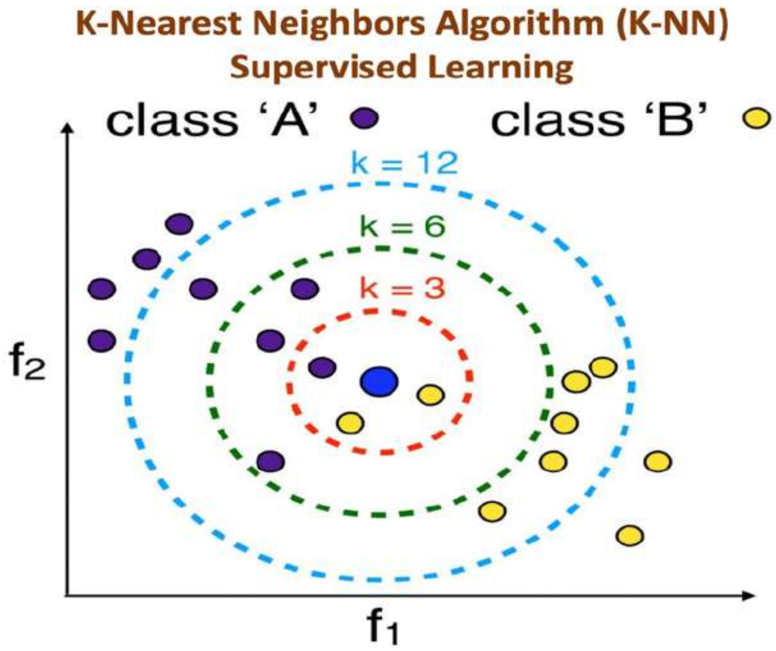


Figure 6: Distinct values of k lead to distinct predictions for the new data points. The optimum value of k is computed using the Elbow method.

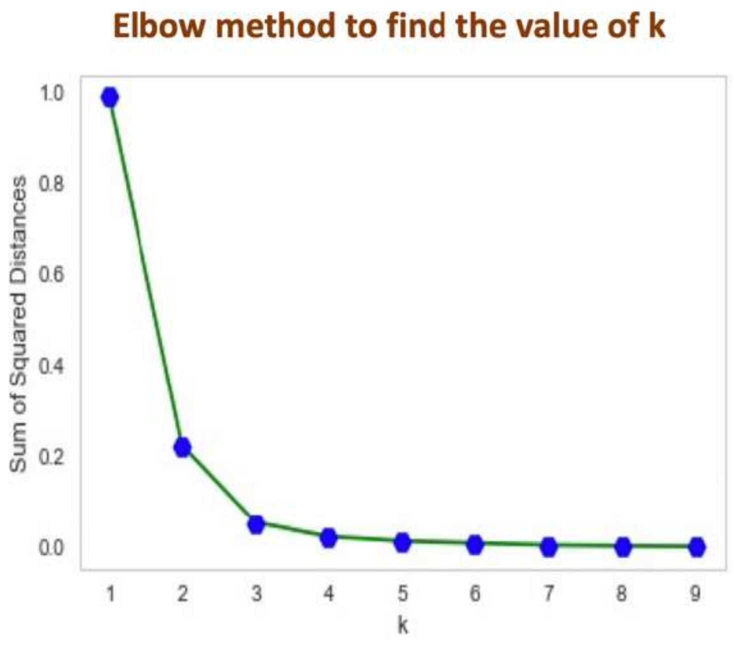


Figure 7: Elbow method to find the value of k for our data points

The below steps are followed for modeling the ML algorithms:

- We read the raw IoT data named final-bat1.csv from data folder (this file is created by data_generator.ipynb).
- We do data mining, cleaning, and aggregation to create a new dataframe for our Machine Learning Model.
- We used K-Nearest-Neighbors (KNN) model to train and predict the user's class based on their device usage behavior.
- We used K-Means-Clustering (KMC) unsupervised learning algorithm to group the users based on their usage behavior.
- For both the KNN and KMC models we used the Elbow method to find the optimum value of K from elbow point.
- Both KNN and KMC were also used to label the battery health status.

3.5 Evaluation

In terms of the model evaluation, we did not rely on prediction accuracy but rather on reducing dimensions like false positives through looking at measures such as confusion matrix. In addition to confusion matrix, the results from the models were plotted for predicting the user types as well as predicting the battery health status. The model evaluation along with the plots are further explained in the Evaluation section of the report.

3.6 Deployment

For model deployment, a hybrid environment was created where the rest service is in java and model is deployed via a python script. The python script receives parameters from the java layer and loads the ML model saved at a particular directory. Upon loading the model, a prediction is performed for the type of user. Endpoint of the rest service will predict and return the type of user when invoked. Users can view their activity levels(low/medium/high) using the restful API.

4 Methodology

The architecture proposed for this solution is a microservices architecture where each component can be packaged and deployed separately. The architecture is featured in the figure below:

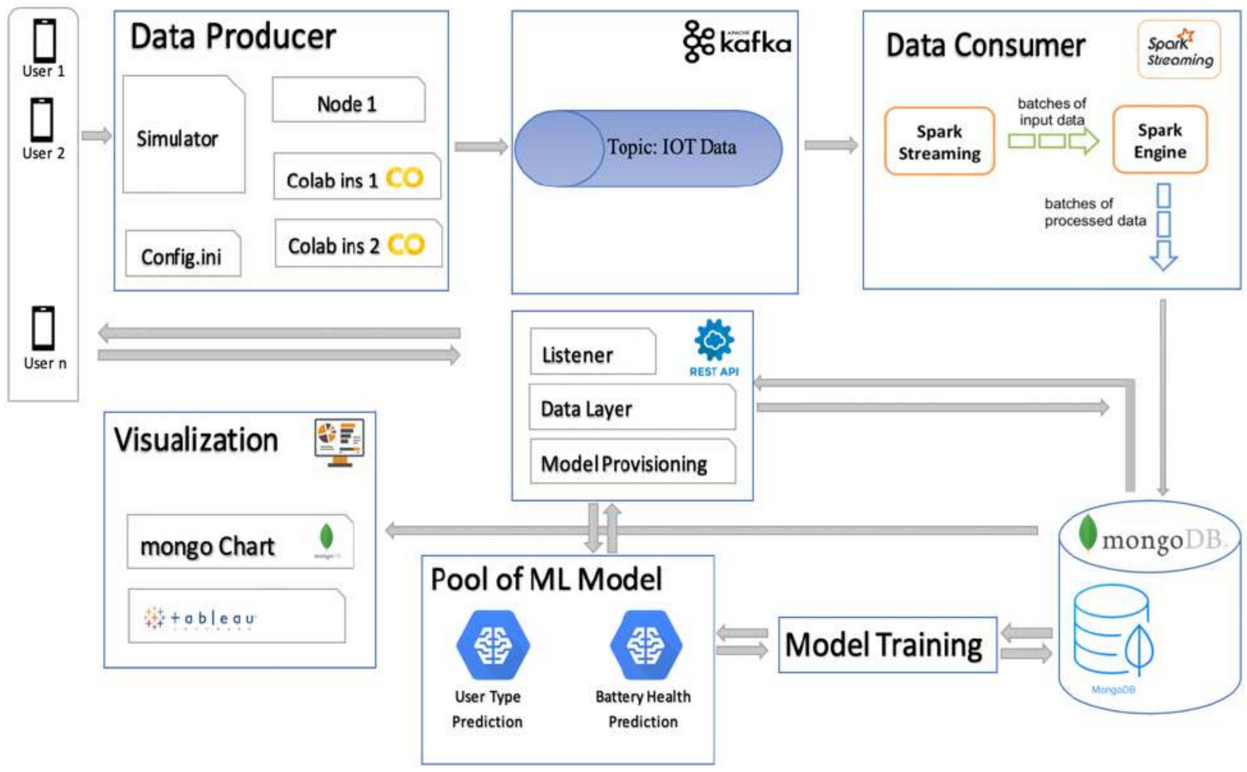


Figure 8: Data-Science architecture (microservices architecture: Each component can be packaged and deployed separately)

