Nutritional Label for Zillow's Home Value Prediction ADS

Project Report

Helen Zhou Nicholas Lin

Background

Zestimate is online real-estate marketplace company Zillow's proprietary home valuation ADS. Zestimate's stated purpose is to give prospective and current homeowners accurate assessments of both home information and valuation. However, given that this system is intended for property valuation, its effects on a community's financial and perceived value cannot be ignored. Any biased or unintended processes within the algorithm must therefore be both understood and minimized to the best of their ability when it comes to individuals in protected groups. Therefore there are some trade-offs within this stated goal, especially in regards to the algorithms and decision-making processes underlying Zestimate, which are incredibly complex, often hidden, require layers of assumptions, and are often built on datasets not originally intended to be used with this ADS. Because of this, the best way to achieve this intended and stated goal is to minimize bias, maximize accuracy, and further public understanding of this ADS. In this ADS, this minimization was achieved by optimizing Zillow's price predictions by generating improved estimated log differences of prices.

Input and Output

Given real estate transactions within the United States are publicly available, this ADS implemented data regarding real estate transactions in three Californian counties in 2016: Los Angeles, Orange, and Ventura county. All input datasets are provided by Zillow from the Kaggle website. Although the transactions were recorded, the training dataset simply calculates the log-error of the Zestimate algorithm in an attempt to preserve differential privacy. The input training data contained log-error and transaction dates for properties through the aft-mentioned Californian counties from January 1 to December 30, 2016. All entries in the training dataset, or the log-error dataset, are valid float data types.

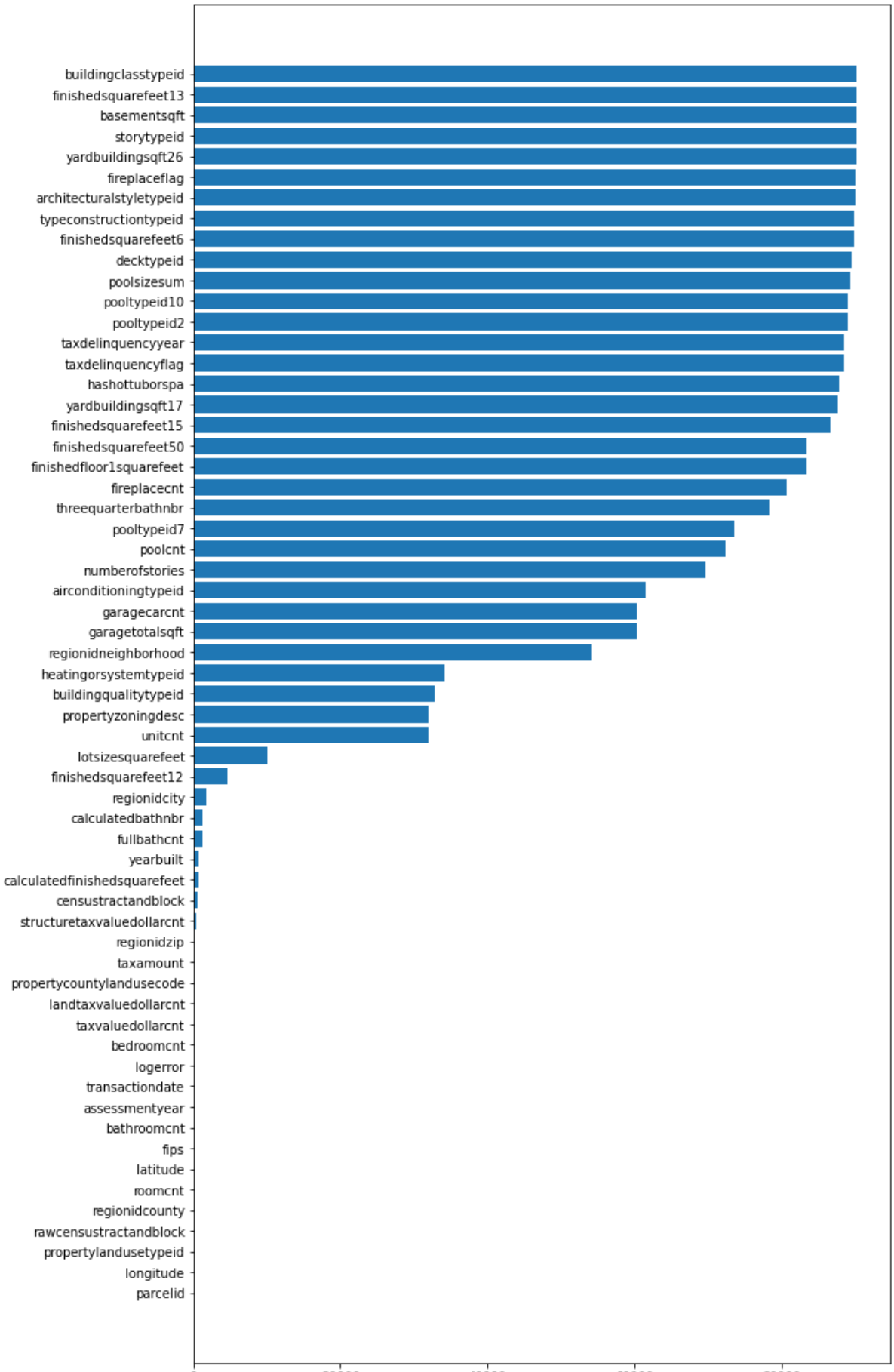
Various features were also given in a separate properties dataset such as home structural information, renovative features, and notably residential and locational identification features. These data were also used as input data and provided by Zillow. Structural information concerned the number of rooms and bathrooms or calculated living area of the home. Renovative features included home pool count or even the presence of a home heating system. Most of these

fields are either integer or float type numerical data. Unavailable information is marked as NaN values within the properties dataset.

