Geography Embeddings for Predicting Prices of Self-Storage Units

Bassem Akoush[1], Hashem Elezabi [2]
{bakoush, hashem}@stanford.edu

[1] Department of Mechanical Engineering, Stanford University, Stanford, CA 94305
[2] Department of Computer Science, Stanford University, Stanford, CA 94305

Project Mentor: Scott Fleming.

1 Introduction

In this project, we consider the problem of predicting prices of self-storage units across the US by incorporating *geography embeddings* that capture information about a location. Developing a model to effectively predict these prices can be helpful to businesses deciding how to price their self-storage services at different locations. To this end, one can train models based on *unit features* like square footage or elevator accessibility, which we do in this project. Beyond that, this project explores how to improve these models using geography embeddings – vector representations that can capture location-based features not contained in unit features. Independent of our specific price prediction task, the problem of finding effective geography embeddings is an active research area [1–5]. This project mainly explores applying satellite image-based embeddings called MOSAIKS [1], as well as fusing MOSAIKS features with census-based features to capture more information about the surrounding neighborhood. Our project involves significant work on two stages: featurization and learning. For featurization, the input consists of various data modalities (satellite images, census data, etc.) and the desired output is an effective vector representation (embedding). For learning, the input is the vector representations we extracted, and the output is the self-storage unit price prediction.

2 Related Work

We are not aware of previous work on predicting prices of self-storage units using machine learning. This problem was recently formulated by our mentor Scott Fleming, and the dataset was recently collected. Thus, this section focuses on related approaches to geography embeddings, a key part of our project. In building our own embeddings, we are mainly inspired by Rolf et. al. [1], who proposed a method to create unsupervised vector representations of satellite imagery. They showed that their representation, termed MOSAIKS, could effectively generalize to different tasks (e.g., predicting forest cover or population density), meaning that anyone can take these "embeddings" and apply them to their own problem by training them with their own task-specific labels. This enabled us to run various learning algorithms on the MOSAIKS vector embeddings, instead of needing to train expensive convolutional neural networks (CNNs) on the satellite images themselves, which is the state-of-the-art tool for ML on satellite imagery.

Other researchers have developed various embeddings by incorporating other data modalities like street-level imagery [2], Point-of-Interest data [5], object detection [6], and a combination of census data and online reviews [4]. Jean et. al. [7] generated geography embeddings for poverty prediction by training a convolutional neural network (CNN) on satellite images, which became a well-known result that inspired later research, including MOSAIKS.

3 Dataset and Features

In this study, multiple datasets were developed by the authors and the mentor for various tasks. The initial dataset, which we call Dataset I, contains *unit features* like square footage, 24-hour access, whether it's climate-controlled, etc. This dataset was constructed by running web scraping algorithms on the top self-storage providers: Storage Sense [8], Extra Space [9] and Primary Storage Group [10]. It has 33,085 units from 2,271 facilities with an overall number of features of 36. A facility is an entire self-storage establishment, while a unit is a single storage room. Each data point represents a unit; different units in a facility share the same location but have different unit features. After some experimentation, we narrowed the unit features down to the 19 most useful features for training our models.

Dataset II is a combination of the unit features from Dataset I with data from the US census. The census data is used to provide information about the surrounding area and neighborhood, which we hypothesized might improve the learned models. The census information is obtained from the American Community Survey 5-Year Data, released in 2019. It contains broad social, economic, housing, and demographic information. We extracted almost 1080 features from the census data, and added these columns to every row (data point) in Dataset I.

The third major set of features is satellite image-based features. We extracted these by applying the MOSAIKS method to satellite images at all 2,271 self-storage facility locations, represented by latitude-longitude pairs found in the original Dataset I. We obtained the images using the Google Maps Static API [11] with zoom level 16. Each image covers an area of 1km x 1km with a size of 620 x 620 pixels and three RGB channels. These images are first resized to 256 x 256 pixels before featurization. The number of MOSAIKS features is a parameter of the algorithm (the value K, explained under Methods). We experimented with different values and found that

1024 features were enough to give good performance. The main analysis in Rolf et. al. used 8192 features, but their experiments showed that as little as 100 features recovered a substantial amount of predictive power across the seven tasks they tried.

