Using Census Data to Predict Solar Panel Deployment

Eddie Sun: eddiesun@stanford.edu | Jeremy Chen: czm@stanford.edu
Brett Szalapski: brettski@stanford.edu

1 Introduction

New renewable energy sources require improvements to the current electric grid. The recent surge in the number of intermittent energy generation facilities, such as solar and wind farms, has resulted in a need for improved monitoring and control methods for the electric grid due to increased supply-side uncertainty. Mitigating the uncertainty in the amount of electricity privately produced would greatly increase power generation efficiency, resulting in less waste of fossil-fuel generated electricity.

One major component of supply-side uncertainty comes from residential solar panel installations. Today, installing solar panels on residential homes is easy and affordable, and will only become easier and more affordable as time progresses. As a result, it is difficult to know how many solar panels exist and supply power to the grid. If energy companies had more insight into this piece of the supply-side puzzle, they could better model an area's energy production and balance power plant production accordingly, resulting in lower energy costs and less environmental impact.

For this project, we implemented and optimized an artificial neural network (NN) and a support vector regression (SVR) algorithm to predict the number of solar installations in a given tract from census data. The input to the model consists of geographical and demographical characteristics, such as land area, average household income, climate data, number of residents of age 30-39, etc. The model takes these census data and then outputs the number of solar systems in a given tract. The model is trained using supervised learning on a labeled dataset. We also used the models and principal-component-analysis (PCA) to determine which features have the most influence on modeling the number of solar systems (i.e. which features are most strongly correlated with solar deployment density).

2 Previous Work

There are surprisingly few previously published studies that use census data to make demographic predictions considering the availability of census data; the few that were found are described here. The University of California, Irvine's (UCI) 1996 census income dataset has been used to predict whether income levels are below or above $50,000 per year with logistic regression and random forests, achieving classification accuracy of around 92% [3]. A previous CS229 project accomplished the same task with the same dataset, also using random forests and logistic regression [4]. These studies suggest that logistic regression and random forests may be suitable algorithms for our work, but these examples differ in that they are classification tasks rather than regression predictions.

Neural networks have also been used in conjunction with census data. Wang et al. predicted population counts using a standard feed-forward neural network with 4-6 hidden layers [5], however the main focus of this work was to compare the neural network performance with that of linear regression. Census and weather data have also been used to augment crime data to forecast crime in Chicago and Portland as a multi-classification problem [6]. The authors accomplished this with several neural network architectures: feed-forward, convolutional, recurrent, and recurrent convolutional, with the best result being about 75% accuracy. A third, deep-learning study predicted age and gender from cell-phone metadata using a convolutional network [7]. Although these works utilized NN's for a different task than ours, these studies demonstrate the power of neural networks in conjunction with census data.

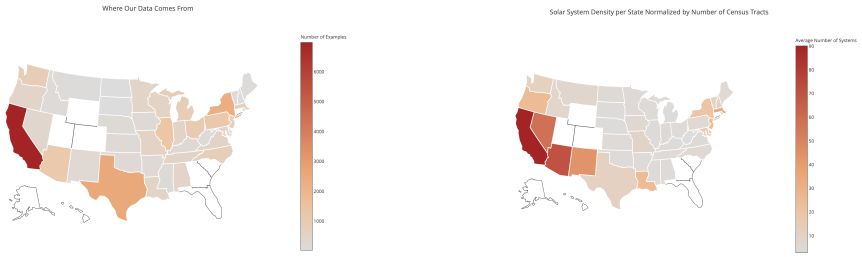
To our knowledge, the soon-to-be-published paper [2] from which the dataset is obtained is the first to utilize machine learning for large-scale surveying of solar systems. Near-term solar power forecasting using machine learning is more commonplace [8], but this project and the aforementioned paper are among the first to study system installation counts, which are more correlated with long-term solar power forecasting and market sizing.