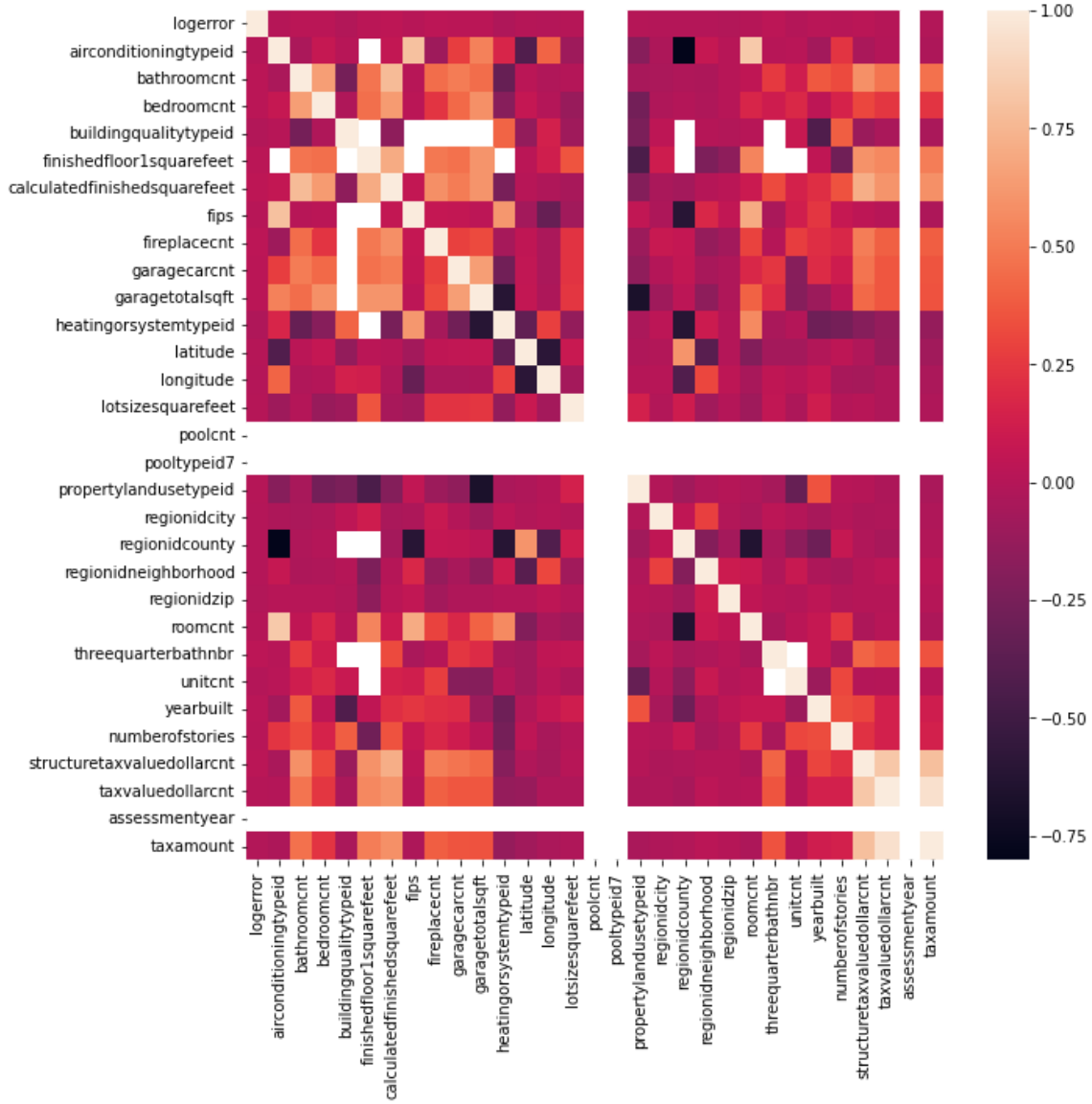
The training dataset was merged with the properties dataset in an attempt to obtain a better visual understanding of the data. Below is a full list of the features in the combined trending dataset. Most features are represented as float data types, with a few exceptions of features with integers, timestamps, and object data types.

