Blockchain-based automatic energy efficiency model for Performance Contract application

Pedro Manuel Ferreira Sismeiro
pedro.sismeiro@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

December 2020

Abstract

Energy transformation and usage is still the major source of greenhouse gas emissions. On demand side, there is a global push to invest on energy efficiency improvements, which has been slowing down. Efficiency improvements have benefits for countries such as increased energy security, less spending on fossil fuels and emissions reduction. Energy Service Companies (ESCOs) deploy energy conservation measures (ECMs) through Energy Performance Contracts (EPCs), which guarantee a level energy/cost savings. Measurement and Verification (M&V) procedures are essential to EPCs, as to audit contract terms and ECM efficiency. Poor M&V frameworks can generate adversarial distrust between parties involved and unclear savings calculations. This integration thesis aims to increase transparency in EPC-mediated ECM implementations by properly assessing, storing and securing savings calculations. We develop a baseline model using XGBoost for two IST campus buildings which underwent retrofits and estimate savings from the difference to actual consumption data, for the same period. The used models presented a CV(RMSE) of under 7.8% and yielded savings percentages of 16.9$\pm$7.3% and 20.6$\pm$6.3%. Savings information are then posted in a blockchain ledger composed of building nodes. The transaction validation mecanism verifies if an accurate baseline model was used as basis for the calculations. A more clear and thrustworthy M&V platform for EPC execution was developed. Conclusions, limitations and future improvements are discussed.

Keywords: Energy efficiency, Energy Performance Contracting, XGBoost, Blockchain, Energy services.

1. Introduction

We are afield from meeting the 3 energy-related Sustainable Development Goals (SDGs), proposed by the UN, which consist on: tackling climate change, assuring universal access to energy and reducing health impacts of air-pollution [12].

At the same time, energy generation and usage remains the major source of greenhouse gas emissions, which need to quickly reduce and establish a *plateau* of net-zero balance between anthropogenic emissions by sources and removals by sinks [21, 12].

In terms of economic push, capital is already moving from fossil to renewable to a significant extent. Improvements shall represent an increase in overall investment, which will be counterbalanced by reduced fuel costs on the consumer end, afterwards. On the supply side, the largest increase in investment comes from renewable-based power installments, which are expected to double until 2050, as well as additional spending on electricity grids and storage. On the demand side, there's a global push for further investment on energy efficiency improvements. Still, it is currently not in pace with supply side developments [22].

Transitioning to a low carbon economy requires a more synergistic energy system, that relies less on fuel combustion and more on renewable based power, but we still have to focus about energy efficiency improvements on the demand side, while not trading off energy security nor affordability. As such, boosting energy efficiency improvements in a reliable way shall be the main focus of this dissertation.

1.1. Motivation

The proposed solution takes an holistic approach towards improving energy efficiency, dealing with it from the client demand side upwards, helping to secure better performance when trying to audit efficiency improvements.

We provide a tool for securely assessing efficiency improvements while helping to boost business models based on energy trading, energy certificates and/or energy performance contracting, well known to Energy Service Companies (ESCOs) [18, 16, 17, 19].

In an ever more stressed energy environment, with rising electricity costs, additional environmental regulations and less marginal profit, buildings owners benefit from reducing their energy consumption and costs. ESCOs provide clients with Energy Performance Contracts (EPCs), which are binding agreements under which Energy Conservation Measures (ECMs) are provided, verified and monitored, during a certain period of time [11].

The level of energy savings secures the financial revenue that is used to fund the capital costs incurred at the ESCO side. The financial savings are, in general, shared between the two parties since the beginning. Once costs have been repaid, the client keeps the whole savings generated [11, 20]. In case of failure in provisioning the contractually-agreed energy savings level, financial penalties are applied to the service provider, which reduces the contract revenue.

Measurement and Verification (M&V) procedures are, then, essential to EPCs, to audit the contract terms and ECM efficiency. The energy savings are generally computed as the difference between a predictive baseline model and the actual post-EPC measured energy consumption, over the considered period [11, 20, 18, 27]. A poor framework for M&V can generate problems such as an unbalanced performance risk and unclear or inappropriate savings calculations [13, 27]. The International Performance Measurement and Verification Protocol (IPMVP) has been developed to provide guidance and standards for M&V procedures [25].

There's also the risk of data tampering from both parts or from external providers, leading to inaccurate savings calculations, especially since depending on the modality of the EPC, the ESCO might be entitled to any excess savings. This way, an adversarial relationship can be generated between parts [11].

In essence, we face a problem of *trust* and understandibility when dealing with EPCs, which is limiting the widespread of this useful tool. Authors also claim lack of standardization or lack of policy concerning EPC execution [16, 17, 14, 28, 26].

For its incorruptible and immutable character together with the lack of need for a trusted third party, Blockchain's ability to track down transactions is becoming of increasing attention on energy sector applications [19]. As such, this present work studies its application on EPCs, to increase parties trust in M&V procedures. By having it designed in a way that all model predictions and calculations happen inside it, we will develop a more trustworthy and standardized framework for executing and auditing EPCs.