The data was split into an "in-distribution" set containing facilities from the first two providers, Storage Sense and Extra Space, and an "out-of-distribution" set containing facilities from the third provider, Primary Storage Group. The in-distribution set was split further into a Train set (22101 units across 1684 facilities) and a Test-ID set (5555 units across 427 facilities), while the out-of-distribution set served as a Test-OOD set (5429 units from 160 facilities). The Test-OOD set was a great way to test how much the patterns we learned were generalizable to unseen facilities. This way, we stress-tested our models not just on unseen data from the same "distribution" of providers, but on unseen data from unseen providers.

We preprocessed the dataset to filter out outliers and remove incorrectly labeled data. We also tried running PCA on the features, though that didn't produce meaningful improvement. Finally, we normalized the dataset to have a range between 0 and 1. We also tested scaling the dataset to have zero mean and one standard deviation, though that didn't produce meaningfully different results from the first normalization method.

4 Methods

4.1 MOSAIKS Features Extractor

MOSAIKS stands for "Multi-task Observation using Satellite Imagery and Kitchen Sinks". Given a set of N satellite images I_l, where l denotes the location of the image, the featurization process works by choosing a random sample of K small image patches across the N images and convolving these patches over each image. For each image I_l and each patch P_k, the output of this convolution is fed into a nonlinear activation function, $ReLU$ in this case. For this single image I_l, this generates a nonlinear *activation map* for every patch k from 1 to K. Averaging the kth activation map over the image pixels gives a single value that represents the strength of feature k in the image. This k^{th} feature becomes the k^{th} entry in the K-dimensional feature vector that is the output of this procedure. Figure 1 shows a visual depiction; the images shown are satellite images at locations of self-storage facilities from our dataset. Like the original MOSAIKS implementation, we used patches of size 3x3x3, with the last dimension representing 3 color channels.

One way to interpret MOSAIKS is that it's a forward pass of a shallow 2-layer CNN with an 8,192-neuron wide hidden layer and randomly initialized weights drawn from random patches in the image dataset. Because we don't need to do backpropagation, the approach is much more efficient than training the CNN weights. The theoretical explanation for why the MOSAIKS approach works lies in the machine learning concept of *random kitchen sinks* [12], which is "a method for feature generation that enables the linear approximation of arbitrary well-behaved functions" [1]. The MOSAIKS authors adapt the random kitchen sinks framework to satellite images by using random convolutional features (as described above) and making the simplifying assumption that "most information contained within satellite imagery is represented in local image structure."

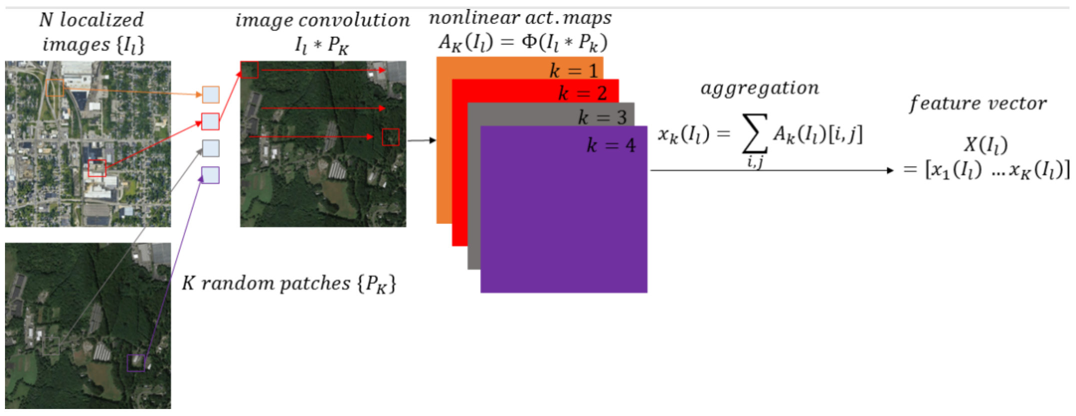
N localized images $\{I_l\}$
image convolution $I_l * P_K$
nonlinear act. maps $A_K(I_l) = \Phi(I_l * P_k)$
$k = 1$
$k = 2$
$k = 3$
$k = 4$
aggregation
$x_k(I_l) = \sum_{i,j} A_k(I_l)[i,j]$
feature vector
$X(I_l) = [x_1(I_l) \ldots x_K(I_l)]$
K random patches $\{P_K\}$

Figure 1: Flow diagram depicting MOSAIKS featurization.

