Climate Change Final Project

Estimating Residential Energy Consumption with Machine Learning Models

Team Jellycats

Brenda Liu, Chang Li

NEW YORK UNIVERSITY

05/05/2023

1. Introduction and Background

1.1 Problem Being Tackled

This research aims to deep dive into various models to predict residential energy consumption for the United States.

1.2 How the Problem Relates to Climate Change

Building energy consumption has historically been one of the most significant contributors to climate change, as electricity and heat even accounts for 30% of the greenhouse gas emissions. The energy used in buildings for heating, cooling, lighting, and appliances is predominantly derived from the burning of fossil fuels, which releases CO2 and other greenhouse gasses into the atmosphere. These emissions trap heat and contribute to global warming, leading to climate change.

The demand for energy in buildings is expected to increase significantly in the coming decades, as the global population continues to grow and urbanization accelerates. To address possible suggestions on the way to reduce energy consumption of buildings in response to mitigating its impact on climate change, data from the United States Energy Information Administration (EIA) on residential energy consumption [2] was collected for analysis.

2 Related Work

Previous research have tried to tackle this problem. Robinson et. al. presented similar objectives in “Machine learning approaches for estimating commercial building energy consumption” [1]. This project takes inspiration from Robinson et. al. and takes into account the specific machine learning models used by them.

3 Data and Methods

3.1 Data Source and Features

The dataset used for this project is Residential Energy Consumption Survey (RECS), downloaded from the United States Energy Information Administration (EIA). It contains 5686 rows and 759 columns, which represents the data from a national sample survey that collects energy-related data for 5686 housing units with 759 potential features. Potential features include general information about the building (room counts, time built, etc.), how the building was used (appliances in the building, their usage frequencies, what type of fuel they rely on, etc.), as well as replicate weights, and the cost of energy sources at the time of the survey.

Notably, the dataset contains 211 features that start with the character “Z”. According to the codebook, these features are imputation indicators. They denote whether the value in the corresponding column was imputed or not. For example, column “CELLAR” means whether the housing unit is over a basement, and the following column “ZCELLAR” means whether the previous value in “CELLAR” was imputed or not.

Several target variables are available to choose from. The dataset contains detailed information about the energy consumption for each type of activity, such as heating, cooling, and cooking. This project aims to estimate the total energy consumption of buildings, so the column “TOTALBTU”, representing total energy consumption in BTU, was chosen as the target variable.

At the same time, all variables that directly relate to the target variable were removed from the feature set. This is to prevent the model from simply calculating the total energy consumption based on other consumption metrics. Only features that directly relate to the building itself were included for our models.

3.2 Preprocessing Steps and Reasons

The dataset is imported into a jupyter notebook with pandas. Several cleaning steps are involved.

First, all rows with missing data were dropped. The dataset still contains 3304 rows after this step. This was a worthy move for us, because the data source does not mention why the values were missing, and there is no explanation about why an imputed value was used, and the corresponding imputation flag set. Because of this reason, all the imputation flags were also dropped from the dataset.

Next, all categorical variables were identified and transformed into dummy variables using one-hot encoding. The codebook describes whether a column is of type “Numeric” or “Character”, but this does not take into account the fact that most categorical variables are encoded as numbers, and those columns have the type “Numerical”. Therefore, all columns in the feature set were manually inspected, and each column deemed as categorical were included into a list in the jupyter notebook. Those columns transformed into dummy variables using the pandas “get_dummies” function.

Finally, the codebook mentions that many columns can have the values -2, -8, and -9, denoting that the feature for that house is not applicable, the respondent refused to answer the question, or the respondent does not know the answer. Considering that most of the features in the dataset are indicators, with a value of either 0 or 1, values like -8 and -9 would severely skew the results of our models, because they operate on a numerical basis. Therefore, all values -2 were replaced with 0, and all values -8 and -9 were replaced with 0.5. This is because if a feature is not applicable for the building, it’s most likely that the feature cannot exist, or does not make sense for that specific building to begin with, so they can be interpreted as “no” or “does not” for the purpose of this project. For the situations where the respondent refused to answer or did not know the answer, the values were replaced with 0.5 because the real situation is unknown.

