Machine Learning in Practice – Project Report
Airbnb Price Predictions

Ziad El Assal, Youssef Lahlou, Lorenzo Rega

April 2023

Introduction

In their research article Airbnb Price Prediction using Machine Learning ans Sentiment Analysis (2019) [1], Kalehbasti et al. use different models to predict Airbnb prices in New York given intrinsic data. The dataset consists in 50,000 NYC listings, with 96 features, scraped from the Airbnb website in 2018.

The variety of the models used and the room for further exploration that this task allows are some of the reasons that guided the choice of this paper.

1 Results reproduction and enhancement

The results reproduction was straightforward besides a few version conflicts, and is summarized in Table 1. We performed hyperparameter tuning using Bayesian search (for computation time reasons) on the model classes used by the authors, and pleasantly found the same optimal hyperparameters for each of the models. We also added statistical significance indicators of the metrics (MSE, MAE and R2) by computing the standard deviation of the results obtained in a 5-fold cross-validation, which was not part of the original paper (see Table 1). We also performed additional feature engineering, namely by removing more outliers than the authors did to allow for a better min-max scaling of the data. Finally, we implemented XGBoost as an additional model, which performed significantly better than the others, except for SVR.

2 Adding External Features

In this section, we explore how adding demographic and socio-economic features about New York's neighborhoods to the dataset could improve our results.

2.1 Choice of new features

The original dataset we are working with contained a lot of information about the characteristics of the listings but very few information about neighborhoods. We extracted the following features from the American Community Survey (ACS) database for 2018:

- *Population*: The estimated population 25 years old in the neighborhood.
- *Education Level*: This feature counts the number of people of 25 years or older that hold at least a bachelor degree. Indeed, we consider that a neighborhood where people are highly educated has more chances to be expensive.
- *Median Household Income*: This feature calculates in the median household income. We choose median instead of mean to avoid extreme outliers that could skew the statistic.
- *Median Age*: We assume that young neighborhoods are more dynamic and can potentially attract more tourists.

We joined the original dataset and the new one on zipcodes. Then, we preprocessed and cleaned the dataset by removing NaN values, removing rows where Population was 0, and finally changed the Education Level column from counts to proportions of the neighborhood's population.

2.2 Results

We compare the results of the new dataset on the same models used by the authors: Ridge Regression, Gradient Boosting, KMeans + Ridge Regression, Support Vector Regression, and Neural Net.

Table 1 compares different metrics for the test data on the enhanced and original datasets.

	Original Performance			New Performance		
Model name	MAE	MSE	R2 score	MAE	MSE	R2 score
Ridge Reg.	0.279 (± 0.003)	0.154(± 0.008)	0.679 (±0.008)	0.278(±0.003)	0.153(±0.008)	0.682(±0.009)
Gradient Boost.	0.280(±0.003)	0.161(±0.009)	0.664(±0.010)	0.276(±0.004)	0.158(±0.010)	0.670(±0.012)
KMeans + Ridge	0.285	0.154	0.675	0.303	0.166	0.649
Neural Network	0.266(±0.014)	0.137(±0.019)	0.715(±0.040)	0.266(±0.016)	0.137(±0.021)	0.713(±0.0435)
SVR	0.276	0.147	0.690	0.302	0.165	0.651
XGBoost	0.257(±0.003)	0.134(±0.008)	0.721(±0.009)	0.254(±0.002)	0.132(±0.008)	0.725(±0.009)

Table 1: Performances of different models on the enhanced and original data using 5-fold cross-validation (Mean ± 1 stdev)

SVR performs best for both datasets, with an R^2 of 0.65 on the new test data.

For Gradient Boosting, we notice an increase in performance for the enhanced dataset, showing that adding new features helped the model find new patterns in the data. Then, we can see that SVR stays the best model in terms of MSE, MAE and R^2 for both train and test datasets. We could attribute this better performance to the fact that SVR handles non-linearities better and is robust against outliers. Finally, we can see that the neural network performs considerably worse in the new dataset($R^2_{new} = 0.49$ versus $R^2_{orig} = 0.67$ for test data). The reason behind that may be that we tried running the neural network with the same hyperparameters as the authors rather than tuning them again.

3 Robustness Analysis

In this section, we study the robustness of the models on two different fronts. First, we add randomly generated rows, which marginal distributions are the same as in the original dataset, to the training data and evaluate the models' performance on the test set. Second, and independently from the first experiment, we train the models on the original dataset and test them on adversarial examples generated using the fast gradient sign method (FGSM).

