# Machine Learning Nanodegree
# Capstone Project
# Zillow Prize: Zillow's Home Value Prediction

Javier Fuentes

August 19, 2017

## 1 Definitions

### 1.1 Domain Background

Zillow's "Zestimate" [1][2] home valuation has shaken up the U.S. real estate industry since first released 11 years ago.

A home is often the largest and most expensive purchase a person makes in his or her lifetime. Ensuring homeowners have a trusted way to monitor this asset is incredibly important. The "Zestimate" was created to give consumers as much information as possible about homes and the housing market, marking the first time consumers had access to this type of home value information at no cost.

"Zestimates" are estimated home values based on 7.5 million statistical and machine learning models that analyze hundreds of data points on each property. And, by continually improving the median margin of error (from 14% at the onset to 5% today), Zillow has since become established as one of the largest, most trusted marketplaces for real estate information in the U.S. and a leading example of high-impact machine learning.

### 1.2 Problem Statement

This is a regression problem. The data is a collection of features related to the property and the $logerror$, defined bellow.

$$logerror = log(Zestimate) - log(salesprice)$$

In the Kaggle's competition [3], Zillow is asking to predict the log-error between their "Zestimate" and the actual sale price, given all the features of a home.

### 1.3 Evaluation Metrics

The evaluation metric for this project will be Mean Absolute Error(MAE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

Where $y$ and $\hat{y}$ are actual the value and the predicted value respectively. This is also the same metric used by the Zillow competition.

Also, as a loss function, the Mean Squared Error was used:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

# 2 Analysis

## 2.1 Data Exploration

All data is available on the Kaggle competition page. The data is divided into three files:

- **properties_2016.csv**: This corresponds to all properties with their home features for 2016. There's a total of 57 features per property. Such as a location, a number of rooms, the year it was built, etc. It contains nearly 3 million properties on the record. Some features are numerical, and others contain categorical data.
- **zillow_data_dictionary.xlsx**: All features are well-defined into this document, which contains details of the nature of each.
- **train_2016_v2.csv**: Which includes all the $logerror$ values and the date of transaction in 2016. This file contains $90,275$ records of transactions, and both file indexes share the same identifier: $parcelid$.

The dataset of properties contains $2,985,217$ elements, but the train file just $90,275$ transactions with their respective $logerror$ values. This is because the Zillow competition is asking to predict values for those missing values. Merging both datasets we obtain a single collection of data with $90,275$ property descriptions and 59 features per property.
In the table below a selection of example features and values is shown:

| logerror | transactiondate | taxamount | bathroomcnt | regionidzip | regionidzip |
|---|---|---|---|---|---|
| 0.0276 | 2016-01-01 | 6735.88 | 2.0 | 96370.0 | 96370.0 |
| -0.1684 | 2016-01-01 | 10153.02 | 3.5 | 96962.0 | 96962.0 |
| -0.0040 | 2016-01-01 | 11484.48 | 3.0 | 96293.0 | 96293.0 |
| 0.0218 | 2016-01-02 | 3048.74 | 2.0 | 96222.0 | 96222.0 |
| -0.0050 | 2016-01-02 | 5488.96 | 2.5 | 96961.0 | 96961.0 |

Table 1: Both datasets merged showing features and data as examples

The dataset contains categorical and linear data. The next table shows an example of categorical data present:

| AirConditioningTypeID | AirConditioningDesc |
|---|---|
| 1 | Central |
| 2 | Chilled Water |
| 3 | Evaporative Cooler |
| 4 | Geo Thermal |
| ... | ... |

Table 2: Categorical feature example.

For more details, the **zillow_data_dictionary.xlsx** file contains all descriptions of the features.

## 2.2 Exploratory Visualization

At the beginning of the exploratory analysis, the most relevant variable to understand it was the target variable: the log error.

As can be seen in Figure 1, a scatter plot was performed. This plot makes it easier to see how distributed the log error is, also through visual exploration it is possible to see some outliers points, farther than −0.4 to 0.4.