After all the cleaning steps, the final feature set contains 3304 rows and 477 columns.

3.3 Models and Reasons

Models chosen for this project are Linear Regression, XGBoost, Random Forest, AdaBoost, Linear SVR, SVR, K Neighbors Regressor, and Multi Layer Perceptron. All models were implemented with scikit-learn, with the exception of MLP using PyTorch. Those models were

chosen because they are the most popular methods for regression. Three models related to trees were considered, namely Random Forest, AdaBoost, and XGBoost. This is because decision trees are more suitable for datasets with lots of indicator variables, where they behave more like categorical ones with discrete value sets.

In addition to the regression models, PCA and t-SNE were also used in the project. PCA was chosen to condense the dataset into fewer principal components, and determine the performance of previously mentioned models on the compressed dataset. t-SNE was used to potentially identify clusters of buildings, where they share similar traits.

3.4 Hyperparameter Selection Process

Hyperparameter tuning was employed for XGBoost, Random Forest, AdaBoost, and MLP. For the three models implemented with scikit-learn, the wrapper GridSearchCV was used for automatically determining the optimal parameters for each model. The parameters "n_estimators" and "max_depth" were tuned for them. The hyperparameters were varied on a scale of 10, and GridSearchCV wrapper automatically chooses the best combination of hyperparameters based on mean squared error performance. The hyperparameter "estimator" was also tuned for AdaBoost, in order to try decision trees with different max_depths as the base estimators for this model. Importantly, decision trees with max_depth of "1" and "None" are both included in the search. These two trees represent the least and most amount of estimation a tree can make. With a max_depth of 1, decision trees are basically binary with only 2 levels and 2 leaves. This is to prevent overfitting, by using simpler trees but more of them. On the opposite side, trees with max_depth of None can expand until all the leaves are pure. This setting was included in the search to investigate the performance on the test set if the trees were given the permission to do the best estimation it can on the training set.

For MLP, the hyperparameters searched were the activation function and the number of layers. A loop was used to test each hyperparameter combination, where a separate model was trained and evaluated to record the performance metrics. Five activation functions and network depths of 1, 2, and 3 were included in the search, for a total of 15 models.

3.5 Evaluation Metrics and Reasons

Three evaluation metrics were chosen for this project: root mean squared error, mean absolute error, and variance explained. Root mean squared error will be the primary metric for evaluating the models. Variance explained will also be considered, as it helps understand how well the model performs overall, compared to the optimum result of perfectly estimating energy consumption.

For the scikit-learn models, the criterion of squared error was chosen whenever possible, and the three metrics were automatically evaluated for each model class on the particular model with the best parameters.

For MLP, the output from the loss function being too high prevented the training on squared loss, and the models were trained on absolute loss instead. They were still evaluated using squared loss to make them comparable to the scikit-learn models. Normalization techniques and batch training were employed to potentially solve the issue. Normalization with logarithm of base 10 did not prevent loss from reaching NaN, but batch training was successful at dealing with this problem. Nevertheless, the models were still trained on absolute loss, and they were at a disadvantage compared to other models.

4 Results

4.1 Comparison Across Models/Parameters

The three metrics mentioned earlier are recorded in a pandas DataFrame. These results are shown in Table 1. The root mean squared errors for MLPs are shown in Table 2.

	Mean Absolute Error	R^2 Score	RMSE
XGBRegressor	21063.03	0.637	28759.41
LinearRegression	21363.96	0.630	29026.67
RandomForestRegressor	20486.36	0.619	29432.47
AdaBoostRegressor	20657.92	0.615	29599.38
LinearSVR	22486.26	0.562	31572.34
XGB on PCA with 100 PCs	23433.49	0.546	32154.49
XGB on PCA with 10 PCs	25133.11	0.481	34351.71
KNeighbors	25216.30	0.480	34390.11
SVR	37074.08	-0.015	48053.68

Table 1: Metrics for Conventional Models

	Sigmoid	ReLU	Tanh	ELU	None
1 layer	83383.12	40660.33	80351.00	44760.30	38296.13
2 layers	80409.13	57265.71	80357.50	55351.40	59556.51
3 layers	80401.06	99501.27	80361.50	99328.04	36383.06

Table 2: RMSE for Neural Networks

The scatterplots for the two best-performing models are included. Figure 1 is for XGBoost and Figure 2 is for Linear Regression.