4.2 Learning Methods

4.2.1 *Linear Regression* **(Baseline)**

We used linear regression as our baseline model. In linear regression, the model tries to fit the data by minimizing the least squares cost function. The hypothesis function can be related to the design matrix X in linear style as shown in Eqn. 4.1. Gradient descent is typically used to determine the weights of the model, with the update rule shown below. We add a regularization term to the cost function to avoid overfitting to the training data.

$$h_\theta(X) = X\theta + b \tag{4.1}$$

$$\theta := \theta + (\alpha/N)(\sum_{i=1}^{n}\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x^{(i)} - \lambda\theta) \tag{4.2}$$

4.2.2 K-Nearest Neighbor

The k-nearest-neighbor is a nonlinear model that estimates the hypothesis function by the mean value of the ground truth labels over the k nearest neighbors. Let $p_1, \cdots, p_k$ be the indices of the k closest points to an input point x, where closeness is measured based on Euclidean distance. Then the prediction is as follows: $h(x) = \frac{1}{k}\sum_{i=1}^{k} y^{(p_i)}$.

4.2.3 Neural Network

Neural networks (NNs) are widely used due to their ability to learn complex nonlinear relationships. The nonlinearity is achieved by the per-layer activation functions; typical examples are ReLU, tanh, sigmoid, etc. Neural networks do not make any assumptions on the input and output distribution like other machine learning methods. The governing equation for the neural network is:

$$Z^{[k]} = g\left(X\left(W^{[k]}\right)^T + b^{[k]}\right) \tag{4.3}$$

where k is an enumerator for the hidden layers. $Z^{[k]}$ becomes the input to the next layer, and so on. This completes the forward pass of the network, which produces a prediction. This prediction gets compared with the true label via a cost function. The gradients of this cost function w.r.t all the NN weights are computed via backpropagation and used to update the network weights via gradient descent, as we learned in class.

4.2.4 Gradient-Boosted Decision Trees

Decision trees are a class of supervised learning methods that "learn" partitions of the input feature space and use these partitions to make predictions about a given input. They are used in both classification and regression. Boosting is a machine learning technique that combines several *weak learners* – simple models that are only slightly correlated with the true labels – into a single *strong learner* – a model that can achieve arbitrarily good correlation with the true labels. A gradient-boosted decision tree (GBDT) model builds a strong predictor out of several weaker predictors, following the boosting technique, but it does so by iteratively improving the model using gradient updates based on an arbitrary differentiable loss function (with the mean squared error (MSE) loss being the canonical example). XGBoost [13] and LightGBM [14] are two well-known efficient implementations of GBDTs. We used them extensively in this project, and they performed well in our task. We describe results in more detail in the next section.

5 Results and Discussion

5.1 Correlation between Unit Features

We started by inspecting the dataset and constructing a correlation matrix to understanding correlations both among features as well as between features and the target variable (price). This shows possible redundancies, suggests which features are most or least useful, and might expose problems with the dataset. Figure 2 shows the correlation matrix, which also shows more examples of unit features.

5.2 NN Hyperparameters Tuning

Tuning the neural network hyperparameters is an essential step toward good prediction performance. The network architecture ranged from 5 to 15 layers while the number of neurons ranged from 25 to 500. The leaning rate was studied on the range between 10^{-2} to 10^{-4}. The effect of changing the activation function was checked by using ReLU and tanh functions. Lastly, two optimizers, Adam and stochastic gradient descent (SGD), were considered in the tuning process. The best prediction was achieved with the following configuration: #layers = 10, # neurons = $100:200$ depending on the dataset, Adam optimizer, learning rate = 10^{-4}, and ReLU. It had the highest correlation coefficient and minimum RMSE.

5.3 Experiments

We started our experiments by training various models on Dataset I, which contains only the unit features. We wanted to find which learning algorithm performs best and how well we can do using only the unit features. The two main metrics we considered are the root-mean-square error (RMSE) and the R^2 correlation coefficient, both defined below. Intuitively, the R^2 score measures the degree of correlation between the predictions of the model and the true labels. It's a way to measure how much variation in the true labels was able to be captured by the model. As previously explained, the Test-OOD out-of-distribution sample is an excellent measure of the model's performance and generalization ability since it contains data from a different self-storage company than those used in the Train and Test-IN samples.