1.2. Concept

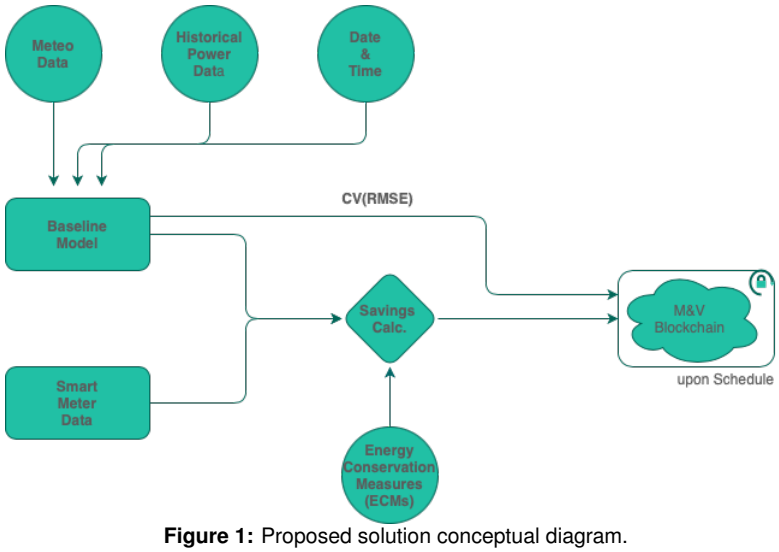

Figure 1: Proposed solution conceptual diagram.

This thesis integration shall bring transparency and security to EPC-mediated ECM implementations, by properly assessing, storing and securing savings calculations, following M&V 2.0 tendencies. We shall develop an energy consumption forecasting model based on relevant data from selected features, called a baseline model. Then, actual data will be compared to the model predictions and the savings are calculated from the difference between them, under a given uncertainty. As to guarantee transparency, these results are stored in a blockchain composed of building nodes posting on a network, which verifies basline model CV(RMSE) (Fig. 1). This is expected to be a relatively new contribution towards the main cited challenges faced by ESCOs in EPCs when tackling en-

ergy efficiency, global primary energy intensity improvement and, ultimately, CO_2 emissions reduction. Khatoon et al. (2019) [23] and Gurcan et al. (2018) [18] are two of the few literature references to a solution like the one proposed by this dissertation, usually relying on common blockchain platforms like Ethereum/Hyperledger. The core values of our prototype are simplicity, adaptability and effectiveness in the scope of the purpose of use, making it easier to adapt further details in this platform when faced with different real world applications/security requirements.

2. Methodology
2.1. Energy Modeling
2.1.1 Exploratory Data Analysis

We shall first analyze collected energy consumption data from 2017-2018 by smart meters deployed throughout four Instituto Superior Técnico - Alameda campus' buildings: Civil, Central, North Tower and South Tower. Most considerations will take the Civil building as basis since it is the most representative one of the whole campus, in terms of data homogeneity and consumption pattern. First, let we need to visualize the data and, for that purpose, we have built a bar histogram of the recorded hourly values of drained power for Civil (Fig. 2).

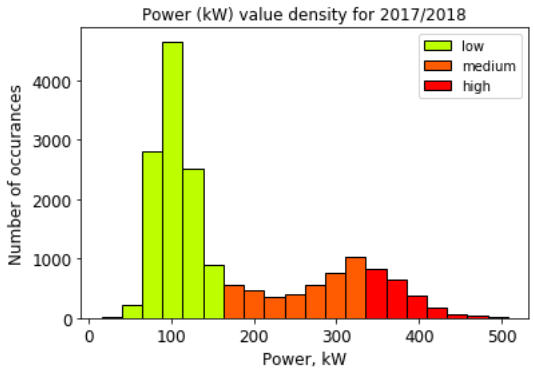

Figure 2: Hourly power consumption (KW) histogram for the Civil building, from 2017 to 2018. Two highly populated regimes of consumption with a transient one in the middle are distinguishable - a bimodal type distribution.

It is clear from a quick analysis that there are two immediate regimes of consumption - a peak higher one and a flat lower one. The two consumption regimes constitute a bimodal-type distribution with a lower transient regime in between. Most counts lie on the 50-150 kW range, with half of the values laying under the 110 kW mark. Then, we study the 10-day consumption pattern by hour, in Civil building, so as to understand the type of consumption cycle (Figure 3).

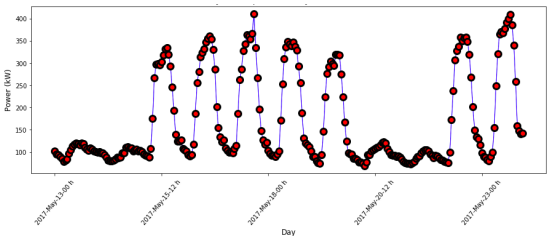

Figure 3: Scatter plot of the hourly power consumption (KW) pattern for the Civil building from 13 to 23 of May 2017. Two patterns of consumption arise: normal operations days where the power consumption peaks at around midday and weekends/holidays when there is significant power consumption reduction.

We can see that there are significant consumption drops on weekends and we could retrieve that this behavior would also occur on school holidays (mainly in the month of August, when there is little to no activity in the campus). We then experience two types of consumption patterns: business days and holidays (where the values drop to around 25% of the peak power drain of business days). All this inferences were verified to be valid for the year of 2018 too.