4.2 Figures with Performance and Other Results

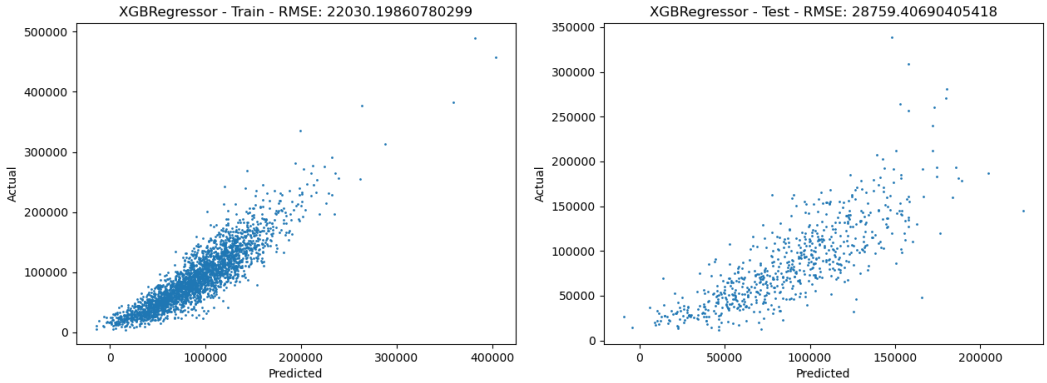

Figure 1: Scatterplots for Training and Testing from Linear Regression

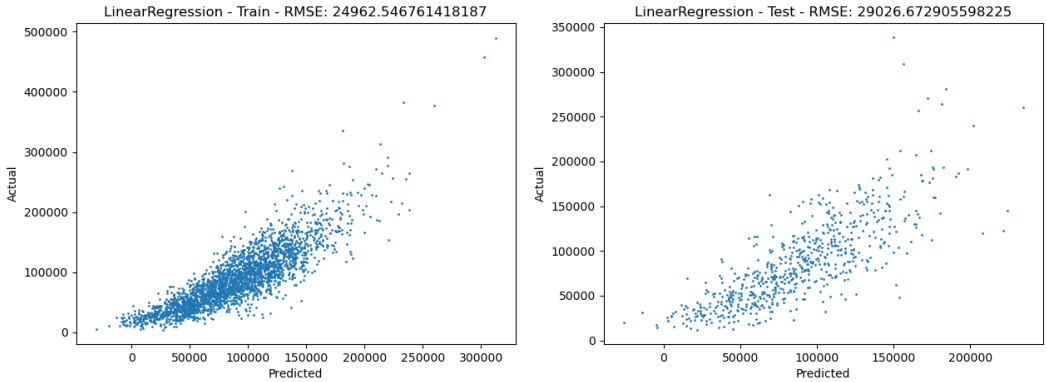

Figure 2: Scatterplots for Training and Testing from XGBoost Regressor

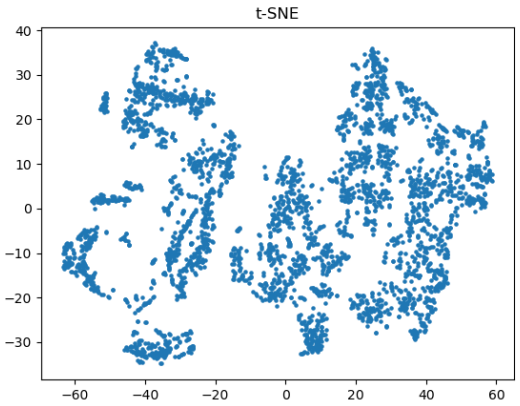

Figure 3: Scatterplot of 2 Components from t-SNE

5 Discussion

5.1 Best and Worst Models

The best models for this regression project are XGBoost, Linear Regression, Random Forest, and AdaBoost. Although XGBoost is at the top of the list with the lowest root mean squared error and highest variance explained, all four models performed comparably, with the worst root mean squared error being about only 3% higher than the best. Most notably from those models, Linear Regression requires very little time to train, and does not require hyperparameter tuning at all. This could be because most features in the cleaned dataset are indicators with simple value sets they can take from. As mentioned earlier in the model selection section, it's expected that models based on decision trees work well here for the same reason, and they do in this project.