parcelid	int64
logerror	float64
transactiondate	datetime64[ns]
airconditioningtypeid	float32
architecturalstyletypeid	float32
basementsqft	float32
bathroomcnt	float32
bedroomcnt	float32
buildingclasstypeid	float32
buildingqualitytypeid	float32
calculatedbathnbr	float32
decktypeid	float32
finishedfloor1squarefeet	float32
calculatedfinishedsquarefeet	float32
finishedsquarefeet12	float32
finishedsquarefeet13	float32
finishedsquarefeet15	float32
finishedsquarefeet50	float32
finishedsquarefeet6	float32
fips	float32
fireplacecnt	float32
fullbathcnt	float32
garagecarcnt	float32
garagetotalsqft	float32
hashottuborspa	object
heatingorsystemtypeid	float32
latitude	float32
longitude	float32
lotsizesquarefeet	float32
poolcnt	float32
poolsizesum	float32
pooltypeid10	float32
pooltypeid2	float32
pooltypeid7	float32
propertycountylandusecode	object
propertylandusetypeid	float32
propertyzoningdesc	object
rawcensustractandblock	float32
regionidcity	float32
regionidcounty	float32
regionidneighborhood	float32
regionidzip	float32
roomcnt	float32
storytypeid	float32
threequarterbathnbr	float32
typeconstructiontypeid	float32
unitcnt	float32
yardbuildingsqft17	float32
yardbuildingsqft26	float32
yearbuilt	float32
numberofstories	float32
fireplaceflag	object
structuretaxvaluedollarcnt	float32
taxvaluedollarcnt	float32
assessmentyear	float32
landtaxvaluedollarcnt	float32
taxamount	float32
taxdelinquencyflag	object
taxdelinquencyyear	float32
censustractandblock	float32
living_area_prop	float32
value_ratio	float32
value_prop	float32

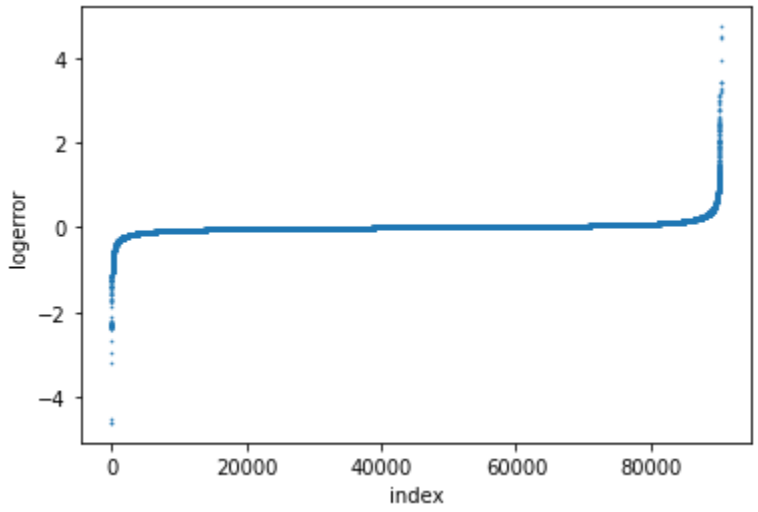
The merged dataset is a table with 63 columns (features) and 90,275 rows. Although all values of *parcelid*, *logerror*, and *transactiondate* are provided, many fields in other features are missing and subsequently labeled as NaN. Within the context of this missing data, the number of such missing data in each column, as well as the percentage of missing data, are recorded. In the horizontal bar graph below, all 63 features have their missing data counts shown. Some features such as *decktypeid*, *finishedsquarefeet6*, t*ypeconstructiontypeid* are mostly empty, with approximately 99% of the data within these features are NaN data, and as such do not provide valid information about properties. On the other hand, *logerror*, *transactiondate*, *bathroomcnt* (bathroom count), *bedroomcnt*, and other features which contain zero missing data provide more detailed information about each property.


buildingclasstypeid
finishedsquarefeet13
basementsqft
storytypeid
yardbuildingsqft26
fireplaceflag
architecturalstyletypeid
typeconstructiontypeid
finishedsquarefeet6
decktypeid
poolsizesum
pooltypeid10
pooltypeid2
taxdelinquencyyear
taxdelinquencyflag
hashottuborspa
yardbuildingsqft17
finishedsquarefeet15
finishedsquarefeet50
finishedfloor1squarefeet
fireplacecnt
threequarterbathnbr
pooltypeid7
poolcnt
numberofstories
airconditioningtypeid
garagecarcnt
garagetotalsqft
regionidneighborhood
heatingorsystemtypeid
buildingqualitytypeid
propertyzoningdesc
unitcnt
lotsizesquarefeet
finishedsquarefeet12
regionidcity
calculatedbathnbr
fullbathcnt
yearbuilt
calculatedfinishedsquarefeet
censustractandblock
structuretaxvaluedollarcnt
regionidzip
taxamount
propertycountylandusecode
landtaxvaluedollarcnt
taxvaluedollarcnt
bedroomcnt
logerror
transactiondate
assessmentyear
bathroomcnt
fips
latitude
roomcnt
regionidcounty
rawcensustractandblock
propertylandusetypeid
longitude
parcelid
0
20000
40000
60000
80000