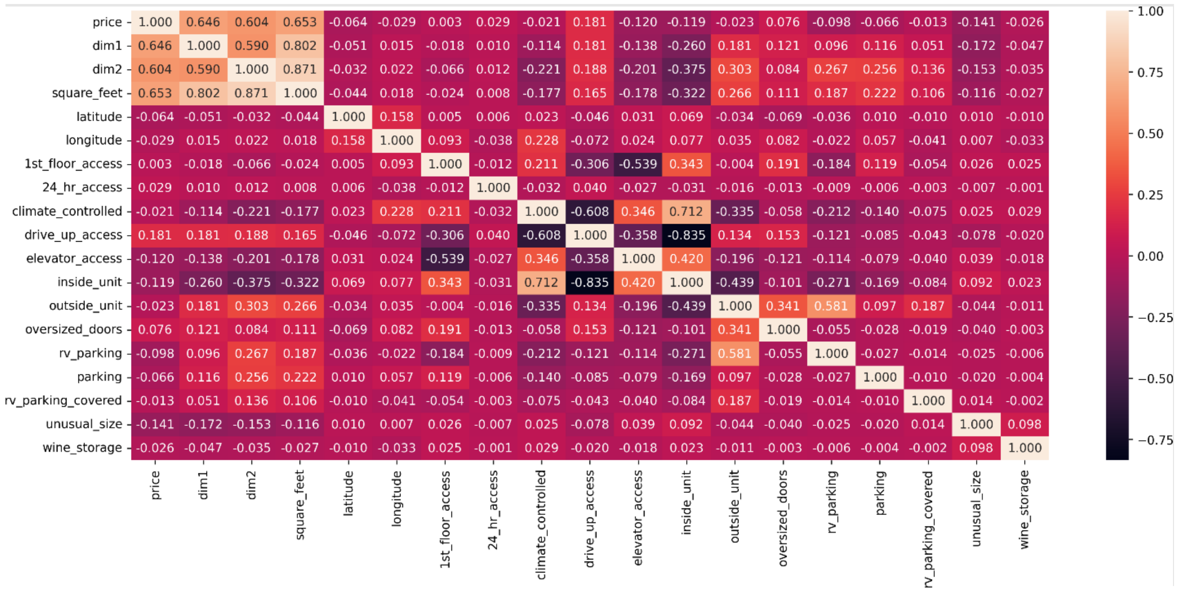

Figure 2: Correlation matrix on unit features.

Table 1 shows the results of these runs. It shows that LGBM achieves the best performance on the OOD sample ($R^2 \sim 0.723$), outperforming the neural network. This is consistent with previous studies showing state-of-the-art performance using LGBM on tabular data like ours [15]. XGBoost achieves similar performance.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\bar{y} - y_i)^2}{N}}, \qquad R^2 = \frac{\left[\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})\right]^2}{\left[\sum_{i=1}^{N}(y_i - \bar{y})^2\right]\left[\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2\right]} \tag{5.1}$$

The remainder of this project focused on improving the performance of our models by augmenting the unit features from Dataset I with more features about the location, thus creating richer geography embeddings that can capture more information and have more predictive power. Specifically, we started by adding census features, creating Dataset II as detailed under Dataset and Features. This yielded an improvement of around ~5% and ~7% in the R^2 and RMSE score of the LGBM model, respectively, achieving a score of around $R^2 = 0.76$ on the Test-OOD sample.

Next, we worked on extracting and incorporating MOSAIKS features in our datasets. We started by training on just the MOSAIKS features to see how much predictive power a model based only on satellite image features can have in our task. One challenge here is that we needed a price label for the entire facility, since different units (with different unit features) in the same facility shared the same latitude-longitude location, and thus the same corresponding satellite image features. Because we didn't use unit features in this experiment, we had to compute a representative price for the facility as the ground truth label for the corresponding MOSAIKS feature vector. We chose to compute a noisy estimate: the average of the unit prices, each normalized by square footage. Normalizing by square footage eliminated the variation due to different unit sizes. There were still other unit-specific factors that we ignored, so this estimate was quite noisy. Training a LightGBM model on this dataset showed poor performance, with the model achieving an R^2 score around 0.2, i.e. capturing around 20% of the variation in the true labels. This showed that, as we suspected, MOSAIKS on its own was not enough to perform well in our task. The next step was to fuse MOSAIKS with other unit and census features and see to what extent, if any, this improves performance.