3.1 Robustness to a compromised training set

In this section, we compromise the training set by replacing a proportion p of the rows by randomly generated rows. In order to keep the same balance in the original and generated training set, we perform Bayesian inference on each feature using the PyMC library to infer its generating distribution, thus preserving the features' marginal distributions.

3.1.1 Generating process

To generate the random rows, we divide the features into 3 categories:

- *binary features*: they are generated using a Bernoulli distribution with parameter π, equal to the sample mean of the observations.

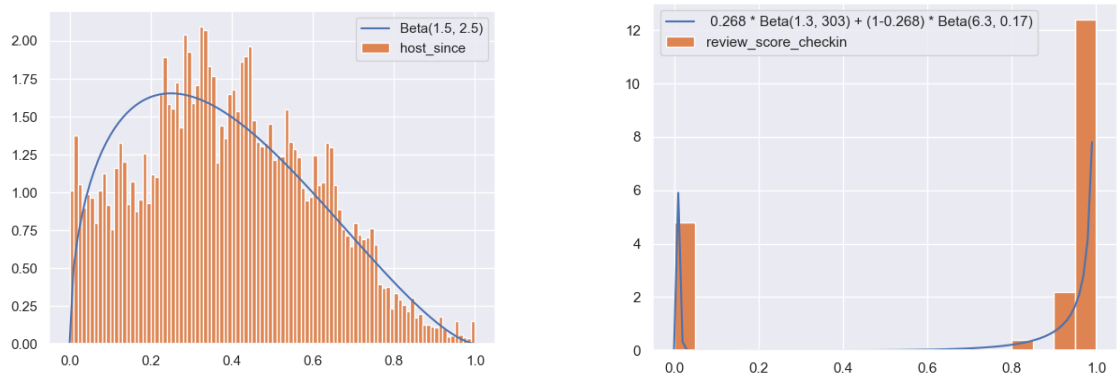

Figure 1: In orange, histogram of the observations for the features 'host_since' (left) and 'review_score_checkin' (right). In blue, probability density function of the Beta distribution with parameters $\alpha = 1.5$, $\beta = 2.5$ (left) and of a mixture model of two Beta distributions with respective parameters $(\alpha_1, \beta_1) = (1.3, 303)$ and $(\alpha_2, \beta_2) = (6.3, 0.17)$ (right), obtained while fitting a Bayesian model.

- *unimodal scalar features*: these features have one mode and are generated using a Beta distributions with parameters α and β estimated using Bayesian inference. Some examples are the number of beds, the host's total listings count or the time since the host has been on Aribnb (Fig. 1, left).
- *bimodal scalar features*: these features have two modes and are generated using a mixture of two Beta distributions. They are common among ratings, as people tend to give either high or extremely low scores (Fig. 1, right).

Note that for computational efficiency reasons, we do not sample from the posterior predictive but simply by using the means of α and β obtained in PyMC as the parameters of the beta distributions.

3.1.2 Analysis and results

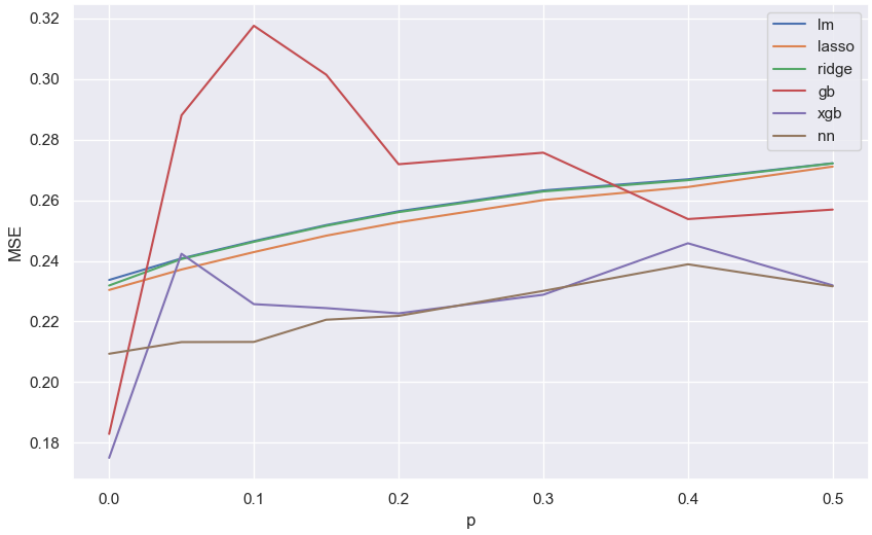

Figure 2: Mean squared error on the test data, where the models have been trained replacing a proportion p of the training data with randomly generated rows.