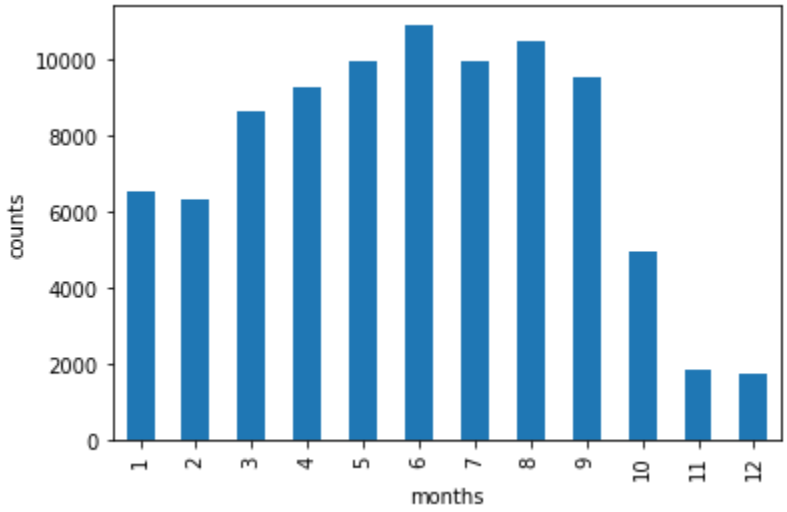
The correlation between features was evaluated through the visualization using a heatmap. Features with high missing data rates or extreme correlation values were eliminated from the correlation matrix. In the end, 31 features and their pairwise correlations are shown in the heatmap provided below. Essentially the heatmap shows the magnitude of a phenomenon using different intensities of colors. We see that logerror obtains weak correlations with all features. Besides that, *roomcnt* appears to have a moderate positive correlation with *airconditioningtypeid*. *calculatedfinishedsquarefeet* also has a moderate positive correlation with *bathroomcnt*. *Regioncountry* has a moderate negative correlation with *airconditioningtypeid*.



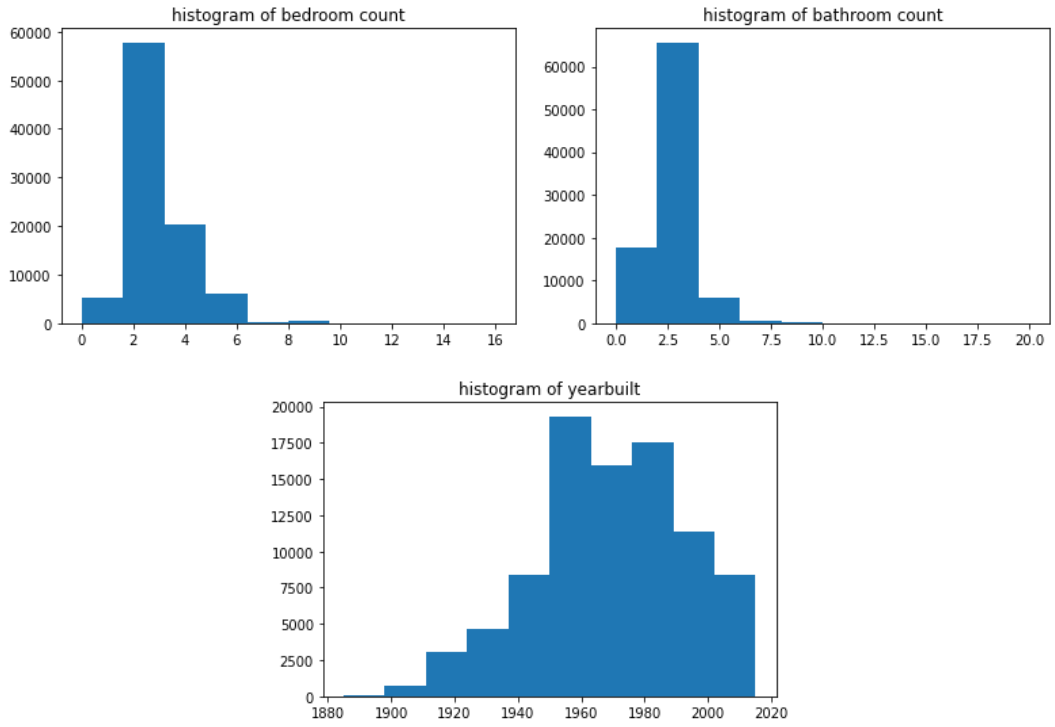
The log error provided by the training set is plotted in the graph below. The log errors in the training dataset have a mean of 0.01146 and a standard deviation of 0.16108. The majority of log errors are close to zero and scattered through the interval -4.60500 (minimum) to 4.7370 (maximum).



Transaction date is an informative feature without any missing values, as such a bar plot of it is shown below. We see the majority of the transactions being made between March and September and fewer sales in January and February, and even more significantly less from October to December.



Most of the properties in the dataset have valid bedroom and bathroom counts. The mean bedroom count is 3 and the 2 for bathrooms. Therefore in the dataset, an average house has three bedrooms and two bathrooms. Utilizing a histogram showing building years for homes in the dataset allows us to see that a majority of houses were built between 1950 and 1990.



The residential and locational identification features are especially noteworthy as this was an area where protected groups could be disparately impacted the most. Therefore it would be this input feature that we would have the most interest in, especially in terms of determining its weight in the algorithm and also how exactly it utilizes residential information and markers into its algorithmic determination of a property assessment.

As stated previously, the output of the system in this case is the log of the actual sale price of the property subtracted from the log of the algorithmic evaluation that Zestimate placed on that property. We can interpret this output simply as the accuracy of the algorithm trained on our training data. Since we interpret the data in this fashion, we can also determine how implied accuracy, in this case, is affected by different mitigation factors we can implement in order to decrease disparate impact or bias towards protected groups.