Re-sampling the data on a weekly basis, for both years, we could clearly identify the school holidays drop pattern (in January and August) and even a middle, less pronounced plateau regime on exam season (January/February and June/July), coming from idle operations in class buildings.

Finally, we study daily behavior in terms of mean hourly power consumption, as seen in Figure 4:

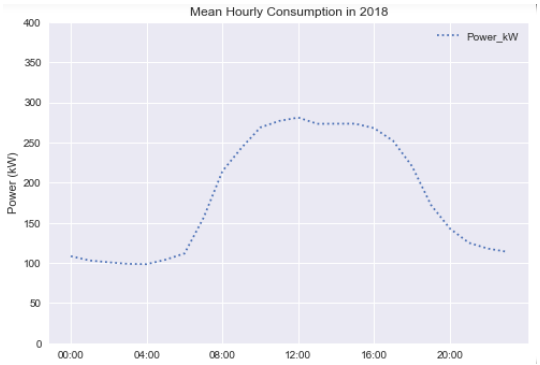

Figure 4: Mean hourly power consumption at Civil building, in 2018. A typical power consumption curve goes up in the morning, peaks at lunch time and starts going down again until the evening.

As expected from Figure X(a de cima), the power consumption begins to rise until around 11 AM, where it reaches a plateau of peak consumption. Around 5 PM, the power drain begins to drop down until it reaches idle levels. We have an idle regime drive throughout all days in which there's little to no activity in the campus. In particular, we have seen the Civil building mean hourly energy consumption gone up by around 20% from 2017 to

2018 ($\approx$ +30kW average offset).

2.1.2 Clustering and Feature Selection

Building an Hierarchical Clustering Dendrogram and retrieving kMeans [24] silhouette scores using Python's *scikit-learn* [5] library confirms that we have two main data clusters.

This way, we establish a proper plateau plot of both consumption regimes, differentiating them by cluster labels. The result plot is presented below, on Figure 5.

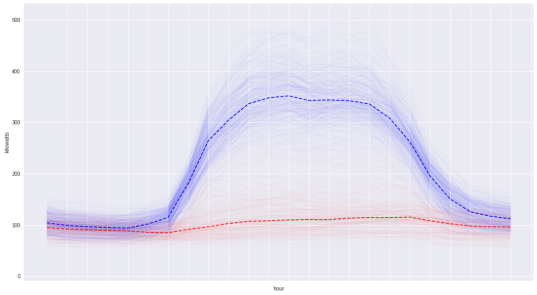

Figure 5: Plateau plot for hourly power consumption at Civil building in both years, during 24 hours, with the two highlighted consumption trends. Typical operation day (in blue) and lower operation day (in red).

Addressing feature selection, we add a parameter *WeekDay* (= 1 on week days and =0 on weekends) to our feature dataset, distinguishing between weekend and business day consumption. In addition, as we experience hourly and monthly periodic variations, two feature columns which convert *timestamps* into discrete values of hour and month are also loaded in the dataset. Most of our features, such as in other studies of the kind [15], are meteorological parameters, which are known to have an impact on energy consumption and positively influence prediction models. We got our data for the period 2016-2019 from IST's Meteorological Services [4]. Their data files include the following features:

1. Temperature (in degrees Celsius)
2. Relative Humidity (in percentage, %)
3. Wind Direction (in degrees)
4. Wind Gust (in m/s)
5. Wind Speed (in m/s)
6. Solar Radiation (in W/m^2)
7. Atmospheric Pressure (in mbar)
8. Precipitation (in mm/h)
9. RainDay (=1 in a day with reported precipitation, =0 otherwise)

To conclude, we add feature columns *Power - 1* and *Power - 2* correspondent to the power consumption of the previous two entries, that is, of the previous two hours. We shall further explain the use of these features upon model loading.

To find the right balance between model complexity, computational time and accuracy, we deployed feature selection algorithms. We used all three approaches to feature selection in order to improve our feature extraction decision. Starting by the filter methods, we use the kBest routine, which uses an ANOVA classifier function, available in Python's *scikit-learn* library [5]. The highest scores are laid on this following table:

Feature	kBest Score
Power-1	44,2
Precipitation	6,6
Solar Radiation	1,68
Temperature	1,39

Table 1: kBest algorithm highest scores and features for k=4.

Moving towards the wrapper approach, using RFE (recursive feature elimination) supported by a linear regression model [5]. Choosing to find the two main important features, we get the following affinity ranking:

Feature	RFE Affinity
Wind Gust	1
Wind Speed	1
Hour	2
Power - 1	3

Table 2: RFE algorithm 2-fold highest ranking features.

Finally, the ensemble approach is deployed using an Extra Trees Regressor [5] to scan for feature importance. We have listed those results on the next table.

Feature	Feature Importance
Power - 1	8,34E-01
Solar Radiation	1,02E-01
Hour	4,64E-02
WeekDay	6,74E-03

Table 3: Extra Trees Regressor feature importances.