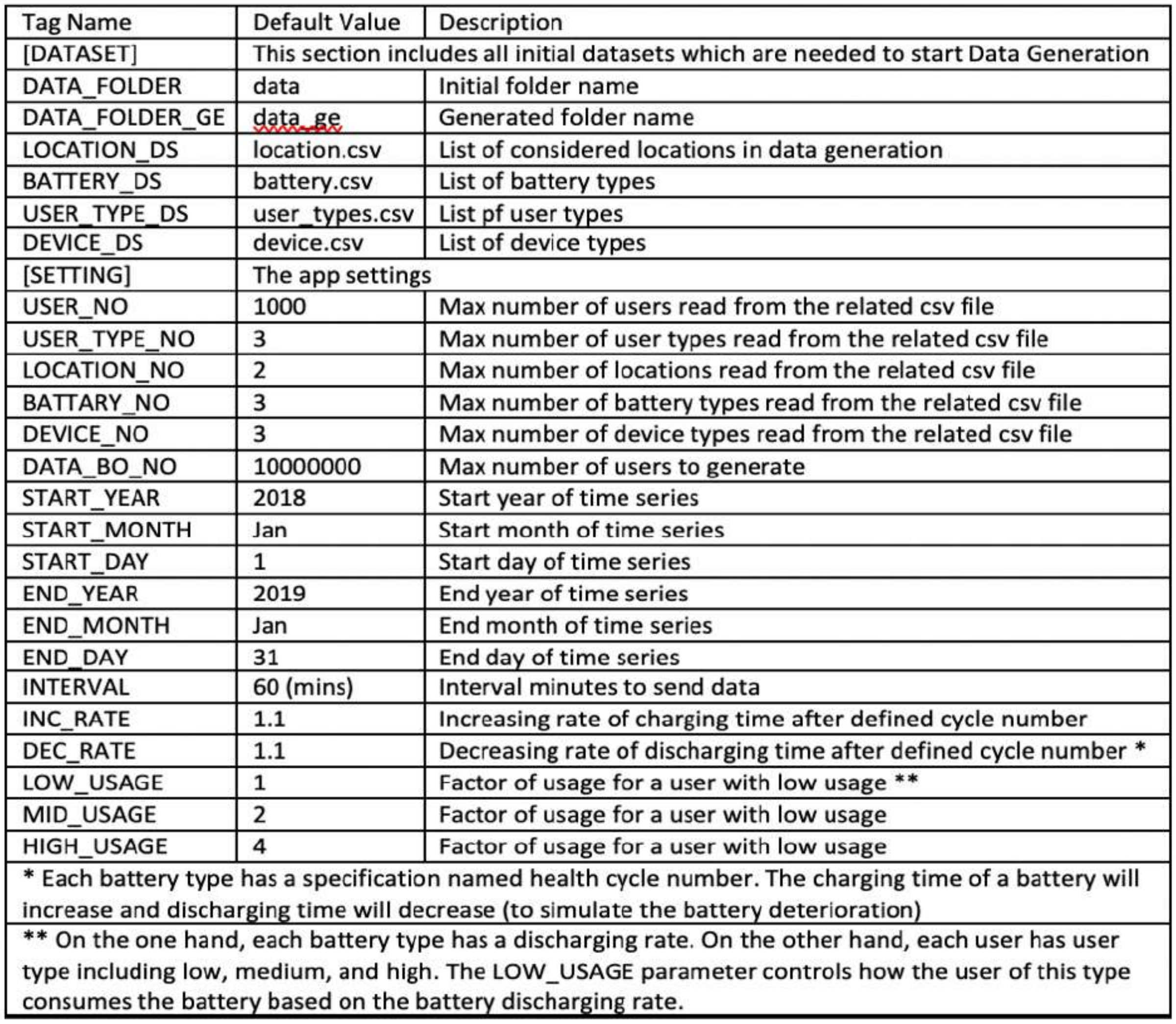
The main components for this architecture are defined as follows:

Data Producer:

Producers can be started at n different machines (n=3 in this case), sourcing data from 3 different users. Before starting producers, Installation.ipynb needs to be ran for creating Colab instances ready with all relevant packages.

Data Simulator:

Its role is to simulate reliable user data following strict guidelines from Samsung Digital Canada. We developed a data generator, which is represented in the data_generator.ipynb, that invokes separate classes involving entities such as Battery, Device, and User. Battery is the core entity that has four specifications including type, drain time (number of days to fully discharge a battery), recharge time (number of minutes to recharge a battery). There is a config.ini file where you can adjust the behavior of the data generator by changing the global parameters. A screenshot of the config.ini file can be found in the appendix section of the report. The structure of the configuration file is explained in the figure below.

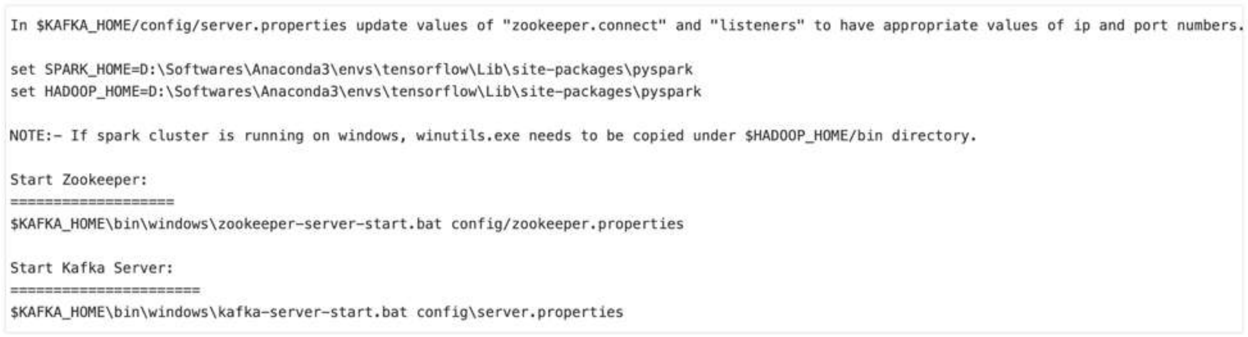


Tag Name	Default Value	Description
[DATASET]	This section includes all initial datasets which are needed to start Data Generation	
DATA_FOLDER	data	Initial folder name
DATA_FOLDER_GE	data_ge	Generated folder name
LOCATION_DS	location.csv	List of considered locations in data generation
BATTERY_DS	battery.csv	List of battery types
USER_TYPE_DS	user_types.csv	List pf user types
DEVICE_DS	device.csv	List of device types
[SETTING]	The app settings	
USER_NO	1000	Max number of users read from the related csv file
USER_TYPE_NO	3	Max number of user types read from the related csv file
LOCATION_NO	2	Max number of locations read from the related csv file
BATTARY_NO	3	Max number of battery types read from the related csv file
DEVICE_NO	3	Max number of device types read from the related csv file
DATA_BO_NO	10000000	Max number of users to generate
START_YEAR	2018	Start year of time series
START_MONTH	Jan	Start month of time series
START_DAY	1	Start day of time series
END_YEAR	2019	End year of time series
END_MONTH	Jan	End month of time series
END_DAY	31	End day of time series
INTERVAL	60 (mins)	Interval minutes to send data
INC_RATE	1.1	Increasing rate of charging time after defined cycle number
DEC_RATE	1.1	Decreasing rate of discharging time after defined cycle number *
LOW_USAGE	1	Factor of usage for a user with low usage **
MID_USAGE	2	Factor of usage for a user with low usage
HIGH_USAGE	4	Factor of usage for a user with low usage
* Each battery type has a specification named health cycle number. The charging time of a battery will increase and discharging time will decrease (to simulate the battery deterioration)		
** On the one hand, each battery type has a discharging rate. On the other hand, each user has user type including low, medium, and high. The LOW_USAGE parameter controls how the user of this type consumes the battery based on the battery discharging rate.		