MLP does not perform well for this project. Despite the fact that they were trained on absolute loss but evaluated using squared loss, they are still not a good fit and perform significantly worse than linear regression and tree-based models. It could very well be possible that it performs worse even if the issues were solved and trained on squared loss. MLPs have hidden layers with neurons and connections between each combination of neurons in different layers. This type of complicated model is best suited to model datasets with complex relationships between features and the target variable, requiring multiple layers with potentially logical relationships between the layers to correctly explain. The dataset used for this project does not contain such complex relationships. Making the problem worse, MLPs have a large number of parameters to train, with one for each connection between a pair of neurons. With the cleaned dataset containing about 3300 rows, there's not enough data for MLPs to train well and tune those parameters optimally. We can clearly see the trend that more layers in the MLP actually makes the performance worse. This could be due to the problem of too many parameters and not enough data.

5.2 Data Cleaning and Model Performance

An alternative version of the jupyter notebook was run without the data cleaning process mentioned earlier, only the variables directly related to the target variable were dropped.

Interestingly, performance of most models increased slightly with this “unclean” dataset, with XGBoost having a root mean squared error of about 27,000.

One possible explanation for this behavior is that the dropped columns contain sample weights. Sample weights denote how often this kind of building exists in the United States. In other words, a higher sample weight means the frequency of this kind of building is higher. Knowing that the target variable is skewed to the right, with many buildings consuming a rather small amount of energy and a few consuming a large amount, it’s possible that the models were catching onto this relationship and making predictions based on how often the building occurs. Even if this is not the case, dropping potential features would never improve a model’s performance if that model is tuned and trained adequately. In the worst case scenario where the feature is not predictive at all, the well-trained models would just ignore that feature and have a coefficient of 0 for it. Therefore, it makes sense that the best performing models in this project lose some performance after the data cleaning process.

Although removing sample weights made the performances worse, this restriction to the models was deemed necessary, as they were expected to estimate energy consumption based on the building itself, not on the distribution of buildings in a specific area or nation. This way, the trained models will have better potential performance if deployed on another dataset with data from outside the United States, potentially with a different distribution of energy consumption for buildings.

5.3 Feature Importance and Dimension Reduction

PCA was fitted onto the cleaned dataset, with n_components set to 100 and 10. XGBoost was then used to estimate the same target variable on the transformed dataset from the two PCA instances. From the performance summary table, the models performed adequately well, with root mean squared errors about 12% and 19% higher than the model trained on the full feature set. Considering that the full feature set contains 477 features, it’s impressive that PCA was able to reduce that set down to 10 principal components, and XGBoost can still estimate our target variable with only 19% higher root mean squared error. This result suggests that there’s severe

collinearity for our feature set, and 10 features was enough to capture the majority of variance in it. Fitting a PCA with only 2 principal components on the cleaned feature set yielded an explained variance of about 87%. This high number of explained variance further strengthens the point of collinearity within the feature set.

Considering the collinearity of the feature set, it's hard to evaluate the most predictive features in the regressions. This is because the permutation_importance function we employed for scikit-learn works by randomly permuting one of the features at a time, and training a separated model on the modified dataset, then calculating the performance loss of the modification. Since features are collinear with each other, removing one feature at a time does not prevent the model from knowing the underlying latent feature that's affecting all the collinear features, and it's very likely that the models can still make the same predictions based on the latent feature. As a result, outputs from permutation_importance are not representative of the true important features for this project. Nevertheless, the outputs make sense, in the fact that most of them are closely related to energy use, such as the size of the building, how many days in a year it requires heating or cooling, the number of appliances in the building, and how often they are used.

Fitting a t-SNE on the cleaned feature set with 2 components yielded a plot with roughly 7 clusters. However, t-SNE only preserves local structure but not global structure, so it's not possible to interpret the 2 components based on the positions and shapes of the clusters.