As such, we shall inject in our model the following features, which scored as important to all three methods: **Solar Radiation**, **Power - 1**, **Hour** and **WeekDay**. These parameters are expected to have an impact on IST buildings' energy consumption. While nearly all of the correlations found were self-explanatory, the Solar Radiation feature was found to be a good proxy value of the power consumption, going up in the morning and, afterwards, going down until the end of the day.

2.1.3 XGBoost Regression Model

In order to build our prediction regressor model, several tests were conducted as to assess which one performed best using our data and features. The elimination criteria was based on the common metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The models verified under a test size of 15% for the concatenated data of both 2017 and 2018 were: Linear Regressor (LR), Random Forest Regressor (RF), uniformized Random Forest Regressor (uRF), Multi-layer Perceptron Regressor (MLP - Neural Network) [5] and XGBoost [8]. In all four buildings, XGBoost outperformed the other models, except for the North and South Tower, where the absence of some data points is expected to might have impacted the results.

Upon using the previously trained model, the Power - 1 feature was replaced in its essence. Instead of representing the previous hour power consumption, and since the model should be used *a posteriori*, it was actively loaded as the previous year consumption at the same hour, day and month. We expect this difference to help us build the model in a quicker and non-recursive way. The error results for the XGB model which was used to determine our 2019 baseline are listed on the table below. The results were averaged from 10 runs of the model, for each building, at a test size of 12,5%, which was found to minimize errors.

Building	$\overline{17}$ (kW)	$\overline{18}$ (kW)	MAE (kW)	MSE (kW^2)	RMSE (kW)	MAE (rel.)	CV(RMSE)
Civil	164,3	183,3	8,23	194	13,5	0,047	0,078
Central	189,2	182,4	7,27	140	11,8	0,039	0,064
N. Tow.	102,5	114,4	8,88	294	17,1	0,082	0,158
S. Tow.	177,6	173,9	17,5	1393	37,2	0,100	0,212

Table 4: IST's buildings average hourly power consumption for both years, XGB basline model error parameters (MAE, MSE and RMSE), and error parameters relative to the 2-year average (MAE and CV(RMSE)). The behaviour at the Towers shows that additional features should be considered, to lower the CV(RMSE). The model was trained with a concatenated 2-year-long data set, which helps reducing meteorological induced variability.

The model feature importance percentages are displayed on the table below, to show us how deeply did our input variables influenced our predictions. We conducted ten rounds of modeling as to better retrieve the final averaged feature importance for each building and a globally averaged feature importance.

Building	S. Rad.	Power-1	Hour	WD
Civil	0,018	0,884	0,092	0,006
Central	0,022	0,869	0,102	0,006
N.Tower	0,017	0,88	0,094	0,009
S.Tower	0,027	0,846	0,111	0,016
Average	2,1%	87%	10%	0,9%

Table 5: Feature importances for each building and global averaged feature importance, in percentage. Power - 1 and Hour are, hence, the most relevant features in our study. WD stands for WeekDay.

We now take a look at the buildings feature correlation heat map (Fig. 6), built from all initial features, in order to explain and validate our feature selection, as well as to confirm or undertake further adjustments to the model. We used Python's *corr()* function together with *Seaborn* to build this map.

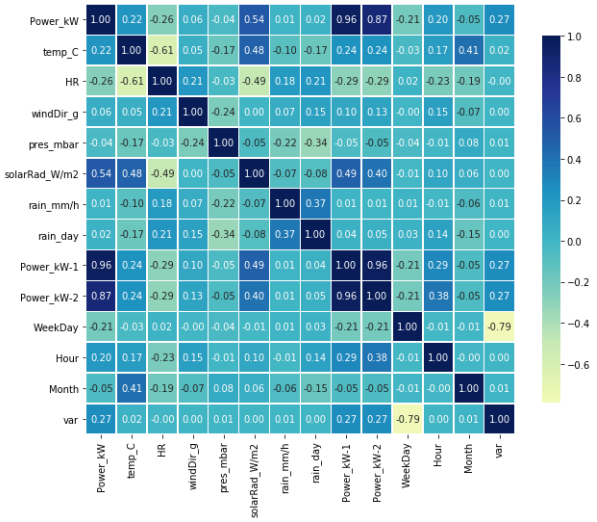

Figure 6: Civil building input feature correlation heat map. Here, "WeekDay" stands for the number of the day relative to Sunday while "var" stands for the used *WeekDay*

In fact, we are using the top four correlated features to our output parameter "Power_kW" (apart from Power - 2, which would turn out redundant), which boosts our model's reliability. Considering the Towers, we are getting higher error parameters. In the South Tower heat map, we can see that there's a higher correlation with temperature, which was confirmed by the North Tower heat map. This can be explained by the building's architecture (glass window coated tower) and HVAC systems,

which is a big contributor to the total load. To improve the model, several training rounds were conducted admitting the temperature as a feature, to see if it would reduce the error. The result came out to be negative, with error factors increasing upon the consideration of temperature. It is then assumed that the lower accuracy of the tower models can be due to the lack of data points, because of faulty smart meters on both towers. Concerning Civil and Central, the ECM impacted buildings, we can confirm that the error parameters are good enough to effectively establish a baseline model, with CV(RMSE)s under 8%. In addition, the 5-fold cross-validation r2 scores yielded a mean value of 98% for both buildings.