Figure 9: Explained structure of config.ini file

Kafka Cluster:

Kafka cluster is setup on machine say instance-1 with topic named as "IOT Data":



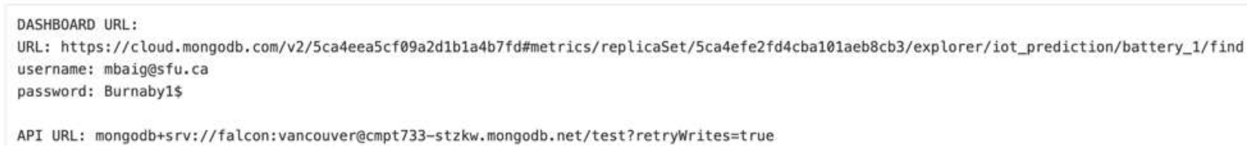
```
In $KAFKA_HOME/config/server.properties update values of "zookeeper.connect" and "listeners" to have appropriate values of ip and port numbers.

set SPARK_HOME=D:\Softwares\Anaconda3\envs\tensorflow\Lib\site-packages\pyspark
set HADOOP_HOME=D:\Softwares\Anaconda3\envs\tensorflow\Lib\site-packages\pyspark

NOTE:- If spark cluster is running on windows, winutils.exe needs to be copied under $HADOOP_HOME/bin directory.

Start Zookeeper:
===================
$KAFKA_HOME\bin\windows\zookeeper-server-start.bat config/zookeeper.properties

Start Kafka Server:
======================
$KAFKA_HOME\bin\windows\kafka-server-start.bat config\server.properties
```

Data Consumer:

Data consumer (Spark-streaming) can be started at another local machine say instance-2. This instance should have spark 2.4 version.

Backend:

For persisting data at backend, we are using free tier version of MongodB Atlas and chose AWS as cloud solution. For model deployment, a hybrid environment was created where the rest service is in java and model is deployed via a python script (as mentioned in the models deployment section). Endpoint of the rest service will predict and return the type of user when invoked. Users can view their activity levels(low/medium/high) using the restful API. Security related settings need to be changed to whitelist ip addresses from where we connect to database.



```
DASHBOARD URL:
URL: https://cloud.mongodb.com/v2/5ca4eea5cf09a2d1b1a4b7fd#metrics/replicaSet/5ca4efe2fd4cba101aeb8cb3/explorer/iot_prediction/battery_1/find
username: mbaig@sfu.ca
password: Burnaby1$

API URL: mongodb+srv://falcon:vancouver@cmpt733-stzkw.mongodb.net/test?retryWrites=true
```

Models deployment:

For model deployment, a hybrid environment was created where the rest service is in java and model is deployed via a python script. The python script receives parameters from the java layer and loads the ML model saved at a particular directory. Upon loading the model, a prediction is performed for the type of user. Endpoint of the rest service will predict and return the type of user when invoked. Users can view their activity levels(low/medium/high) using the restful API.

Visualization:

The visualization in this project was performed in Tableau as well as in the beta version of the newly launched Mongo Charts. Thus, experimenting with different visualization tools.

1. **Tableau** loads data in CSV format.
2. **Mongo Charts** (Beta Version) – Data is passed directly from MongoDB Atlas (hosted on cloud with AWS)

The Beta version of mongo charts is used for basic visualization, data-source (database is located at Mongo Atlas) is mentioned at mongo chart and frequency of update is at 1 minute. Currently this product version is having limitation as it is at earlier stage and does not support complex visualization therefore Tableau is used for full breadth of plots and charts. In terms of Tableau, since trial versions of product is used, there is no support for continuous integration. As a workaround, a daemon process (shell script at background) is ran to continuously update data in file which is imported to Tableau. This provides an end-to-end connected layer without any manual intervention for updating data.

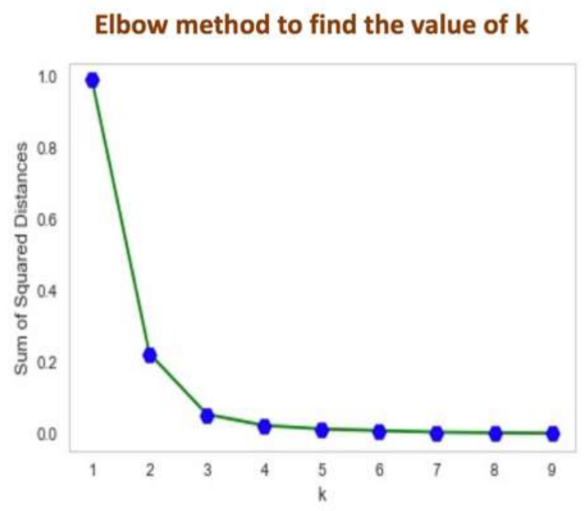
Link to Tableau charts:
https://public.tableau.com/profile/mehdi.lebdi#!/vizhome/tableau_eda/Userbehaviorshowinghighmedlowactivity

Link to MongodB charts: https://charts.mongodb.com/charts-project-0-sltax/dashboards/fe20a563-fc84-4f07-b5ae-e7166756a013

5 Evaluation

1. Elbow Method:

The Elbow method was used in order to find the value of k to categorize user types and battery health status. K denotes the number of classes or clusters that are present in the dataset.



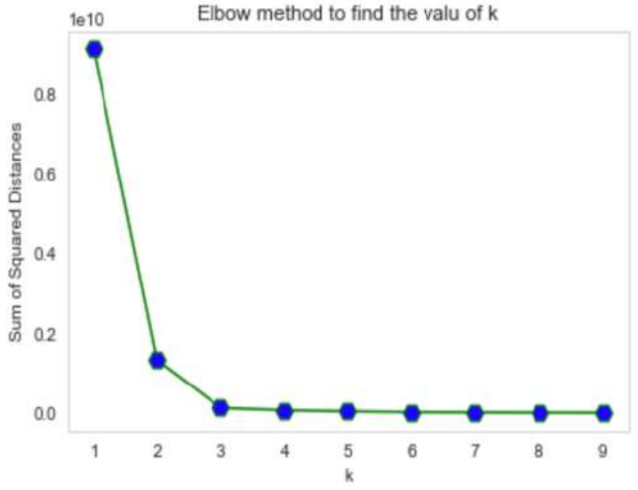


Figure 10: Elbow method to find the value of k for battery status

Figure 11: Elbow method the find the value of k for user types

From the above plots, we can clearly see the Elbow point at k=3. Therefore, denoting that the model is classifying into 3 different categories of user types (and battery health status).

2. Analysis of **K-NN** model:

Confusion Matrix:

The confusion matrix is often used to describe the performance of a classification model. In predictive analytics, a table of confusion (sometimes also called a confusion matrix), is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives. This allows more detailed analysis than mere proportion of correct classifications (accuracy). Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced.

Confusion matrix for **K-NN** model for predicting the **battery health status**:

		Actual Class		
	Battery Health	Poor	Intermediate	Healthy
Predicted Class	Poor	**5197**	91	6
	Intermediate	66	**5203**	74
	Healthy	3	104	**5348**

To better understand the table above, 5197 represents the number of actual poor batteries that were correctly predicted as poor batteries. On the other, 66 represents the number of actual poor batteries that were predicted as intermediate and 3 as healthy. This means that 5197 health status were correctly predicted as poor while 66 and 3 were wrongly predicted.

Since, there is a higher number of poor batteries that were correctly predicted as poor and lower numbers for poor batteries that were wrongly predicted as intermediate or healthy, we can say that these measures are a good representation of how well the model is performing.

In the case of battery health status, the number of actual battery health to predicted battery health was relatively high and therefore a sign that the predictions are accurate, and the model is performing well.