3 Dataset

The dataset used for this project contains 155 different demographic statistics of each respective census tract, along with the number of solar systems in each tract [1]. The dataset contains just under 36,000 valid examples. The labels were determined from a previous project originating from ES's research group, which used a semi-supervised convolutional neural network to count solar panel systems from satellite images (not yet published) [2]. The data is split into 80 percent train, 10 percent dev, and 10 percent test sets. Pre-processing was done to remove invalid data from the dataset, such as null, infinite, or NaN entries. Furthermore, string and categorical columns were excluded. All features were normalized to zero mean and unit variance before being used in PCA, SVM, or NN.

The label "number of solar systems" refers to the count of solar installations, not individual panels. This means that a rooftop solar panel array, such as on SEQ buildings, and a large solar farm, such as the Ivanpah Solar Power Facility in western California, are each counted as a single solar system. This labeling system is used to avoid skewing the data with large solar farms. In addition, this labeling system is more useful for mitigating the uncertainty in the number of solar panels, the majority of which is due to rooftop solar arrays rather than large solar farms.

Figure 1(a) shows the density (examples per state) of examples across the United States, while Figure 1(b) shows the density of solar systems (number of systems per state divided by the number of census tracts in the state).



(a) Example Density

(b) Solar System Density

Figure 1: Where the Data Comes From

4 Methods

Based on the limited amount of previous work on census data-related predictions, we decided to implement two different machine-learning algorithms: support vector regression (SVR), and a fully connected neural network (NN) to predict solar system deployment density. The NN was chosen because of the aforementioned preceding work for census-prediction tasks, while the SVR was chosen as an candidate for improvement on the logistic regression used in previous works. These two algorithms were also chosen because both can learn highly-nonlinear trend-lines, and after carrying out PCA, it was immediately clear that the data is nonlinear. The following sections detail the PCA, SVR, and NN algorithms.

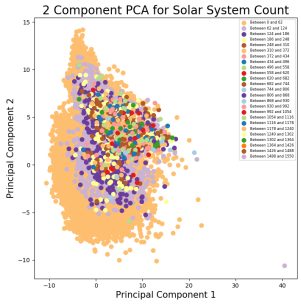


Figure 2: 2-Component PCA

4.1 Principal Component Analysis

In order to visualize the data, as well as gain insight into which features are the most influential in modeling, two-component principal component analysis (PCA) was implemented. The first k principal components of a dataset correspond to the first k eigenvectors of the matrix $\frac{1}{m}\sum_{i=1}^{m} x^{(i)}x^{(i)T}$. This is equivalent to maximizing $u^T\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}x^{(i)T}\right)u$ subject to $||u||_2 = 1$. Figure 2 shows that a model capable of representing radial contours may be the most effective. As such, it is unsurprising the support vector regression using an RBF kernel is reasonably successful. Table 1 shows a breakdown of some of the features with highest variance maintained in the first two components.

Table 1: Features Contributing Top Variance to Each Principal Component

Feature Name	Var. retained: PC-1	Feature Name	Var. retained: PC-2
High School Graduation Rate	-0.1618	% 2016 Rep. Vote	-0.1399
Rate of Families Below Poverty	-0.1586	Frost Days	-0.1389
Public Health Insurance Rate	-0.1496	Relative Humidity	-0.1356
Master's Education	0.1887	Overall Electricity Price	0.1698
Per-Capita Income	0.1887	Rebate	0.1717
Median Household Income	0.1905	Residential Incentive	0.1773

4.2 Neural Network

The standard feed-forward neural network takes in 155 demographic/geographic features (x_1, x_2, ..., x_{155}) and then predicts the number of solar panel systems for the given input features ($\hat{y}$). A NN consists of layers of neurons which first linearly combines all inputs x_i into the neuron as $z_j^{[l]} = \sum_i^{155} w_i^{[l]} x_i + b^{[l]}$ where $z_j^{[l]}$ is the j^{th} neuron in layer l; w_i is the weight for the i^{th} feature; and $b^{[l]}$ is the bias in layer l. The weights w_i and biases b for each layer are fitted by the NN to the data. Note that all of the w_i's of a layer are represented as a single matrix $W^{[l]}$, where the individual w_i^T's make up the rows of W. The logit $z_j^{[l]}$ is then passed through a rectified linear unit (ReLU) activation function $a_j^{[l]} = \max\{0, z_j^{[l]}\}$ which allows the model to learn non-linear trends. This result is then passed to the next layer of neurons, and similarly to the next, until the final layer outputs a prediction for the number of solar systems $\hat{y}$.