2.1.4 Savings M&V

To assess the measured savings, we integrate all parts of our data model. Loading the model with the IST's meteorological features from the year of implementation (2019) and plugging previous year's consumption data, at the same time and day, as *Power - 1*, the previously trained model retrieves its value predictions for the hourly consumption at each building. We then compare it to the actual smart meter retrieved consumption data for the year of implementation, on a hourly basis. To calculate the CO_2 emissions reduction, we use the value obtained from [7] [9] of 0,265 Kg/kWh, valid for Portugal.

We know that ECMs have been implemented across the Civil and Central buildings during the month of April, by *Campus Sustentável - IST* [1]. It focused on changing the lighting scheme to more efficient LEDs on the highest consuming buildings. The reporting period for the retrieved savings in this section was considered to be from 01/06/2019 to 31/12/2019. This way, we leave one month for ECM impact stabilization and then analyze a period consisting of three months of normal scholar activity, three months of holidays and one month of exam season, so we can better estimate an overall savings percentage, for an university campus. Formulas 1, 2 and 3 are used for calculating percentage savings, actual savings and the error parameter, in which M stands for Model, A for Actual, S for Savings and Avg for the model average value, followed by a table in which these calculated values are displayed. The sums are performed on a hourly basis and the used average RMSE was the value present in Table X, for each building. Representative plots of the model behaviour versus predicted data, hourly savings and CO_2 kilograms saved for a week in October 2019 at the Civil building are also displayed in Figures 7 and 8.

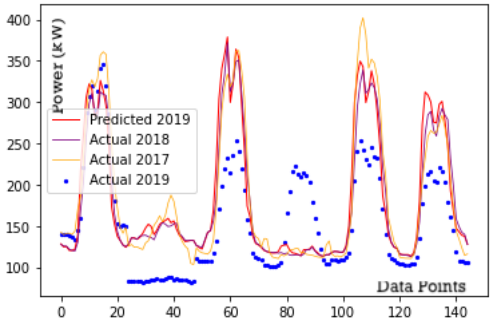

Figure 7: Model hourly behaviour for a week in October 2019 at the Civil building.

Figure 8: Hourly energy savings and hourly CO_2 savings (b) for a week in October 2019 at the Civil building.

$$S = \sum_{1/6/19}^{31/12/2019} (M - A) \tag{1}$$

$$S(\%) = \frac{\sum_{1/6/19}^{31/12/2019} (M - A)}{Avg} \times 100 \tag{2}$$

$$Error = \frac{\overline{RMSE}}{Avg} \times 100 \tag{3}$$

After ten averaged runs, the retrieved savings results were:

Building	Savings (%)	Savings (MWh)	CO_2 saved (ton)
Civil	16,9 $\pm$ 7,3	147,8	39,2
Central	20,6 $\pm$ 6,3	179,6	47,6

Table 6: Civil and Central buildings reported savings from 01/06/2019 to 31/12/2019, according to our model.

2.2. Blockchain

In order to deploy a ledger algorithm, an IBM Developer public blockchain prototype was used. The code and its significant updates are hosted on *github* [3] and a thorough tutorial on how to program and use it is present in [10]. This application shall allow users to share information by posting on the network using a simple web interface (Fig. 9). The used web framework was Flask [2]. In this section, we shall explain the code and adjustments that have been made, in order to deploy our M&V solution and properly assess EPC execution.

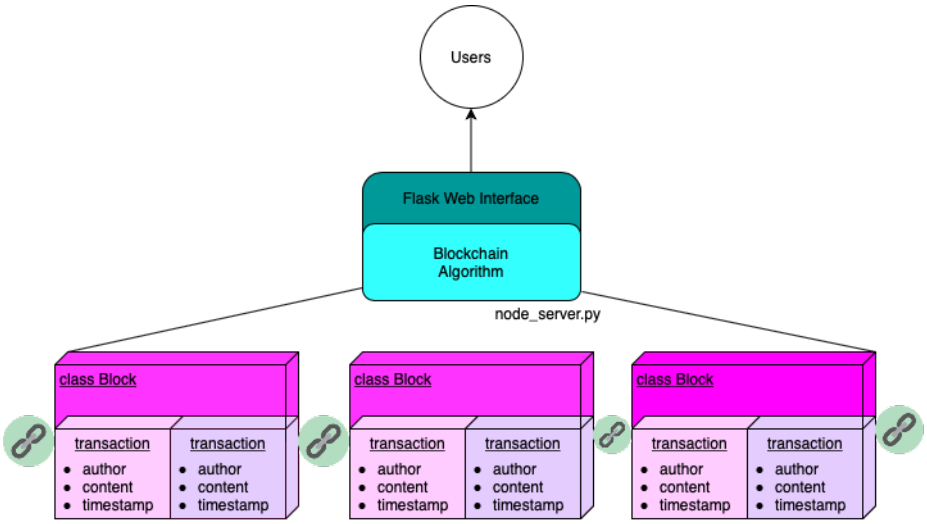

Figure 9: Blockchain network scheme.