Confusion matrix for **K-NN** model for predicting the **user type**:

		Actual Class		
	User Type	Low	Medium	High
Predicted Class	Low	**83**	4	2
	Medium	0	**97**	6
	High	2	3	**103**

The same logic applied to the confusion matrix for the battery health status can also apply to the user type. In this confusion matrix, of the 85 actual low users, the system predicted 83 correctly and 2 wrongly as high users. Of the 104 actual medium users, the system predicted 97 correctly and 7 wrongly as low and high users. Of the 111 high users, the system predicted 103 correctly and 8 wrongly as low and medium users.

Since, there is a higher number of low users that were correctly predicted as low and lower numbers for low users that were wrongly predicted as intermediate or healthy, we can say that these measures are a good representation of how well the user type classifier is performing.

In the case of user types, the number of actual users to predicted users was relatively high and therefore a sign that the predictions are accurate, and the model is performing well.

➔ Hence, the main goal of the confusion matrix is to evaluate the number of correctly predicted classes. The tables produced serve as indicators to evaluate and improve the performance of the models used in this project.

3. Analysis of **K-Means clustering** model:

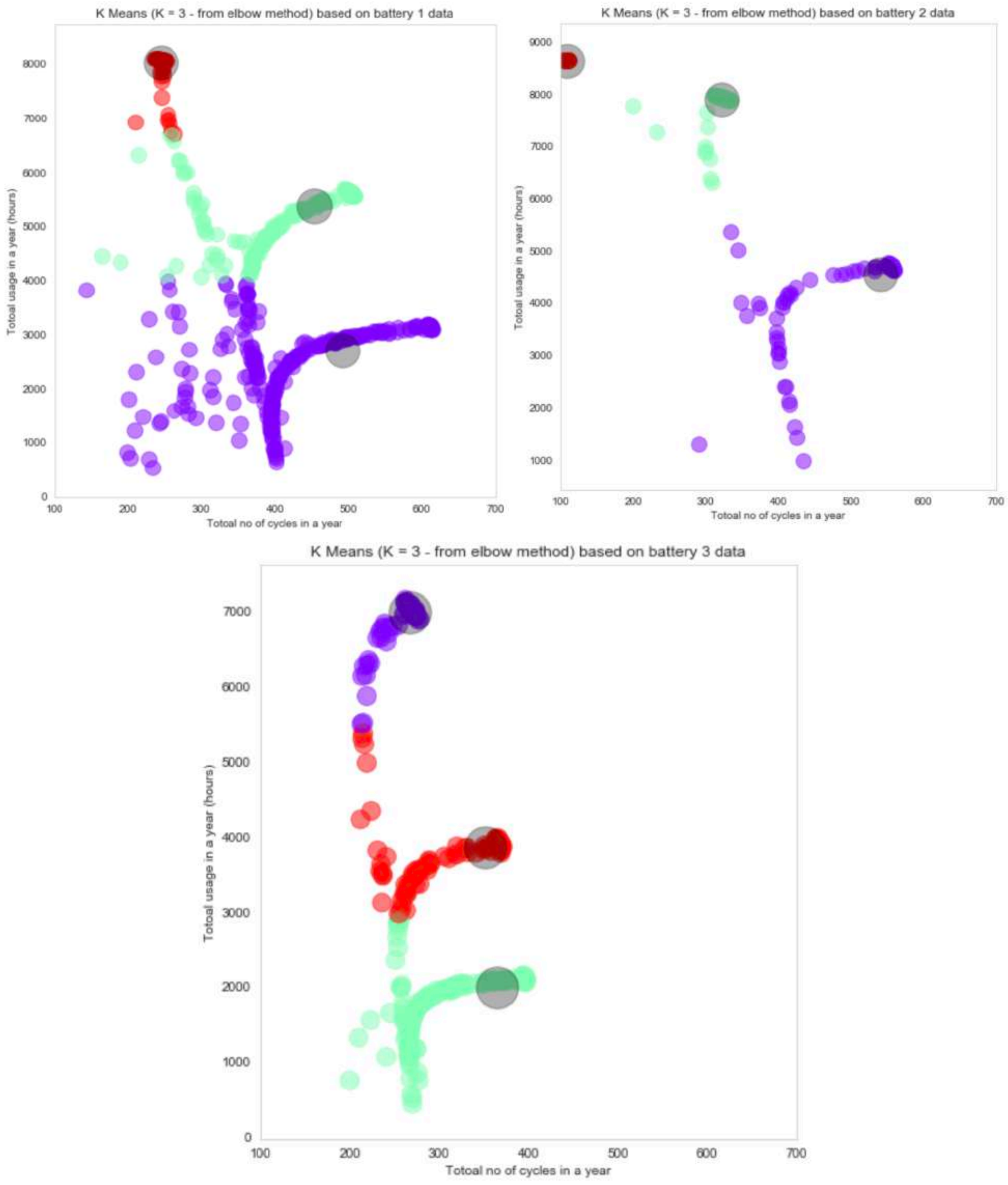


Figure 12: K-Means clustering results to classify user types based on their usage behavior performed on 3 types of batteries.

➔ The above plots denote that 3 distinct users were detected after performing K-Means clustering used for predicting the user type. The user is labeled as low, medium, or high activity user. This was performed on 3 different batteries.

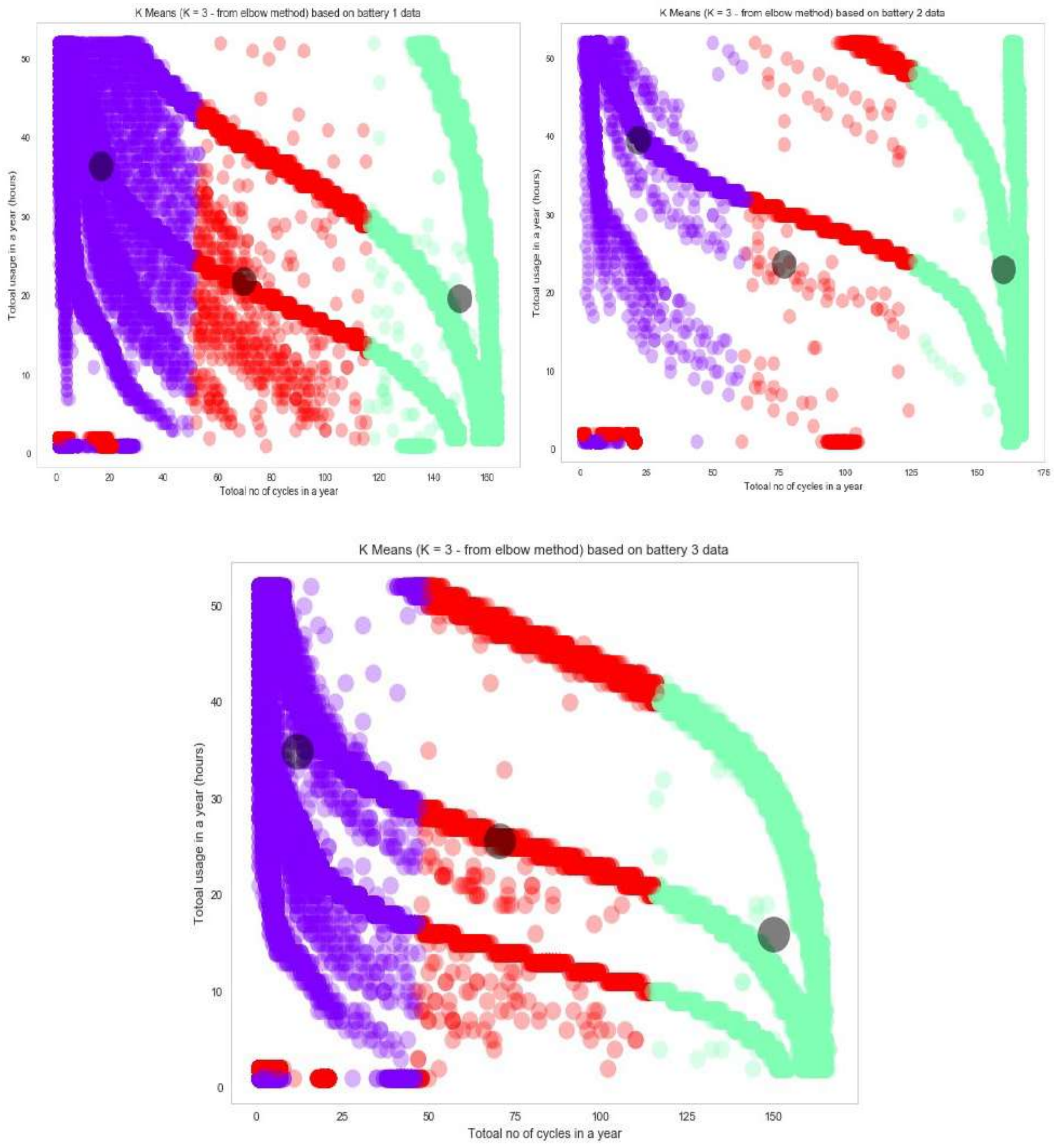
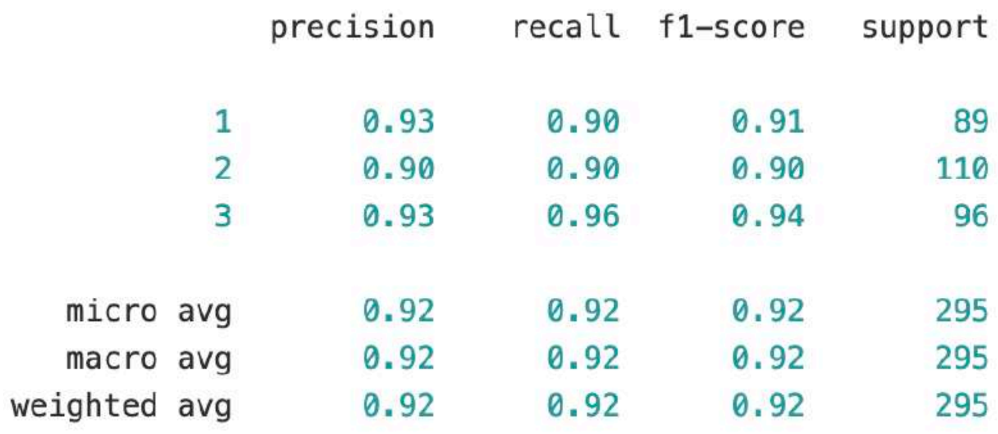