We fused MOSAIKS features with each of Dataset I and Dataset II. In all our MOSAIKS experiments, to avoid information from our test samples "leaking" into our training procedures, we extracted the MOSAIKS features using random patches from only the training set, and we then applied the resulting featurizer network to the satellite images from the test samples. Adding MOSAIKS features to Dataset I improved the R^2 score by around 1% and decreased the RMSE by around 5% compared to just unit features. Adding MOSAIKS features to Dataset II improved the R^2 score by around 3% and also decreased the RMSE by around 5% compared to just unit features. While they yielded an improvement, satellite image-based features (MOSAIKS) were not as helpful to our price prediction task as census features, which more directly capture socioeconomic information. We had hoped that MOSAIKS would contain features roughly correlating with socioeconomic factors relevant to our price prediction task, but it seems they are better suited for information more directly observable from satellite imagery, such as forest cover or population density.

Table 1: Regression statistical error metrics.

Model	Training			Testing-IN		Testing-OOD	
	RMSE	R^2_{tr}	$\overline{R^2_{val}}$	RMSE	R^2	RMSE	R^2
Dataset I (Unit only)							
Linear Reg.	80.21136	0.63392	0.61761	87.24828	0.58482	108.85204	0.57953
KNN	65.13425	0.75861	0.71391	71.16024	0.69998	96.36328	0.68104
XGBOOST	57.17571	0.81399	0.75469	67.86799	0.74878	88.67562	0.72096
LGBM	56.81966	0.81630	**0.78564**	66.03489	**0.76217**	88.38667	**0.72277**
NN	63.02124	0.77402	--	72.42443	0.71392	90.93299	0.70657
Dataset II (Unit + Census)							
XGBOOST	47.84797	0.86973	0.76600	66.74784	0.75700	84.38663	0.74730
LGBM	47.63630	0.87088	**0.77242**	66.41609	**0.75941**	82.28039	**0.75976**
NN	44.47064	0.88748	--	78.82553	0.66111	102.88404	0.62437
Dataset I + MOSAIKS (Unit + MOSAIKS)							
Linear Reg	72.74770	0.69888	0.65973	81.69317	0.63601	103.35437	0.62093
XGBOOST	25.33794	0.83720	0.74354	70.92833	0.73285	87.03605	**0.72969**
LGBM	37.41008	0.83971	**0.75340**	67.39999	**0.74214**	83.66533	0.72779
NN	54.21112	0.83278	--	83.58285	0.61897	89.36030	0.71663
Dataset II + MOSAIKS (Unit + Census + MOSAIKS)							
XGBOOST	48.27899	0.86738	0.76591	67.15500	0.75403	84.21366	0.74833
LGBM	47.36459	0.87235	**0.77140**	66.75418	**0.75696**	83.98908	**0.74967**

Below are three plots that show goodness of fit of the LGBM model with 20 leaves and 50 estimators (these are hyperparameters of LGBM), run on Dataset II with MOSAIKS features added. We found these hyperparameters by performing a grid search over the GBDT parameters and using cross-validation with 10 folds on the training dataset.

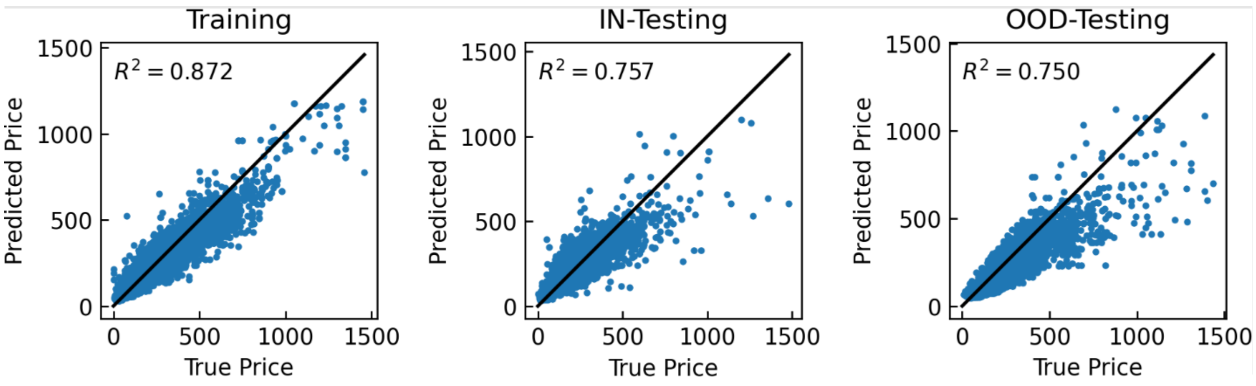

Figure 3: Goodness of fit on training and testing datasets.