Implementation and Validation

Kaggle, a community based around data science and machine learning projects and competitions, was the site of a competition intended to improve the performance of the Zestimate ADS. The specific ADS investigated was a published solution to the aforementioned Kaggle public-sourced competition written by Dr.Usmani. As stated previously, it was imperative that there was a certain amount of differential privacy within the functioning of the algorithm, as such

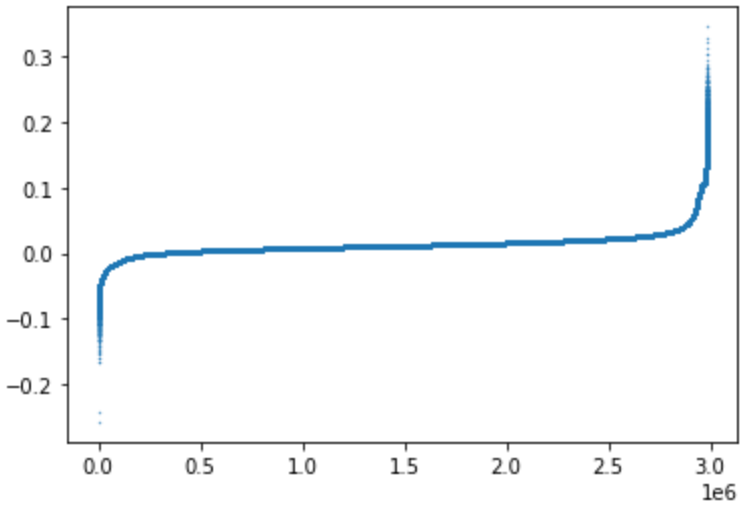
the actual sales prices of the property were hidden within the actual training dataset. Furthermore, in the public round of this challenge, in order to improve upon Zestimate, automated decision systems were required to predict the log error for given property values at six time points. This performance was rated in terms of their intended goals, in terms of the logerror of the algorithm, and was subsequently validated through this same method. This method of validation allowed for a numerical, unbiased, and logical evaluation of the efficacy to how this ADS accomplished the goals established for it. The logerror estimation within the algorithm was calculated using the equation: $logerror = log(Zestimate) - log(SalePrice)$.

Because many features in ADS obtained a large amount of missing data as shown previously, data imputations, cleaning, and label encoding were employed for pre-processing. Some features that were missing can be derived from existing data. For example in some properties, the boolean value for fireplace was missing while the fireplace count was not. Therefore, some missing boolean values were able to be updated based on valid and closely related features. Other missing values within relevant columns are replaced with zero.

The training data was then label encoded to convert categorical variables into numerical forms. Because the machine learning algorithm deployed by the ADS only accepted numerical data, label encoding helped prepare the training dataset for further modeling. After converting the dataset into numerical form, outliers of logerror columns were removed from the dataset, and features that are not human comprehensive or meaningful are dropped in this stage too. Examples of some of these dropped features included *parcelid*, *logerror*, and *transactiondate*. The original training set, which includes log errors and properties for estates transacted in 2016, was further splitted into a training set and valid set.

The Xgboost module for model training was employed in ADS. Cross validation was also adopted to monitor if the model overfitted the training dataset. In each modeling process, the xgboost trains one model and improves its next model by learning errors from the previous model. After multiple models were created, the model with the best result that had acceptable overfitting was preserved. The ADS then used the trained model to predict log error for estates in the sample submission file provided by the competition host. Instead of predicting different log errors for different time points, six columns that represent six time points were imputed with one array of log error prediction. Therefore, the ADS disregard the time factors and only predicted the log error based on given properties.

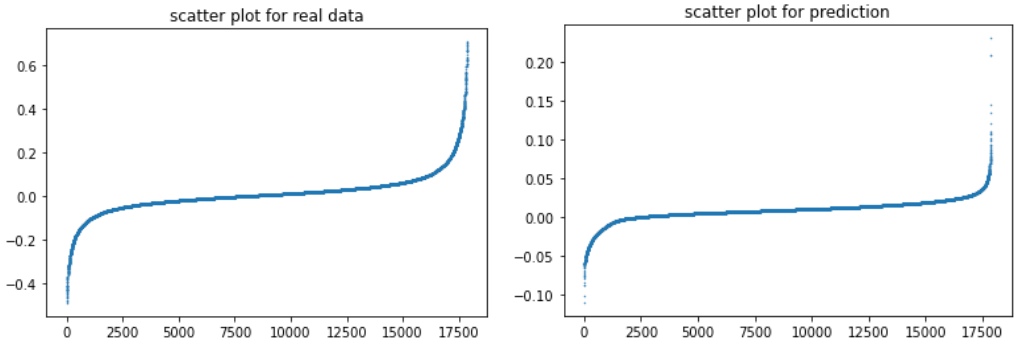
Given that every row of calculated logerror was validated, we calculated the statistics and plotted the log error in the scatterplot below. The majority of the logerror are close to zero with a few outliers like minimum(-0.25805) and maximum(0.34636) logerror. The mean for prediction was 0.013022 and majority of log errors are close to zero, indicating that Zestimate is close to real sale price.



The ADS scored a private score of 0.07609 and a public score of 0.06489 in the public sourced competition which it was implemented for, validating its predictions while also meeting its stated goals of preserving privacy to a certain degree. However, because it disregarded calculation for 6 different time points of data and only predicted one column of unique log errors for an unspecified date, it did to some degree fail to fully complete the competition's requirements for ADS design and implementation.