The results are shown on Fig. 2. The linear models perform consistently worse when the proportion p of the dataset that has been compromised increases, which is not surprising since it is a reaction to noise.

Lasso also performs better than ridge and than the standard linear model. The neural network performs consistently worse as well, which could be explained by the fact that it is also a parametric model. Gradient boosting and XGBoost display an interesting behaviour: they first perform much worse, before performing better. Here is a possible explanation: when p is small, the individual trees can overfit on specific examples from the randomly generated rows, creating specific branches for these rows. When p increases and these rows start to represent a higher proportion of the training set, they are considered as noise and not outliers, and their presence is taken into account by the regularisation methods. Finally, as XGBoost introduces stronger regularization mechanisms compared to gradient boosting, it is not surprising that it resits better to the compromised dataset.

3.2 Robustness to adversarial attacks

In this section, we study the robustness of the models to adversarial examples.

Intuitively, a model is considered robust if a small perturbation $\mathbf{x}_*$ of a data point $\mathbf{x}$ doesn't change the classification label. In a regression context, it shouldn't significantly change the output. Given the continuous setting of the regression framework, it is more of a challenge to assess the robustness of a model compared to classification, as even small perturbations of the input should yield a different output.

3.2.1 Generating process: Fast Gradient Sign Method (FGSM)

FGSM is a type of attack, introduced by Goodfellow et al.[2]. An attacker tries to deceive a neural network model by adding small, carefully crafted perturbations to the input data in order to cause misclassification or incorrect output.

The FGSM attack is a simple but effective method that involves calculating the gradient of the loss function of a model with respect to the input data, and then adding a small perturbation to the input in the direction of the gradient. The perturbation is chosen such that it maximizes the loss function increase:

$$\mathbf{x_{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)),$$

where $\mathbf{x}$ is the original input, ϵ is the size of the perturbation, θ represents the neural network weights, J is the loss function and $\mathbf{x_{adv}}$ is the adversarial example.

FGSM is typically used and is most effective to generate adversarial examples on image data and on classification tasks. Still, it provides good adversarial examples of tabular data to test our models on.

3.2.2 Analysis and results

As mentioned at the start of the section, robustness is harder to assess in a regression task as the output should always be different with an adversarial input, however close the adversarial input is. Therefore, we tackle this problem using two different approaches:

1. The first one is essentially to compare the models to each other while remaining in the regression framework, by computing the mean squared error. The error increase of a model on a particular adversarial example has little value if not compared to those of other models.

2. In the second approach, we turn the problem into a binary classification problem. The output y_{pred} is considered correctly classified if it is within a distance δ of the true value y_{true} ($|y_{pred} - y_{true}| \leq \delta$), and misclassified otherwise, which allows the use of accuracy as a metric.

The results of the first approach are displayed on Fig 3. First, the neural netwrk is unsurprisingly the least robust to adversarial FGSM examples since they have been specifically generated for it. The mean squared error of the parametric models increases consistently and smoothly with ϵ. Intuitively, this makes sense since in a linear model the MAE should be roughly proportional to the perturbation size ϵ. It is also reassuring to see that the regularized linear models are more robust than their standard cousin. Finally, the gradient boosting models show greater robustness than the others, especially for high values of ϵ. The way they are built, by combining many weak predictive models into a strong ensemble to increase robustness, is a very plausible explanation.