5.4 Unexpected Behavior from Linear Regression

Lastly, unexpected behavior was observed during the project, shown in Figure 4. Linear regression would sometimes perform well on the training set, with reasonable values for all three metrics, but predict mostly values of 0 on the test set, with a few outliers at values very close to 0. It's still unclear why this type of behavior occurs. Further debugging was not successful at pinpointing the cause of the issue. It's known that this behavior is isolated to linear regression, and it depends on the random seed and test size for the data splitting process for training and testing.

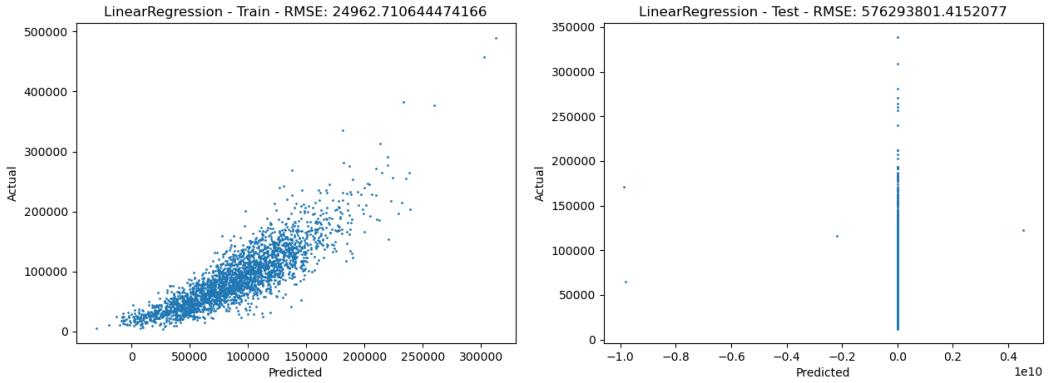

Figure 4: Unexpected Behavior from Linear Regression

6 Potential Future Work

6.1 Better Model Tuning

The Neural Networks used in this model were not adequately trained. The issue of loss reaching NaN could be solved using better normalization techniques. All the layers were using the same activation functions for models with 2 or 3 layers, but they don't necessarily have to. Further hyperparameter tuning could be done to experiment with more combinations of layer sizes, counts, and activation functions for each layer. Lastly, sample weights could be used for the Neural Networks to alleviate the problem of not having enough data.

During the testing phase of this project, the SVR and K Neighbors Regressor models were taking too much time to train, and would not complete within a reasonable time. Bagging wrappers were used as a result. Smaller and simpler models were trained on a sample of the dataset, and a number of them were aggregated together to represent the final model. The hyperparameters of those ensembles were not tuned well in this project, since they weren't suited for this dataset in the first place. This oversight is partly responsible for the poor performance of those models. Future work could devote more computational resources and hyperparameter tuning to better train those models.

6.2 Collinearity and Feature Importance

Severe collinearity exists in the feature set, and the permutation_importance function from scikit-learn was not ideal at determining feature importance for this project because it only permutes one feature at a time. Potential future work could determine which sets of features are correlated with each other. Packages similar to permutation_importance could be employed to automate that process, and permute the entire correlated feature group at the same time to better understand feature importance. However, it is worth noting that such work would need to interpret the meaning of the correlated feature group, and it is unsure whether such feature groups would be interpretable or summarizable into a single trait of the building.

6.3 Data Cleaning and Imputation

The data cleaning methods for this project could also be improved. The current cleaning process is still rather primitive, with only simple imputations and replacements. Future work could improve on the cleaning process and fill in more accurate data for the missing, imputed, and inadequately encoded values.

6.4 Alternative Validation Sets

The RECS project collects survey data from buildings, and it was first conducted in 1978, with iterations about 5 years apart in recent decades. This project used the microdata from 2015 to train the models. However, data from 2020 and previous years of the RECS project were also available for download, and future work could evaluate how the models trained on the 2015 dataset performs on the 2020 dataset.

References

1. Robinson, Caleb, etc. *Machine learning approaches for estimating commercial building energy consumption.* Applied Energy. Volume 208, ScienceDirect. 15 December 2017. https://www.sciencedirect.com/science/article/pii/S0306261917313429#s0060

2. Residential Energy Consumption Survey (RECS) Microdata. United States Energy Information Administration. 2015. https://www.eia.gov/consumption/residential/data/2015/index.php?view=microdata