6 Conclusion and Future Work

The concept of generating extra features about the location of the storage facility showed great potential in enhancing the price prediction. The error metrics were improved by taking into consideration the census that gives additional information about the neighborhood. Furthermore, the combination of the unit features and the geography embeddings from MOSAIKS led to boosting the price prediction. There was also an even more significant gain from picking the right learning algorithm and tuning it. Overall, from adding location-based features and using better learning algorithms, we improved price prediction performance by ~31% from our baseline of $R^2 = 0.58$ (linear regression on Dataset I) to a best score of $R^2 = 0.76$. Still, the MOSAIKS embedding did not improve model performance as much as we expected. We suspect the reason is that MOSAIKS features are better suited for prediction tasks to which satellite images are more directly applicable, such as forest cover or population density. For future work, we would consider 1) exploring feature extraction using street-level imagery, 2) extracting features about nearby competitors (e.g. how many other self-storage facilities within a 1-mile radius), and 3) exploring different fusion strategies that might combine different data sources in more effective ways.

7 Contributions

Bassem implemented the learning algorithms and ran most of the experiments in this project. He also implemented the code to download satellite images from the Google Maps Static API and worked on the MOSAIKS features extraction. Hashem studied MOSAIKS embeddings and set up the software infrastructure for extracting MOSAIKS features from our satellite images. He also ran MOSAIKS-specific experiments to help understand the potential of MOSAIKS on our price prediction task.

8 References

[1] E. Rolf, J. Proctor, T. Carleton, I. Bolliger, V. Shankar, M. Ishihara, B. Recht, S. Hsiang, A generalizable and accessible approach to machine learning with global satellite imagery, Nature Communications. 12 (2021) 4392. https://doi.org/10.1038/s41467-021-24638-z.

[2] J. Lee, D. Grosz, B. Uzkent, S. Zeng, M. Burke, D. Lobell, S. Ermon, Predicting Livelihood Indicators from Community-Generated Street-Level Imagery, Association for the Advancement of Artificial Intelligence. (2021). https://doi.org/https://doi.org/10.48550/arXiv.2006.08661.

[3] S. Ermon, Stefano Ermon: Satellite images can pinpoint poverty better than surveys, (2017).

[4] M. Salari, M.D. Kramer, M. Reyna, H.A. Taylor, G.D. Cliffor, Combining Crowd-Sourcing, Census Data, and Public Review Forums for Real-Time, High-Resolution Food Desert Estimation, SSRN Electronic Journal. (2021). https://doi.org/10.2139/ssrn.3823672.

[5] Y. Xi, T. Li, H. Wang, Y. Li, S. Tarkoma, P. Hui, Beyond the First Law of Geography: Learning Representations of Satellite Imagery by Leveraging Point-of-Interests, in: Proceedings of the ACM Web Conference 2022, ACM, New York, NY, USA, 2022: pp. 3308–3316. https://doi.org/10.1145/3485447.3512149.

[6] K. Ayush, B. Uzkent, M. Burke, D. Lobell, S. Ermon, Generating interpretable poverty maps using object detection in satellite images, in: IJCAI International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, California, 2020: pp. 4410–4416. https://doi.org/10.24963/ijcai.2020/608.

[7] N. Jean, M. Burke, M. Xie, W.M. Davis, D.B. Lobell, S. Ermon, machine learning to predict poverty, 353 (2016).

[8] https://www.storagesense.com, (2022).

[9] https://www.extraspace.com/, (2022).

[10] https://www.primestoragegroup.com/, (2022).

[11] Google Developers Static Maps API, (2022).

[12] A. Rahimi, B. Recht, Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning, in: Advances in Neural Information Processing Systems 21, 2008.

[13] T. Chen, Guestrin Carlos, XGBOOST: A Scalable Tree Boosting System, in: KDD ’16, San Francisco, California, 2016.

[14] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A Highly Efficient Gradient Boosting Decision Tree, in: 31st Conference on Neural Information Processing Systems, California, 2017.

[15] R. Shwartz-Ziv, A. Armon, Tabular Data: Deep Learning is Not All You Need, ArXiv. (2021). http://arxiv.org/abs/2106.03253.