In the backpropagation step, the NN optimizes the weights W and biases b of each layer by minimizing them against the loss function $\mathcal{L}(y, \hat{y})$, where y is the ground truth label, using gradient descent

$$W^{[l]} := W^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[l]}} \qquad b^{[l]} := b^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[l]}}. \tag{1}$$

where α is the learning rate. The derivatives are calculated by using the chain rule of calculus starting from the loss function, working backwards through the layers. Further details can be found in [11].

The neural network designed for this project was coded using Keras [12] and TensorFlow [13]. The final model consists of 2 hidden layers with 512 and 128 neurons respectively, ReLU activation functions for each hidden layer, a linear activation function for the output node, Adam as the optimizer with default hyper-parameters [9], a learning rate of 0.0003, batch size of 512, and 200 training epochs. The mean-squared-error $MSE = \mathcal{L}(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$ is used as the loss function, where m is the number of examples in the dataset.We use dropout [10] to regularize and reduce over fitting the model with keep probabilities of 75% for both hidden layers. We use the coefficient of determination R^2 and mean absolute error (MAE) of the validation set as metrics to evaluate the model,

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{m} (y_i - \mu_y)^2} \qquad MAE = \frac{1}{m} \sum_{i=1}^{m} |y_i - \hat{y}_i| \tag{2}$$

where μ_y is the mean of the labels y.

The hyper-parameters of the final NN were chosen after two rounds of hyper-parameter searches. In the first search, we ran 25 models with randomly selected hyper-parameters and then compared the R^2 and MAE of each model. A summary of the varied hyper-parameters are shown in the table below.

Table 2: 1st Hyper-Parameter Search

Hyper-Parameter	Range
Learning rate (α)	10^{-4} to 10^{-2} (log scale)
Batch size	16, 32, 64, 128, 256, 512
# Hidden layers	2 to 5
# Neurons per layer	8 to 512
Dropout prob.	0.0 to 0.6

Tuning based on the hyper-parameter ranges in table 2, the best performing model achieved an R^2 of 0.77 and a MAE of 10.6 with a learning rate of 0.003, a batch size of 512, 2 hidden layers, 493 and 216 neurons in the hidden layers, and a dropout probability of 0.22 and 0.24 for the hidden layers.

To arrive at the chosen hyper-parameters for our final model, we performed a second optimization by varying individual hyper-parameters and comparing the metrics one model at a time. The performance of the final model is shown in Figure 3.

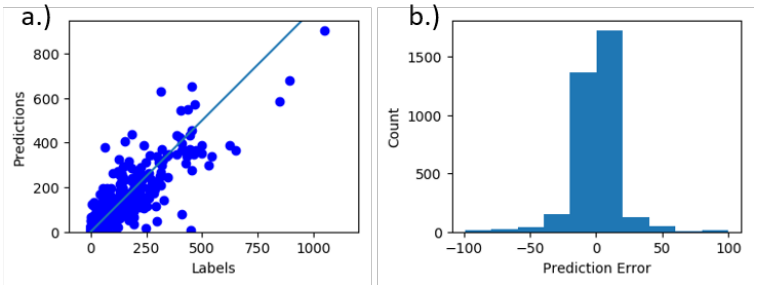


Figure 3: Neural Network Performance: **a)** Test predictions vs test labels. **b)** Histogram of the error (difference) between the model predictions and the true test labels.