Figure 13: Clustering of data points for battery health from K-Means clustering algorithm.

➔ The above plots denote that the battery health status can be classified into 3 categories. This was detected after performing K-Means clustering used for predicting the batter health status. The battery status was labeled as poor, intermediate, or healthy. This was performed on 3 different batteries.

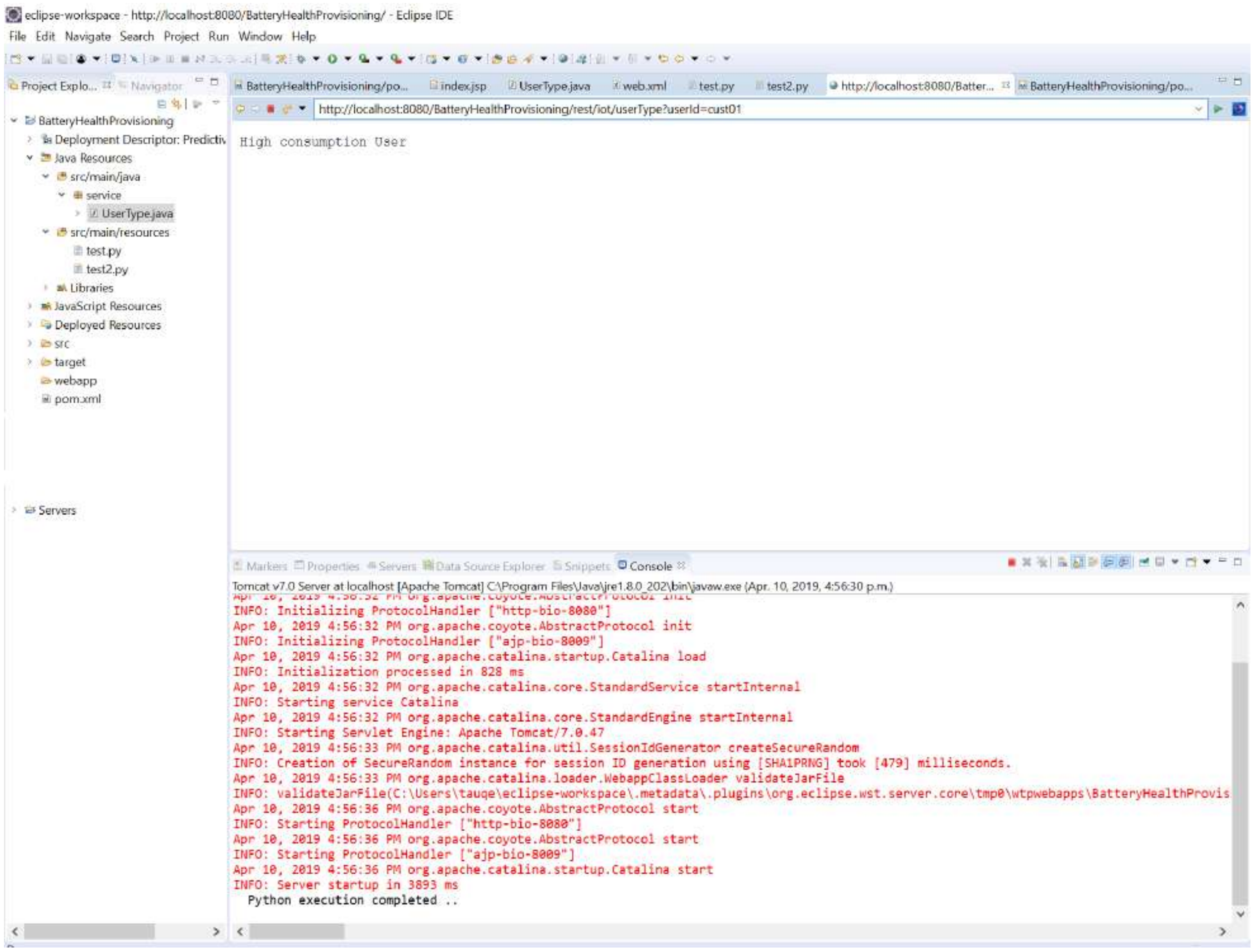
Other performance measures were also used to further evaluate the models, such as Precision, Recall and F-score. The report for predicting the user type is shown below depicting promising predictions from these measurements (precision/recall around 0.90).



```
              precision    recall  f1-score   support

           1       0.93      0.90      0.91        89
           2       0.90      0.90      0.90       110
           3       0.93      0.96      0.94        96

   micro avg       0.92      0.92      0.92       295
   macro avg       0.92      0.92      0.92       295
weighted avg       0.92      0.92      0.92       295
```

Figure 14: Report generated in notebook for predicting the user type

6 Data Product



```
http://localhost:8080/BatteryHealthProvisioning/rest/iot/userType?userId=cust01
```

```
High consumption User
```

```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_202\bin\javaw.exe (Apr. 10, 2019, 4:56:30 p.m.)
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Apr 10, 2019 4:56:32 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Apr 10, 2019 4:56:32 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 828 ms
Apr 10, 2019 4:56:32 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Apr 10, 2019 4:56:32 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.47
Apr 10, 2019 4:56:33 PM org.apache.catalina.util.SessionIdGenerator createSecureRandom
INFO: Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [479] milliseconds.
Apr 10, 2019 4:56:33 PM org.apache.catalina.loader.WebappClassLoader validateJarFile
INFO: validateJarFile(C:\Users\tauqe\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\BatteryHealthProvis
Apr 10, 2019 4:56:36 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Apr 10, 2019 4:56:36 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Apr 10, 2019 4:56:36 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3893 ms
  Python execution completed ..
```

Figure 15: Rest service in java - invokes python script for model deployment

The product resulting from this project is the deployment of the models using a java Restful API. A hybrid environment was created where the rest service is in java and model is deployed via a python script. The python script receives parameters from the java layer and loads the Machine Learning model saved at a particular directory. Upon loading the model, a prediction is performed for the type of user. Endpoint of the rest service will predict and return the type of user when invoked. Users can view their activity levels(low/medium/high) using the restful API. The same development concept can be mimicked for users to view their battery health status using a restful API.