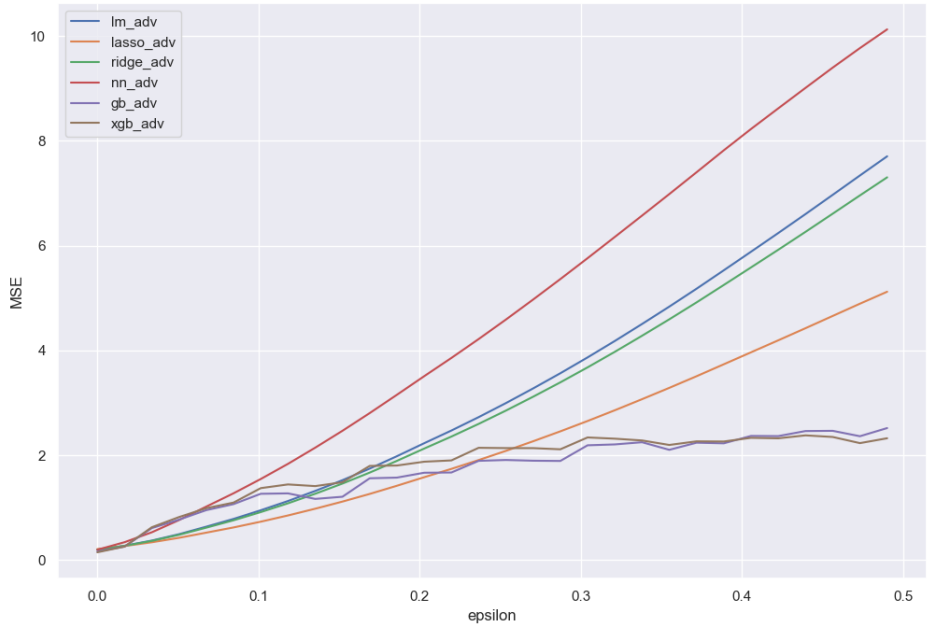

Figure 3: Mean squared error on 200 adversarial examples for different values of ϵ.

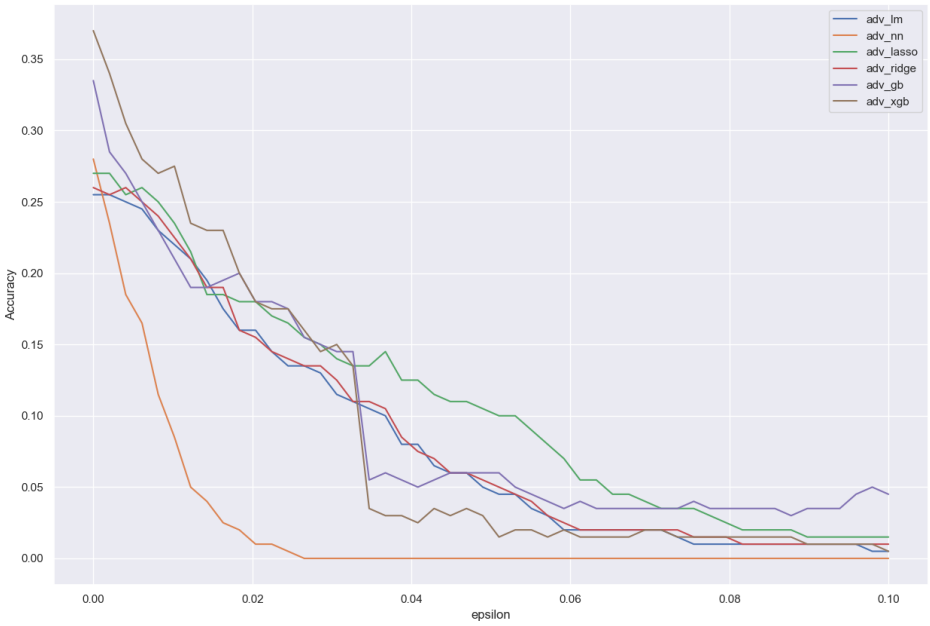

Figure 4: Accuracy for the classification task on 200 adversarial examples for different values of ϵ ($\delta = 0.15$).

The second approach is summarized on Fig 4. Once again, the neural network's performance drops significantly and very quickly. The Lasso linear model outperforms Ridge and the standard LM. Both gradient boosting models show a drop around $\epsilon = 0.034$ for reasons that are unexplined for now. It is also surprising that XGBoost performs worst that the standard gradient boosting model on the long run, as the former introduces more regularization that the latter.

Conclusion

This paper provided an great basis for improvement and experiments. The robustness analysis provided interesting, sometimes surprising insights on the models used and their resistance to adversarial attacks or compromised data. While we tried an NN-specific adversarial attack, an avenue for development would be to design attacks specific to other model classes (tree-based models, for instance) and test the robustness of different models to them.

References

[1] Pouya Rezazadeh Kalehbasti, Liubov Nikolenko, and Hoormazd Rezaei. "Airbnb Price Prediction Using Machine Learning and Sentiment Analysis". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 173–184. DOI: 10.1007/978-3-030-84060-0_11. URL: https://doi.org/10.1007%2F978-3-030-84060-0_11.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].