4.3 Support Vector Regression

In classification tasks, Support Vector Machines (SVM) work by maximizing the margin between the classes. Support Vector Regression (SVR) works similarly, but instead attempts to find a best-fit curve. SVR's can also utilize kernels to fit non-linear curves by mapping features to a high-dimensional space. Below are the kernel functions used in the SVR models, where x_i, x_j are generic feature vectors:

$$\text{Linear kernel} : K(x_i, x_j) = x_i^T x_j \tag{3}$$

$$\text{Polynomial kernel} : K(x_i, x_j) = (1 + x_i^T x_j)^d \tag{4}$$

$$\text{Gaussian (RBF) kernel} : K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{5}$$

The kernelized-SVR model was explored using these three kernels. The penalty parameter was first set at 300, kernel cache size at 200, and the kernel coefficient set at $1/(n * \sigma_x)$. Model selection and parameter tuning was performed with SciKitLearn's GridSearchCV module [14]. Like the NN, MSE was used as the loss function and MAE and R^2 coefficient were used as metrics.

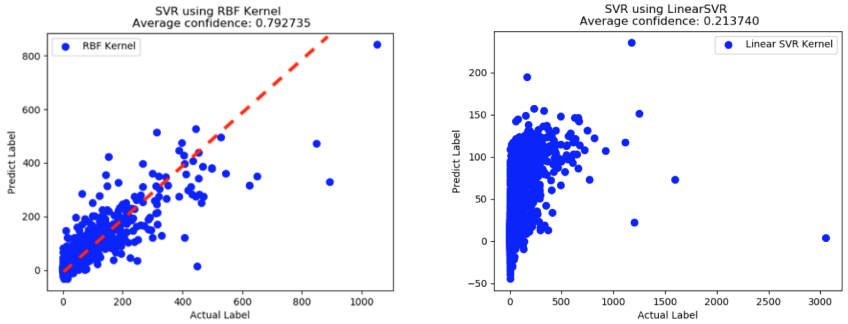


Figure 4: **Left:** SVR-RBF Results **Right:** LinearSVR Results

After running all three kernels with the full dataset and GridSearchCV parameter tuning, LinearSVR results in poor model performance, and the polynomial kernel did not converge after many hours of running. Among all three kernels, the RBF kernel resulted in the best accuracy and a reasonable model fitting time, which, as previously mentioned, is was expected based on the PCA results. The results of the final LinearSVR and SVR-RBF models are shown in Figure 4.

4.4 Model Performance

The SVM performed best with a radial basis function kernel, while the NN achieved similar performance after two rounds of hyper-parameter searching. These results are summarized in Table 3.

Table 3: Model Performance Summary

Model	Train MAE	Dev MAE	Test MAE	Train R^2	Dev R^2	Test R^2
SVR w/ Linear Kernel	29.1	29.9	29.3	0.26	0.16	0.22
SVR w/ RBF Kernel	4.2	17.4	18.1	0.93	0.79	0.78
Neural Network	7.3	10.5	18.5	0.95	0.79	0.71

Both the SVR with an RBF kernel and the NN achieve similar MAE results. We attribute this to the variance in the data. Our dataset is inherently noisy, since there are many census tracts with very few solar panel installations and a select few with extremely high numbers of solar panel systems, such as northern California. Further work may include analyzing the distribution of the data down to the census-tract level and modifying the models based on any findings.

4.5 Feature Influence

For this application, we computed numeric gradients using the NN to determine the influence of each feature on solar system deployment, identifying which features correlate most directly with the number of solar systems. To do this, numeric gradients of the label with respect to each input feature were calculated by changing a single feature by a small ϵ and completing a forward pass to make a prediction. Features that have a large positive numerical gradient have a strong positive correlation with the number of solar systems, while features with a large negative gradient have a strong negative correlation. Table 4 contains a few selected features of interest.

Table 4: Feature Influence

Feature Name	Relative Pos. Correlation	Feature Name	Relative Neg. Correlation
% of Dem. Voters (2012)	0.85	Population Density	-1.00
Median Power Usage	0.84	% of Rep. Voters (2012)	-0.64
Median House Value	0.81	% of Rep. Voters (2016)	-0.63
% of Dem. Voters (2016)	0.67	Population Age 5-9	-0.57
Solar Irradiance (KWh/yr)	0.66	# Frost Days	-0.29
Sales Tax Rate	0.46	Poverty Level	-0.24

In Table 4, the relative positive/negative correlation is the fraction of the numerical gradient of each respective feature normalized to the absolute value of the maximum numerical gradient of any feature, which was population density.