7 Lessons Learnt

Kafka integration: The Kafka messages produced and consumed were in two separate machines. The network connection required port and firewall configuration to integrate the connection between the two machines. This required knowledge on peer-to-peer ad-hoc wireless network.

Data generation and simulation subsystem: Due to the project statement, we needed to work with the Samsung battery data which was not publicly accessible anymore (it was publicly shared last year for a challenge) and the process of getting approval to share that data would take at least two months to get the required licenses. Unfortunately, it wasn't possible to work with the original Samsung data. Therefore, we decided to generate the dataset to simulate real world IoT data. Having met with Bandeep Singh, Data Scientist from Samsung Digital Canada, we understood their requirements and the real data specifications. So, we developed the Data Generator based on the latest standards as thoroughly described in the data_generator notebook in the code.

Practical experience in big data: Gained practical experience in deploying an end-to-end big data solution and integrate the different components. For the implementation, several technologies had to be merged to work in sync, such as Kafka, Spark, MongoDB, and Tableau.

Practical experience in IoT data: Gained practical experience in implementing solution to optimize the performance of IoT devices. In addition to classifying the types of users by analyzing their activity as well as predicting the current health status of a device battery.

8 Summary

The project is an end-to-end solution that is implemented to evaluate and optimize the performance of IoT devices. It revolves around the IoT domain and the business of predictive maintenance. For industrial equipment, more efficient maintenance translates into big savings. It's a combination of longer equipment life and avoiding expensive downtime, which can lead to costly liabilities such as device failure. We can make use of algorithms based on big data to predict the type of users adopting these machines and the equipment life span. This will provide a deeper understanding about equipment optimization as well as if and when it needs to be serviced or replaced. For this project, based on history of battery usage the objective was to determine whether we can predict the current health of a battery as well as categories users. For the implementation, several technologies had to be merged to work in sync, such as Kafka, Spark, MongoDB, and Tableau. **Real-time analytics of events** was done on Tableau, generating graphs that provide insights around IoT data. Link to Tableau charts: https://public.tableau.com/profile/mehdi.lebdi#!/vizhome/tableau_eda/Userbehaviorshowinghighmedlowactivity

Future Work

- ❑ Explore other ML models for predicting Remaining Useful Life (RUL) of batteries.
- ❑ Explore IoT data from other devices such as electric cars, home appliances, etc. to optimize their performance.
- ❑ Provide device recommendation to users depending on their daily device usage.

Acknowledgments

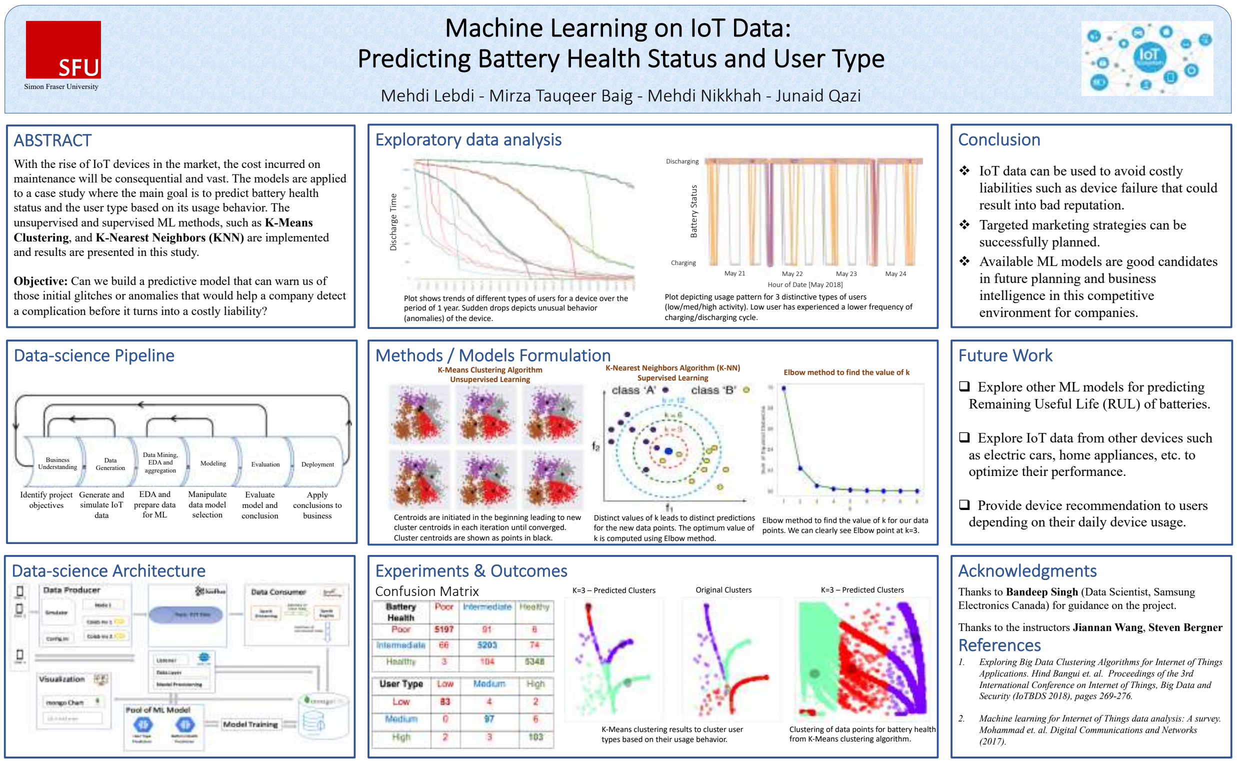
- Thanks to **Bandeep Singh** (Data Scientist, Samsung Electronics Canada) for his guidance on the project.
- Thanks to the instructors **Jiannan Wang** and **Steven Bergner** for their guidance and many lessons.

Reference

[1] Exploring Big Data Clustering Algorithms for Internet of Things Applications. Hind Bangui et. al. Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018), pages 269-276.

[2] Machine learning for Internet of Things data analysis: A survey. Mohammad et. al. Digital Communications and Networks (2017).

Appendix


Machine Learning on IoT Data:
Predicting Battery Health Status and User Type

Mehdi Lebdi - Mirza Tauqeer Baig - Mehdi Nikkhah - Junaid Qazi

Simon Fraser University

ABSTRACT

With the rise of IoT devices in the market, the cost incurred on maintenance will be consequential and vast. The models are applied to a case study where the main goal is to predict battery health status and the user type based on its usage behavior. The unsupervised and supervised ML methods, such as **K-Means Clustering**, and **K-Nearest Neighbors (KNN)** are implemented and results are presented in this study.

Objective: Can we build a predictive model that can warn us of those initial glitches or anomalies that would help a company detect a complication before it turns into a costly liability?

Exploratory data analysis

Plot shows trends of different types of users for a device over the period of 1 year. Sudden drops depicts unusual behavior (anomalies) of the device.

Plot depicting usage pattern for 3 distinctive types of users (low/med/high activity). Low user has experienced a lower frequency of charging/discharging cycle.

Conclusion

- ❖ IoT data can be used to avoid costly liabilities such as device failure that could result into bad reputation.
- ❖ Targeted marketing strategies can be successfully planned.
- ❖ Available ML models are good candidates in future planning and business intelligence in this competitive environment for companies.