Outcomes

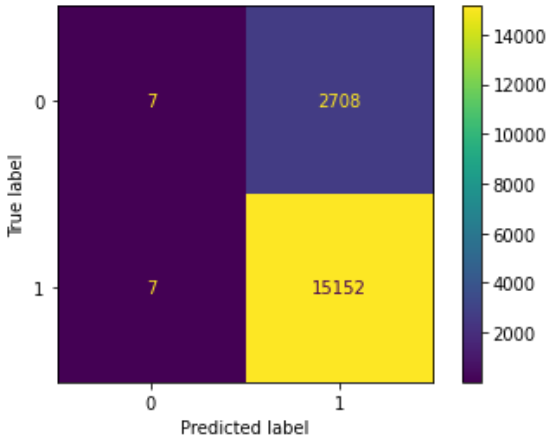
Since the real data for log error prediction in the submission file was a key metric for scoring, it was not provided to the public. Therefore instead of analyzing the submission version of predictions, we investigated the ADS using the test set provided, *yvalid*, which was a split set of the original training set before modeling. Predictions were made based on the test data and were called *y_pred*. On this test dataset, we compared the distribution of real log errors of the test set and predicted log errors in the graphs below. The predicted log errors had a mean around 0.00954 while real data had a mean around 0.01179. The real data spread out in a broader interval from -0.4878 to 0.7066. On the contrary, the predicted log errors only spread out between -0.10962 and 0.23126.



We then ran a simple classification method on both the real and predicted data. Data with an absolute value less than or equal to 0.1 was classified as 1, representing that Zestimate was close to the real sale price. Otherwise, the log error was classified as 0, representing an inaccurate estimate. Within the real data, 2715 out of 17874 cases were classified as 0 (inaccurate estimate) while 15159 cases were accurate estimates. In the predicted data, only 14 were inaccurate estimates while 17860 cases were accurate estimates. The general accuracy was about 0.848, the precision is around 0.84838, and the Area Under Curve (AUC) has a value of approximately 0.50106.

```
accuracy: 0.8481033903994629
AUC: 0.5010582484959611
precision: 0.8483762597984322
recall: 0.999538228115311
average_precision: 0.8483761337950155
```

A confusion matrix was then generated based on this classification. We see most incorrect predictions are False Positive cases, where cases have a real log error lower than 0.1 but a prediction of log error higher than 0.1. To put it simply, this just means that bad estimates at the extremes of poor calculation are incorrectly predicted as good estimates. These specific false positive rates and false negative rates were calculated and also provided below.




```
false positive rate is: 0.9974
false negative rate is: 0.0005
```

Fairness in such a system is essential and both the false positive and false negative rates give us important insights into the performance of such a system in the case of both group and individual fairness. However, due to the extremely high false positive rate within this algorithm, it is hard to give this system a vote of confidence in either category. Despite this, such a stark contrast in false positive and false negative rates is not the only measure of fairness. We then observe the predictive parity of this algorithm, also known as the overall calibration. Even in this instance, it can be said that this algorithm is not reasonably well-calibrated as it seems to be chronically overshooting on predictions which regardless results in a bad estimate despite what its actual impact is. A bad estimate is a bad estimate. A final note about fairness based on the FP and FN rates and calibration of the system is that although the FP rates are high and calibration is not well-fitted, the system does not seem to fail differently across different groups. So although

fairness is questionable based on observations, actual disparate impact and bias within protected groups are not concretely observed