The results both validate the model and offer some interesting insights into the data. For example, a strong positive correlation with yearly solar irradiance is expected. A strong negative correlation with number of frost days is also expected, since cold areas do not get as much sunlight as warmer areas, thus these census tracts are less likely to have high concentrations of solar panels. The strongest negative correlation is with respect to population density, which is also justifiable because areas with a lot of large houses, or low population density, are bound to have more solar panels than dense urban areas where residents do not have the same control over their own roofs.

The most striking, but not surprising, results are the strong positive correlation with % Democratic voters and the strong negative correlation with respect to % Republican voters. In both 2012, and 2016, the results rank near the top of each respective category. This distinction between the two political parties is stark, though not surprising given each party's respective stance on climate change.

5 Conclusion and Future Work

Both an SVM and a standard feed-forward NN can predict the number of solar systems reasonably well from US census data. Both models have similar performance, achieving an mean-absolute-error of around 11 solar panels systems and an R^2 value of the best fit line between the predictions and the true labels of around 0.79. Principle component analysis determined which features contributed the most variance to the labels, and NN numeric gradients of the prediction with respect to each feature quantified each feature's influence on the model. Strongly correlated features include voting tendencies, solar irradiance, median housing values, and population density.

Opportunities for future work include separating residential and commercial installations, which could even more accurately help model un-tracked energy contributions to the grid. Furthermore, it would be worth exploring how much power is generated in a given census tract, which would be a function of not only number of solar systems, but solar panel area and weather conditions. Similarly, it could be useful for companies selling solar systems to be able to more accurately predict the monetary gains from energy generated over time in a given neighborhood as a selling point or revenue predictor.

Contributions

Eddie

Neural Network code and write-up, poster and final report skeleton and major writing

Jeremy

Support Vector Regression code and write-up

Brett

Data pre-processing, PCA code and write-up, poster and write-up final editing

Code

Github Repo: `https://github.com/brettski15/cs229_solar`
Branch: `master`
See included `README.md` for running instructions

References

[1] American Community Survey. 2015. 2015 ACS 1-year estimates [data file]. Retrieved from http://factfinder.census.gov

[2] Yu, Jiafan, et al. "DeepSolar: A Machine Learning Framework to Efficiently Construct Solar Deployment Database in the United States." Joule (2019), accepted.

[3] Sheremata, Jeff. "Machine Learning Income Prediction Using Census Data." Medium.com, Medium, 11 Jan. 2017.

[4] Voisin, M. Prediction of Earnings Based on Demographic and Employment Data.CS 229 Report, 2016.

[5] Wang, Jun, et al. "Advances in Neural Networks - ISNN 2006." Lecture Notes in Computer Science, May 2006, doi:10.1007/11759966.

[6] Stec, Alexander, and Diego Klabjan. "Forecasting Crime with Deep Learning." ArXiv [Stat.ML], 5 June 2018.

[7] Felbo, Bjarke, et al. "Using Deep Learning to Predict Demographics from Mobile Phone Metadata." ICLR Workshop Track, 13 Feb. 2016.

[8] Isaksson, Emil, and Mikael Karpe Conde. "Solar Power Forecasting with Machine Learning Techniques." KTH Royal Institute of Technology, Royal Institute of Technology, 2017.

[9] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[10] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[11] Katanforoosh, Kian, and Andrew Ng. "Deep Learning." CS229: Machine Learning, 2018.

[12] Chollet, François. "Keras." (2015).

[13] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.

[14] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12.Oct (2011): 2825-2830.

[15] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing in science & engineering 9.3 (2007): 90-95.

[16] Walt, Stéfan van der, S. Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." Computing in Science & Engineering 13.2 (2011): 22-30.

[17] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for High Performance and Scientific Computing (2011): 1-9.

[18] Sievert, Carson, et al. "plotly: Create Interactive Web Graphics via 'plotly. js'." R package version 3.0 (2016).