Data-science Pipeline

Business Understanding: Identify project objectives
Data Generation: Generate and simulate IoT data
Data Mining, EDA and aggregation: EDA and prepare data for ML
Modeling: Manipulate data model selection
Evaluation: Evaluate model and conclusion
Deployment: Apply conclusions to business

Methods / Models Formulation

K-Means Clustering Algorithm Unsupervised Learning

Centroids are initiated in the beginning leading to new cluster centroids in each iteration until converged. Cluster centroids are shown as points in black.

K-Nearest Neighbors Algorithm (K-NN) Supervised Learning

Distinct values of k leads to distinct predictions for the new data points. The optimum value of k is computed using Elbow method.

Elbow method to find the value of k

Elbow method to find the value of k for our data points. We can clearly see Elbow point at k=3.

Future Work

- ❑ Explore other ML models for predicting Remaining Useful Life (RUL) of batteries.
- ❑ Explore IoT data from other devices such as electric cars, home appliances, etc. to optimize their performance.
- ❑ Provide device recommendation to users depending on their daily device usage.

Data-science Architecture

Data Producer
Data Consumer
Visualization
Pool of ML Model
Model Training

Experiments & Outcomes

Confusion Matrix

Battery Health	Poor	Intermediate	Healthy
Poor	5197	91	6
Intermediate	66	5203	74
Healthy	3	104	5348

User Type	Low	Medium	High
Low	83	4	2
Medium	0	97	6
High	2	3	103

K=3 – Predicted Clusters / Original Clusters: K-Means clustering results to cluster user types based on their usage behavior.

K=3 – Predicted Clusters: Clustering of data points for battery health from K-Means clustering algorithm.

Acknowledgments

Thanks to **Bandeep Singh** (Data Scientist, Samsung Electronics Canada) for guidance on the project.

Thanks to the instructors **Jiannan Wang**, **Steven Bergner**

References

1. *Exploring Big Data Clustering Algorithms for Internet of Things Applications. Hind Bangui et. al. Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018), pages 269-276.*
2. *Machine learning for Internet of Things data analysis: A survey. Mohammad et. al. Digital Communications and Networks (2017).*

Figure 16: Poster for Machine Learning in IoT data – Predicting Battery Health Status and User Type

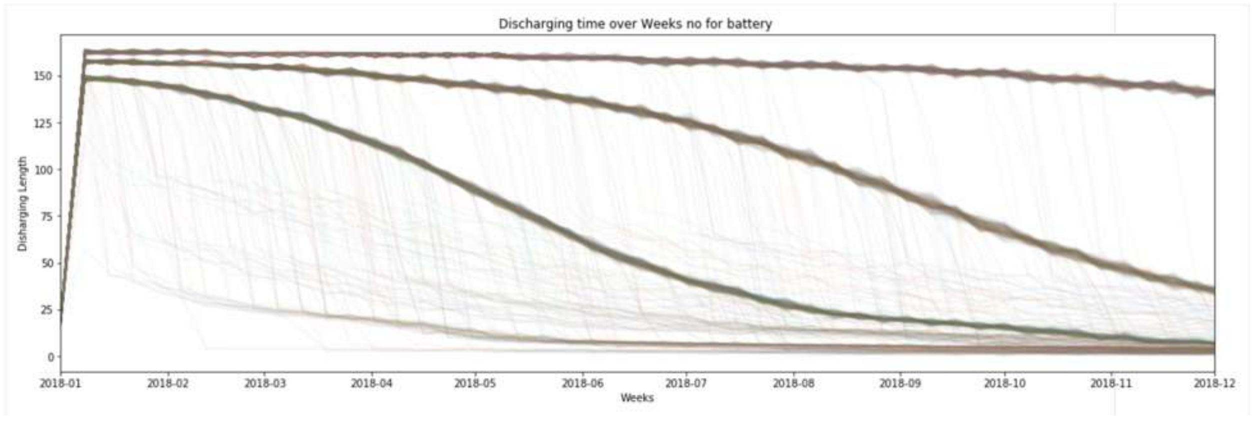


Figure 17: Discharging time over a period of 1 year for all users plotted with Matplotlib

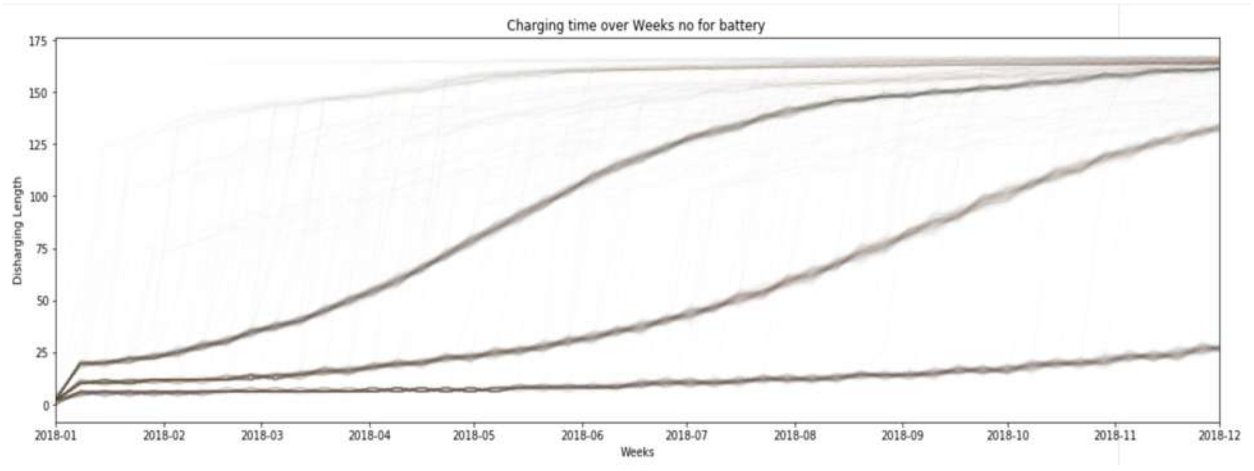


Figure 18: Charging time over a period of 1 year for all users plotted with Matplotlib

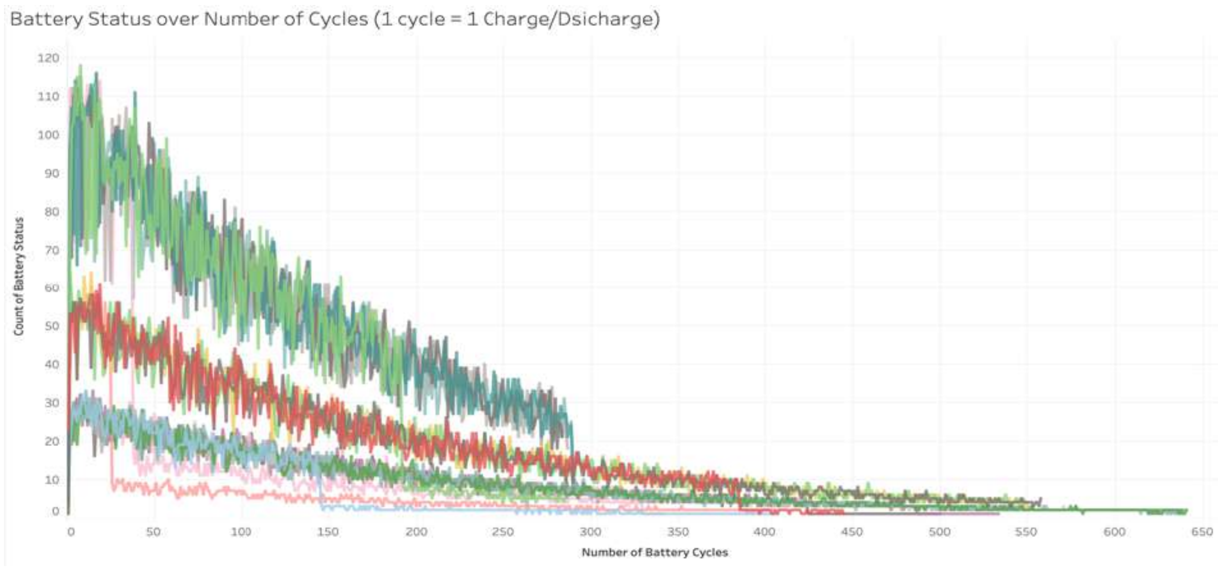


Figure 19: Discharging time over the number of cycles. The plot shows that the battery health is deteriorating as the number of cycles is increasing. The rate at which the battery is losing health differs by battery type and user type