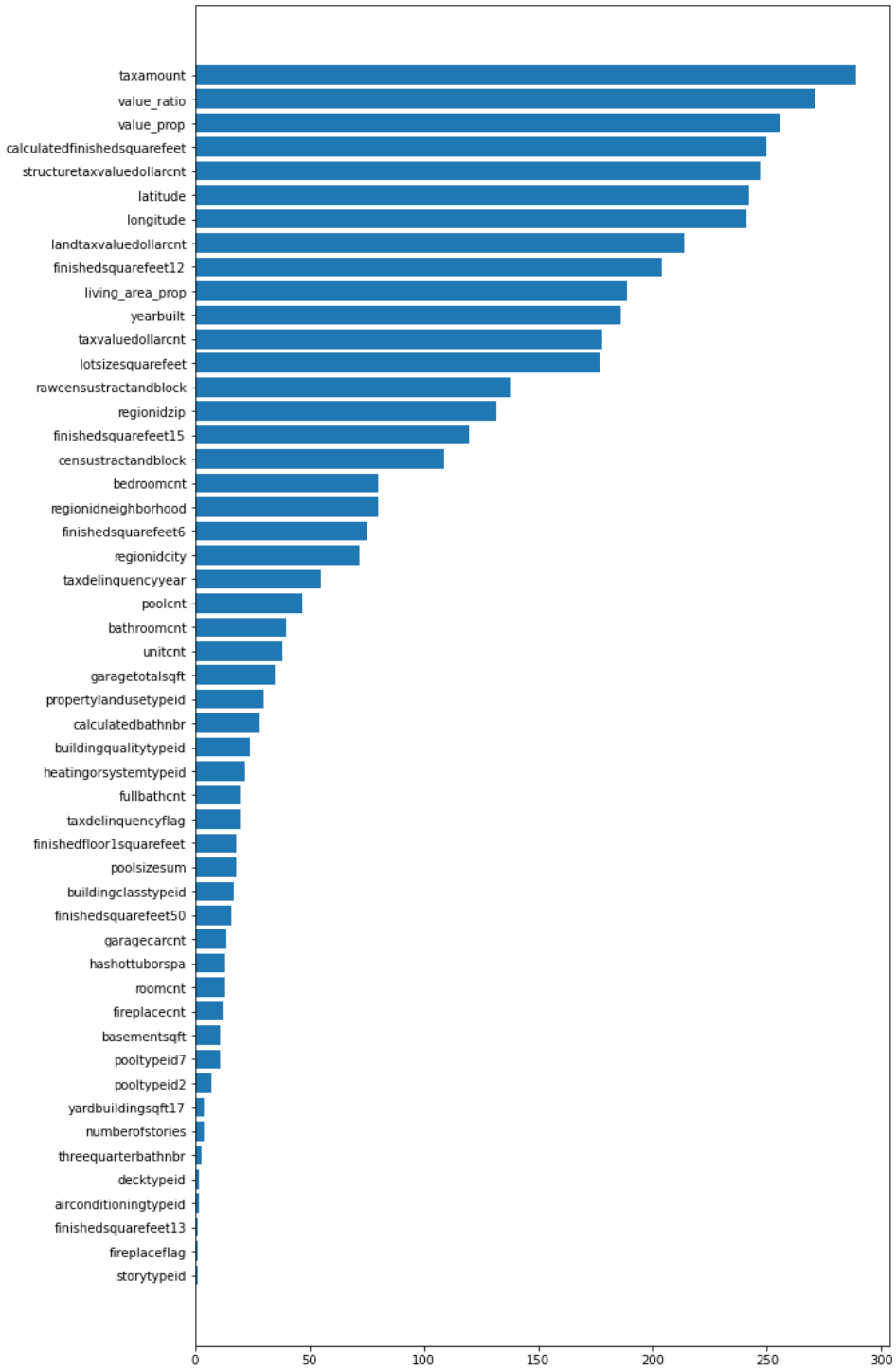
Branching off of our analysis above about fairness of this algorithm, we then intended to explore machine learning performance across different subpopulations. This was difficult as the ADS converted data into numerical values during the division into training and test datasets. This converted data type was then stored into array data structures, essentially making it nearly impossible to reconstruct it into a new dataframe that could be used to reassociate predictions with features. Likewise, many features such as *buildingclasstypeid*, *storytypeid*, *architecturalstyletypeid* featured data which was composed of roughly 99% of null data. A separate issue with features such as *regionidcounty*, *regionidcity*, and *regionidneighborhood* was that these features were coded and not human-interpretable. In addition, no external dictionaries were provided for interpreting these codes.

Despite these setbacks, we continued with a different method in order to analyze performance within population subgroups by using the MetricFrame data structure using a call to within the Fairlearn Metrics package. This allowed for functionality within computing metrics, especially disaggregated metrics which was especially applicable in our case, based on one sensitive feature which in this case was *propertylandusetypeid*. However, as we continued this endeavor we eventually realized that reconstruction of associated features within this prediction was also not feasible.

Therefore although it was of utmost importance for us to create concrete visualization of the differences in performance of this Automated Decision System across multiple subpopulations, we eventually came to realize that we would be unable to do it in a way that could easily be explained or interpreted in an accurate way by users and relevant stakeholders. This is not a unique issue within our field of data science and analysis, being that many models are either purposely or unintentionally masked over layers of data manipulation, pre and post-processing, or even simply omission of certain key features. Such “black-box models” are frequently used in this industry of processing and understanding data which too often results in disparate impact or bias affecting protected groups. Black-box models fundamentally go against one of the core principles of modeling and classifying within our field, being that classifiers must have good explanations within their implementation. These explanations must be interpretable, faithful in terms of representing how a model actually behaves, and model agnostic in regards to being to be utilized for any and all ML models. As stated previously, this pursuit of proper classification explanation was intended to be essential for our application of this system。 However, limitations within the transparency of this system such as specific encoding, pre-processing methods, and missing or inconsistent documentation made this too complex to be presented in a way in which we felt comfortable pursuing.

The effectiveness of the ADS however does not seem to be impacted across different subpopulations as stated above. Although there is a presence of a high FPR, the system does not fail differently across subpopulations and different groups. This is because the accuracy of the algorithm is not affected by the presence of specific subpopulations. There are features within the training datasets that do take into account certain attributes that could be seen as being protected, features such as zip-code, city name, or even county identification. However, these features were masked by data typing which resulted in these characteristics not being fully representable in visualization for influence on the algorithm. Despite this lack of categorical and consistent data typing, we were able to demonstrate the influence of features using the heatmap model employed above, which indicated that no one feature was skewing predictions in a biased manner, which is a very favorable sign for assessing the influence of various subpopulations and other features.