2.2.1 *Blockchain* and *Block* classes

The Block object is initialized taking an index (which serves as an unique identifier), the transactions array, a time stamp, the previous hash string and a default zero *nonce* as arguments.

In a network like this one, we want to prevent transaction data tampering. To encode transaction information, we use a cryptographic hash function. A hash function is a function that takes input data (of any size) and retrieves data of a fixed size from it (the hash), which is used to identify the input [10]. These functions have to be easy to compute, deterministic (in the sense that the same data must retrieve the same hash) and uniformly random regarding changes in input. This way, it is virtually impossible to figure out the input data from the hash (the only way being to compute all possible input combinations) but, having the input and the hash, one can simply pass the input through the hash function to verify a provided hash. This is known as effort asymmetry.

In the context of our application, a compute_hash() function is responsible for encrypting the data referent to the transaction string and, in our case, encoding the savings information text to be put on the block. The cryptographic hash function used to encode our strings was the Secure Hash Algorithm 2 (SHA-256, [6]) (256 bits), designed by the US National Security Agency (NSA), considered to be safe.

In order to avoid chain tampering, we chain the blocks together by having each of them store the previous block hash. This way, we make sure that any change in the previous blocks invalidates the whole ledger.

In the *Blockchain* class, we can find the initializer function, the genesis (first) block creator and the last block retriever property. The initializer creates two empty arrays: *chain* and *unconfirmed_transactions*. To initialize posting on the chain, we have to create the first block using function *create_genesis_block* that joins an empty *block* object to our current chain with index ”0”, so as to ensure coherence between blocks.

At this point, it is still possible to tamper with data by just changing the previous block and easily recomputing all the blocks that follow. We avoid this by exploiting effort asymmetry upon calculating the hash, making it difficult and random. In our case, we add the constraint that our hash should start with n leading zeros. To prove that this computation was performed, we store a nonce (dummy) variable on our blocks, that is incremented until the calculated hash satisfies our constraint.

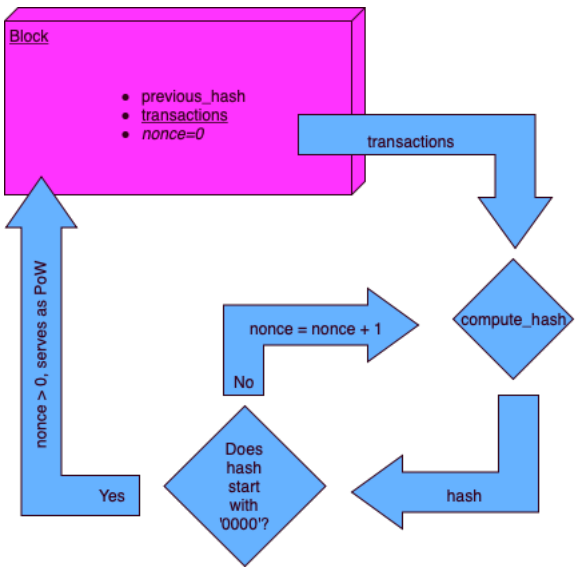

Figure 10: Proof-of-Work.

The *proof_of_work* function certifies that the hash

is retrieved following an hash header constraint and its difficulty ('0000'), saving proof of computation afterwards (Fig.10). Correspondingly, we have a Boolean function called *is_valid_proof* that checks if the computed hash for the block matches its input content. After that, the *add_block* routine receives a block and its hash which then appends to the main chain, after it confirms previous hash coherence (perserved order of transactions) and computation proof (data tampering). In addition, our *add_new_transaction* function appends new transactions to the unconfirmed_transactions array.

The process of appending the unconfirmed transactions to a block and computing proof-of-work is called mining. Once our hash constraints are met, a block is said to be mined and can be added to the ledger. In most cryptocurrencies, this mining computational activity is compensated by a share of cryptocurrencies [10].

The *mine* function appends all unconfirmed transactions to a block and adds it to the chain, after undergoing PoW and previous block hash coherence, resetting the unconfirmed transactions' array.

The *check_chain_validity* routine is used on the consensus mechanism for different chains. It checks if the computed hashes for the blocks in the chain match what they were supposed to, according to our cryptographic scheme.

On a longest-chain (more produced work) consensus approach like ours, we thereby validate each conflicting chain while checking what is the longest one. This method is used on the further explained web-interface function *consensus*[1] for that exact purpose.

2.2.2 *Flask* framework

Concerning the web interface, we explain how this solution handles the blockchain instructions on the client side, by using submitting HTML requests through *app routes*, using Flask [2] to create a REST API that invokes operations in our blockchain node [10].

To add new transactions to a block, the *new_transaction* routine saves the author and the content of our transaction (hence, our savings), recording the current time stamp.

Aditionally, our adapted version of this blockchain requests the RMSE for a given used baseline model and the mean energy consumption value during the baseline period (in kW), in order to confirm that our model CV(RMSE) is satisfactory. This way, only transactions that guarantee a certain level of model accuracy can be added to a block. In our solution, we chose demand a CV(RMSE) of under 15%. The submission of these parameters by the nodes is further discussed on Chapter 5.

If intentional manipulation or network latency occurs, the copy of the chain in some nodes can be compromised and differ from the other nodes. In that case, the network needs to agree upon some version of the chain to maintain integrity [10].