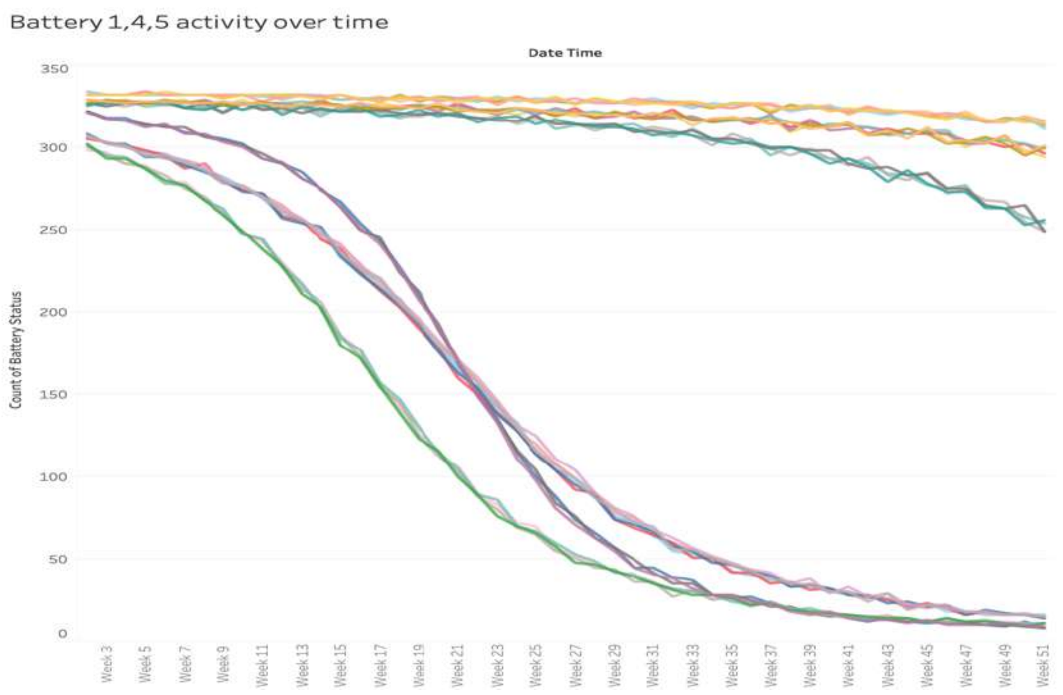
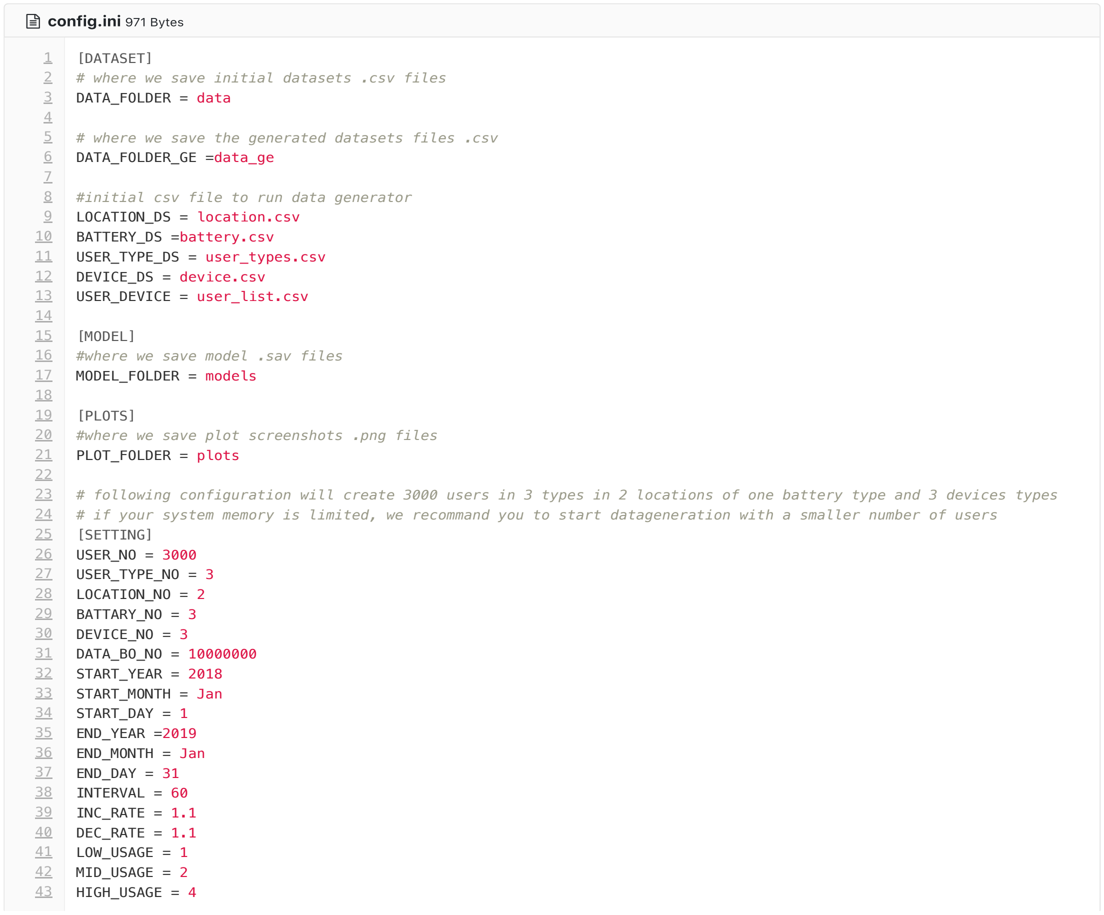


Figure 20: Plot showing 2 types users with 3 different devices – For one user the battery deteriorates in week 23-25 and for the other user the battery stays healthy for a longer period of time proving that one user is a High user while the other is a Low user



config.ini 971 Bytes

```
[DATASET]
# where we save initial datasets .csv files
DATA_FOLDER = data

# where we save the generated datasets files .csv
DATA_FOLDER_GE =data_ge

#initial csv file to run data generator
LOCATION_DS = location.csv
BATTERY_DS =battery.csv
USER_TYPE_DS = user_types.csv
DEVICE_DS = device.csv
USER_DEVICE = user_list.csv

[MODEL]
#where we save model .sav files
MODEL_FOLDER = models

[PLOTS]
#where we save plot screenshots .png files
PLOT_FOLDER = plots

# following configuration will create 3000 users in 3 types in 2 locations of one battery type and 3 devices types
# if your system memory is limited, we recommand you to start datageneration with a smaller number of users
[SETTING]
USER_NO = 3000
USER_TYPE_NO = 3
LOCATION_NO = 2
BATTARY_NO = 3
DEVICE_NO = 3
DATA_BO_NO = 10000000
START_YEAR = 2018
START_MONTH = Jan
START_DAY = 1
END_YEAR =2019
END_MONTH = Jan
END_DAY = 31
INTERVAL = 60
INC_RATE = 1.1
DEC_RATE = 1.1
LOW_USAGE = 1
MID_USAGE = 2
HIGH_USAGE = 4
```

Figure 21: Structure of configuration file - config.ini