As this Automated Decision System employed an Xgboost model, we took advantage of this by continuing analysis of feature importance within our algorithmic model using built-in functions within XGBoost's distributed gradient boosting library. Using built-in functions of this library, we were able to obtain feature weights in a way that could easily be visualized and understood. Analysis of this visualization shows that *taxamount* holds the highest factor weight within the model while more categorical renovative features such as *storytypeid* and *fireplaceflag* have minimal influence and weight. Based on this analysis we continued on to endeavor with LIME analysis to further explore feature influence and importance in specific cases within the data. However, due to the issues within the algorithm in terms of encoding and pre-processing methods and missing documentation briefly mentioned above, we were unable to reliably and accurately retrieve feature information based on specific predictions made by the model.


taxamount
value_ratio
value_prop
calculatedfinishedsquarefeet
structuretaxvaluedollarcnt
latitude
longitude
landtaxvaluedollarcnt
finishedsquarefeet12
living_area_prop
yearbuilt
taxvaluedollarcnt
lotsizesquarefeet
rawcensustractandblock
regionidzip
finishedsquarefeet15
censustractandblock
bedroomcnt
regionidneighborhood
finishedsquarefeet6
regionidcity
taxdelinquencyyear
poolcnt
bathroomcnt
unitcnt
garagetotalsqft
propertylandusetypeid
calculatedbathnbr
buildingqualitytypeid
heatingorsystemtypeid
fullbathcnt
taxdelinquencyflag
finishedfloor1squarefeet
poolsizesum
buildingclasstypeid
finishedsquarefeet50
garagecarcnt
hashottuborspa
roomcnt
fireplacecnt
basementsqft
pooltypeid7
pooltypeid2
yardbuildingsqft17
numberofstories
threequarterbathnbr
decktypeid
airconditioningtypeid
finishedsquarefeet13
fireplaceflag
storytypeid
0
50
100
150
200
250
300

Summary

In terms of the testing and training datasets used within the Zestimate algorithm, we believed it is not completely appropriate or accurate in terms of being a representative sample. This is because both the testing and training dataset are limited in terms of demographic and representative location, being that the datasets focus on three counties in the state of California, all three of which are mostly well-off and demographically homogeneous areas, far from representative of a national or even global sampling of the assessment and value of properties, neighborhoods, and communities.

The implementation of the Zestimate algorithm was certainly robust with predictions being made on a myriad of features within the split training dataset. However, in terms of accuracy, this algorithm did have some flaws which cannot be overlooked or understated. As mentioned above, the predictions made by this Automated Decision System did seem to overfit predictions to a degree which brings up very important questions. One of these questions surrounds the fairness of the algorithm, especially when considering the tradeoffs inherent within the fairness in classification framework. It is important to address the issue of bias within the fairness of the system even though the algorithm is robust. One of the biggest concerns is how technically sound the algorithm is, especially given the shockingly high False Positive Rate (FPR). And although the FPR does not seem to be playing off an excess of technical or emergent bias within the system, it is still important to note that the implementation of the predictions does result in frequent overshooting of property values even if there is no clear skew in direction. To put it succinctly, an inaccurate estimate is a bad estimate and a bad estimate is a bad estimate regardless of the context.

Regardless of there not definitively being technical or emergent bias within the design and operation of the algorithm, there is a certain amount of pre-existing bias “baked into” the operation of this ADS. In this regard, the False Negative Rate (FNR) of the algorithm is incredibly low so this allows us to assume that there is not a specific group or subpopulation which is being disparately impacted. We then must consider the predictive parity, or calibration, of the system, which given that the system is chronically overshooting given the high FPR we see that the calibration is not well-fitted in the system however we can also come to the conclusion that the system does not fail differently across different groups.

Given these considerations, we do not believe that this current implementation of the Zestimate ADS can and should be deployed in the public sector. This is because although the evaluation algorithm is well established and most parts can be explained and understood by the public relatively easily, it does fail to be completely transparent and explain the entirety of its classifiers. Many of its most important features, such as effects across different subpopulations and disparate impact of implementation are still largely gatekept which makes some of its most important features black-box models. Additionally and perhaps even more importantly, this

implementation of the Automated Decision System was trained on limited data provided by Zillow in the public round of evaluation and thus its output is limited to exclusively log error calculations. And although log error allows for fantastic evaluation of accuracy and other metrics at a glance, as a standalone evaluation system it does provide limited accessibility and information to the public sector. The log error calculation only provides a calculation between the ADS prediction and a real sales price while not giving other features and context which may affect public understanding of such an algorithm.

We would suggest improvements to this automated decision system through the implementation of more representative and complex training and testing datasets. This would allow for more distributive and representative data collection and result in a more representative and "fair" algorithm. This concept of fairness is summarized above, and as such, understanding how more demographically diverse and representative data would play into the training and operation of this ADS is imperative. In terms of data processing, the Zestimate algorithm is robust however in terms of public sector deployment there would have to be a way in which feature and result outputs could be more readily accessible and interpretable other than a complex log error calculation. Differential privacy is essential within the functioning of an algorithm. In addition, interpretability is also one of the most important attributes when it comes to the explainability of a classifier. Finally, in terms of analysis, features of the algorithm that fall under a black-box model must be further explored and improved upon. The features must be readily interpretable as the algorithm must be clearly understood by any and all stakeholders interested in the accuracy of the algorithm within the inherent processing and functioning of said ADS.

Citations:

Zillow Zestimate Kaggle Competition:
https://www.kaggle.com/c/zillow-prize-1/overview

Solution Code by ZEESHAN-UL-HASSAN USMANI:
https://www.kaggle.com/code/zusmani/scrpt/script

Nutritional Labels for Data and Models :
http://sites.computer.org/debull/A19sept/p13.pdf

A Unified Approach to Interpreting Model Predictions:
https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Collab Notebook:
https://colab.research.google.com/drive/1PKsL3btGJSaXmxhzovsp_o7WeerT81af?usp=sharing