The *consensus* function clarifies just this latter point, by checking chain length when chains of different nodes appear to diverge. This way, it's agreed that the longest chain corresponds to largest amount of work (PoW) done and, hence, valid.

In order to submit the mining command, the *mine_unconfirmed_transactions* app route uses the *mine* function of the *Blockchain* class, making sure we have the longest chain before anouncing it to the network, which enforces our consensus criteria.

After a certain block is mined by some node, we need to add it to each nodes' chain. We do that by defining the *verify_and_add_block* function, which loads the PoW to the *Blockchain add_block* function.

We need a way for any node to announce to the network that it has mined a block, so every node can update their blockchain [10]. This way, the other nodes can simply verify proof-of-work and add the mined block to their respective chains. Because of this, the *announce_new_block* method is called after every block is mined by a certain node.

Finally, to establish a network, we need to be able to securely register new nodes and put them up to date regarding the valid chain. The *register_new_peers* and *register_with_existing_node* functions guarantee those exact methods, enabling a node to register new other trusted nodes. This last method will allow the remote node to add a new peer to its list of known peers and initializing the blockchain of the new node with that of the network-trusted node.

3. Implementation and Usage

We now integrate the model and blockchain components to produce automated savings posts that can be seen in an HTML page. Here we explain how to run the *localhost* ports that host our two blockchain nodes and how each building savings are posted on each node, after calculation. The average script execution time, in seconds, is listed below on Table 5.1.

	central_node.py	*civil_node.py*
Exec. time (s)	5,576	5,416

Table 7: Average execution time (s) for each script.

[1] We highlight the difference between block coherence, referent to previous hash matching, and conflicting chain consensus, in which we design a decision mechanism to chose between two different chains, hosted by different nodes.

3.1. Blockchain Node Server and Application

To deploy our application, we first need to assign our *flask* application to the *node_server.py* script. After that, we run the server ports in which we will be able to post savings informations, using the *run* command. We shall initialize two *localhost* ports, one for each building to post on.

After this, an instance of our blockchain node is running at *localhost* ports 8000 and 8001. On a different terminal session, we can now run the *curl* commands to register a new node (port 8001) with a proxy node (port 8000) . Symmetrically, because of application constraints, we need to do the same thing to register the first node (port 8000) with the latter one (port 8001) (Listing 5.2). This will make the node at port 8000 aware of the nodes at port 8001 and vice-versa [10]. New nodes will also sync their chain with the existing node so that they are able to participate in the mining process.

Now, we just need to run the *run_app.py* front-end application on a different terminal session. This will start our HTML interface at http://localhost:5000, in which we can visualize our chain transactions. By default, this application syncs with *localhost* port 8000, but that parameter can be changed by updating the *CONNECTED_NODE_ADDRESS* field in the *views*.py file [10].

3.2. Savings script

The *central_node.py* and *civil_node.py* scripts which retrieve the energy consumption savings information were built to: 1) treat the existing datasets and train a XGB model 2) load the reporting period features to the same model 3) retrieve the savings information post (string) and model CV(RMSE) and 4) automatically post weekly savings information on our blockchain, corresponding to the same week in 2019.

To make automatic posts we use *python*'s *schedule* library, forcing code execution every Sunday. We do this by defining a function which calculates and posts savings and scheduling it to post in a given weekday.

To post information on the nodes, we use HTML requests and the *httplib2* library. They retrieve information about the current week consumption in the reporting year (2019) and previous years (2017 and 2018). Our posts also configure two request fields, "Mean" and "RMSE", in order for the blockchain to verify model CV(RMSE) upon posting.

It was possible to verify that the value variation between trained model results in different script executions was under 1%.

3.3. HTML page

In our visual interface, we can visualize our savings information content, the "author" building and the timestamp of the node post. There are buttons to request mining, returning to homepage and refreshing the current page. Additionally, there are two disabled features, which can be used to further improve UI/UX: a Reply button, on each post, that can be used to insert observations/comments to each week's savings information and input boxes, which may allow us to submit a post directly on the HTML page through the *CONNECTED_NODE_ADDRESS* (default port 8000). Notice that the input box posts are (currently) disabled since we lack the "RMSE" and "Mean" model accuracy fields which validate our transactions.

4. Conclusions

4.1. Achievements

We were able to deploy a blockchain solution that accurately estimates and stores savings in a transparent manner. The validation mechanism is associated to the accuracy of the forecasting model, which secures a framework for EPCs to be audited in a clear, safe and trustworthy manner.

M&V 2.0 technology integration allows stakeholders to better examine and estimate energy efficiency improvements locally and globally. These represent the most significant reduction in energy-sector CO_2 emissions globally and help us get on track with the three energy-related SDGs, saving energy and balancing demand with supply side improvements.

Considering the lighting ECM deployment at the two main buildings of the IST *Alameda* campus - Civil and Central - we estimate savings on a weekly basis, throughout the reporting period. Resorting to a trained XGB [8] baseline model, we have reached a savings level of 16.9 $\pm$ 7.3% and 20.6 $\pm$ 6.3%, in the Civil and Central buildings, respectively, and a cumulative 86.8 metric tons reduction in energy related CO_2 emissions, over a reporting period of 7 months (Jun-Dec 2019). The models yielded an average CV(RMSE) of 7.8% and 6.4% for the Civil and Central buildings, respectively.

