**Buy Back Modelling of the Community Battery**

A buyback model for a community battery refers to a scheme where energy stored in a shared battery system is made available to members of the community at a lower cost than purchasing energy from the grid. This model enables individuals or businesses to store excess energy generated from renewable sources such as solar panels and wind turbines, which can be sold back to the battery provider at a fair price. The community battery then redistributes this stored energy to members of the community who need it at a lower cost. The buyback model creates a more sustainable and affordable energy system, reducing the reliance on fossil fuels and promoting the use of renewable energy sources.

To ensure the financial viability of a community battery, the council must set competitive pricing strategies, maintain the battery system's efficiency and reliability, and explore additional revenue streams. The council can achieve this by offering incentives for customers to switch to the community battery, regular maintenance and testing, and participating in energy markets or offering energy storage services to third-party providers. By doing so, councils can generate sufficient revenue to cover the initial cost and ongoing maintenance expenses of the community battery while promoting sustainable energy use and supporting the local community.

A screenshot of a computer

Description automatically generated with medium confidence

**Usage of the code:**

File: BuyBackModel.py

Adjust variables you want to test: (If you want to set a specific battery size, you can just go to the function and set it to return the size in MW)

Functions:

* PrintBuyBack()

*Prints a pandas DataStructure table into the terminal.*

* Analyse()

*Analyses the table and produces an automated paragraph outlining some of the statistics,*

* GraphSavings()

*Graphs the community savings, council savings and cost of the battery benchmark.*

* GraphSavingsSave()

A screen shot of a graph

Description automatically generated with medium confidence*Saves the GraphSavings() graph as a .png file.*

* GraphEnvironment()

*Graphs the CO2 emissions savings.*

* GraphEnvironmentSave()

*Saves the GraphEnvironment() Graph as a .png file.*

* DataFrametoCSV()

A screen shot of a graph

Description automatically generated with medium confidence*Saves the pandas DataStructure table as a .csv file.*

Example Outputs:

A picture containing screenshot

Description automatically generated

File: OptimumBuyPrice.py

About: Finds the optimum battery price by iterating +0.005 from the initial sell cost until the council makes back all their money. You can then use this to make decisions and readjust the initial BuyBackModel.py.

Adjust variables you want to test: (If you want to set a specific battery size, you can just go to the function and set it to return the size in MW) Make sure the buyprice is set at 0 or at your sell price.

Functions:

* OptimumBuyBack()

*Prints a statement of the Optimum Buy Price, how much the council saves and the battery cost.*

* OptimumBuyBackGraph()

A screen shot of a graph

Description automatically generated with medium confidence*Graphs the battery cost over the councils savings.*

* OptimumBuyBackGraphSave()

*Saves the graph as a .png.*

Example Outputs:

File: OppurtunityCostModel.py

About: Just graphs the total opportunity cost of the model of the size of your battery and number of households.

Adjust variables you want to test: (If you want to set a specific battery size, you can just go to the function and set it to return the size in MW) Make sure the buyprice is set at 0 or at your sell price.

Functions:

* Graph()

*Graphs the opportunity cost of the model.*

* GraphSave()

*A screenshot of a graph

Description automatically generated with medium confidenceSaves the graph in .png form.*

Example Output:

File: OptimumOppurtunityCosts.py

About:

* This model is less straightforward. In order to determine the ideal scenario for your battery, you will need to iterate through the BuyBack Model and this model.
* This model assumes that all battery sizes are integers of MW.
* This model assumes that the battery will be replaced at the end of its estimated lifetime.
* Essentially optimising opportunity cost is based on the following scenarios:
  + A picture containing text, screenshot, line, plot

    Description automatically generatedHave enough houses to maintain 100% utilisation of the battery so you will have a high opportunity cost, however you maximise the amount of money saved by the battery. (225 Houses, 1MW, 10 Years)

A screenshot of a computer screen

Description automatically generated with low confidence

* + A picture containing text, screenshot, line, display

    Description automatically generatedYou want to try gain as much as you can from all the power generating, meaning the battery will only reach 100% utilisation at the end of its lifetime. (185 Houses, 1 MW, 10 Years)



A screen shot of a computer

Description automatically generated with low confidence

* + Or a midground where opportunity cost kicks in half-way. (202 Houses 1MW, 10 Years)

A screen shot of a computer

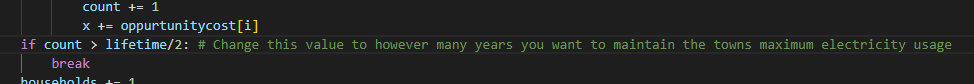
Description automatically generated with low confidenceA picture containing text, screenshot, line, plot

Description automatically generated

* These 3 models form a generic basis for deciding on how many households to connect to a battery.

Usage of the model:

* Change the variables to what you desire.
* You can modify the size of the battery manually if you directly edit the function.
* Within the optimum function there’s an if statement you can change if you want to modify after how many years you want the battery to achieve this 100% utilisation; by default it is set to half the battery life expectancy.



Functions:

* Optimum()
* *Iterates over household values until it finds a value that reaches x amount of years of when opportunity cost is > 0 and prints the value of the household and opportunity cost lost.*
* Graph()

*Displays a rough graph of the optimum value.*

* GraphSave()

*Saves the graph in a .png file.*