We store our savings information in safe blockchain nodes, using an adapted version of the IBM Developer [10] blockchain algorithm, which only validates savings posts after verifying that the baseline model's CV(RMSE) is under 15%. The information is displayed to the user via a Flask web interface, which interacts with each node server on the established network.

Future adjustments and limitations are discussed in the sections below, so as to drive future developments. Our results allow us to state that our endeavor objectives were met and that this was

a promissing innovation in the field of EPC auditing.

4.2. Limitations and Future Work

During the development of this first prototype, we've came across some limitations which prevent us from presenting an universally accepted solution.

Regarding the baseline model, it is known that the deployment of XGBoost regressor models can be more computationally costlier than similar performing algorithms, like Random Forests, which can raise concerns of scalability. In this initial study, we focused on delivering the most accurate model in detriment of the most efficient one, which should be taken in consideration when engaging in wider applications.

In what concerns the integration of savings calculations with blockchain node servers, there is a need to establish an actual network and securing a public-private key cryptography scheme [10], instead of running the scripts on *localhosts* and having virtually every possible user posting/changing data on our buildings' ledger. This way, the posts can be added another level of security, after which some data protection work is recommended on the model side too, when handling key parameters, like the model CV(RMSE).

Focusing on energy data and information, we didn't find any documentation available regarding the deployment of these ECMs that we could compare our results against. The baseline model was proven to behave satisfactorily, well below the 25% used as accepted standard for this industry, but we exhort the community to further boost smart-meter deployment in public buildings and thoroughly documenting ECM implementations in the future, helping to maintain data consistency.

For the prosperity of technologies like this one, we further recommend the standardization and debureaucratization of energy savings and EPC procedures, making savings knowledge easily tangible to the end-user. Further adjustments can be employed in smart cities applications of this solution which additionally would comprise water savings and renewable energy production/storage.

References

[1] Campus Sustentavel - IST, 2020.

[2] Flask Web Interface, 2020.

[3] Ibm Developer Blockchain Algorithm - GitHub, 2020.

[4] Meteo Tecnico, 2020.

[5] scikit-learn Library, 2020.

[6] Secure Hash Algorithm-2 - Wikipedia, 2020.

[7] APREN. Boletim Electricidade Renovável, 2019.

[8] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. New York, USA, 2016. Association for Computing Machinery.

[9] E. Comercial. Origem da energia, 2020.

[10] I. Developer. IBM Developer Blockchain Algorithm, 2020.

[11] E3P. Energy performance contracting.

[12] U. N. Economic and S. Council. *Special edition: progress towards the Sustainable Development Goals - Report of the Secretary-General*. 2019.

[13] P. Fennell, P. Ruyssevelt, and A. Smith. The impact of measurement and verification option choice on financial returns for clients in energy performance contracts. 2017.

[14] J. N. Ferrer. Leveraging funding for energy efficiency in buildings in south east europe, March 2019.

[15] M. G. Fikru and L. Gautier. The impact of weather variation on energy consumption in residential houses. *Applied Energy*, 144:19–30, 2015.

[16] M. Frangou, M. Aryblia, S. Tournaki, and T. Tsoutsos. Potential of energy performance contracting for tertiary sector energy efficiency and sustainable energy projects in southern european countries. Springer, 2018.

[17] M. Frangou, M. Aryblia, S. Tournaki, and T. Tsoutsos. Renewable energy performance contracting in the tertiary sector standardization to overcome barriers in greece. 2018.

[18] Ö. Gürcan, M. Agenis-Nevers, Y.-M. Batany, M. Elmtiri, F. Le Fevre, and S. Tucci-Piergiovanni. An industrial prototype of trusted energy performance contracts using blockchain technologies. IEEE, 2018.

[19] J. Hwang, M.-i. Choi, T. Lee, S. Jeon, S. Kim, S. Park, and S. Park. Energy prosumer business model using blockchain system to ensure transparency and safety. 2017.

[20] IEA. *Energy Service Companies (ESCOs)*. 2018.

[21] IEA. *Global Energy and CO2 Status Report 2019*. 2019.

[22] IEA. *World Energy Model*. 2019.

[23] A. Khatoon, P. Verma, J. Southernwood, B. Massey, and P. Corcoran. Blockchain in energy efficiency: Potential applications and benefits. 2019.

[24] K. Krishna and M. N. Murty. Genetic k-means algorithm. 1999.

[25] NREL. *International Performance Measurement and Verification Protocol*. NREL, 2002.

[26] B. Richter, E. Mengelkamp, and C. Weinhardt. Maturity of blockchain technology in local electricity markets. IEEE, 2018.

[27] J. A. Shonder and E. Morofsky. Best practice guidelines for using energy performance contracts to improve government buildings. 2010.

[28] M. Zhang, M. Wang, W. Jin, and C. Xia-Bauer. Managing energy efficiency of buildings in china: A survey of energy performance contracting (epc) in building sector. *Energy Policy*, 114:13–21, 2018.