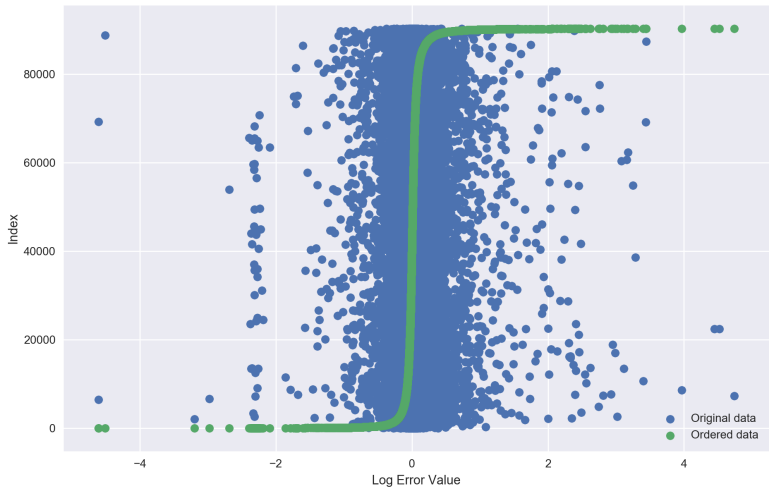


Figure 1: Outliers detection.

Another interesting analysis is the monthly log error average which is in Figure 2 and shows that the most accurate months were April, May, and June, besides the most inaccurate month, which was December.

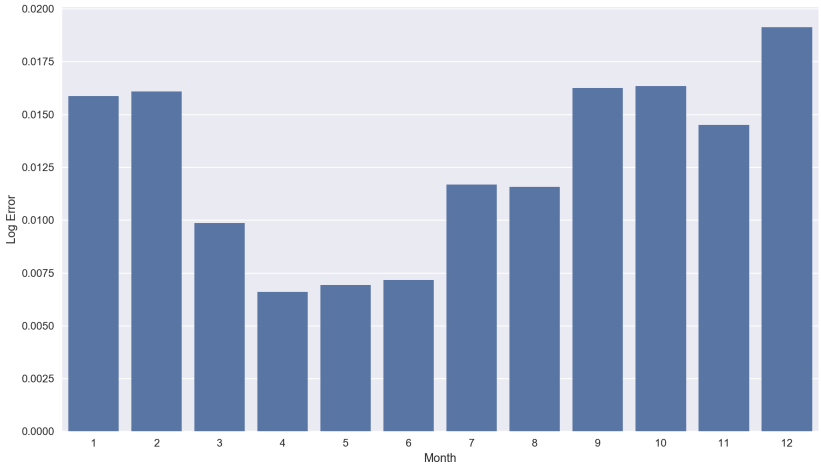


Figure 2: Log Error Monthly Average.

One of the biggest issues with the dataset is that there are plenty of missing values on features, which represent a complicated scenario, in terms of how to replace those missing values or if to drop features or to compute predictions for replacement. As is shown in Figure 3, the first 2 quantiles are almost based on complete values, but in the next 2 quantiles there is a lot of missing data.

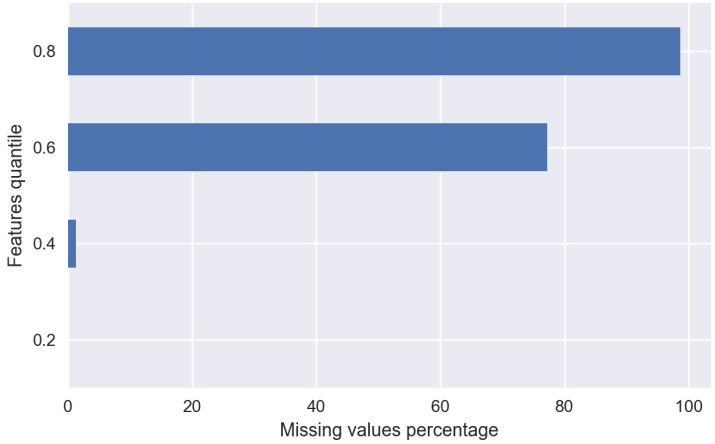


Figure 3: Missing Values on Features.

Using an ensemble method for regression is suitable to obtain a feature relevance. The selected method was `ExtraTreesRegressor` from the scikit learn framework. The results are shown in quantiles in in Figure 4. In comparison with the distribution of missing values in Figure 3, the feature relevance is better distributed among the quantiles. This means that the lack of data did not significantly affect in the features.

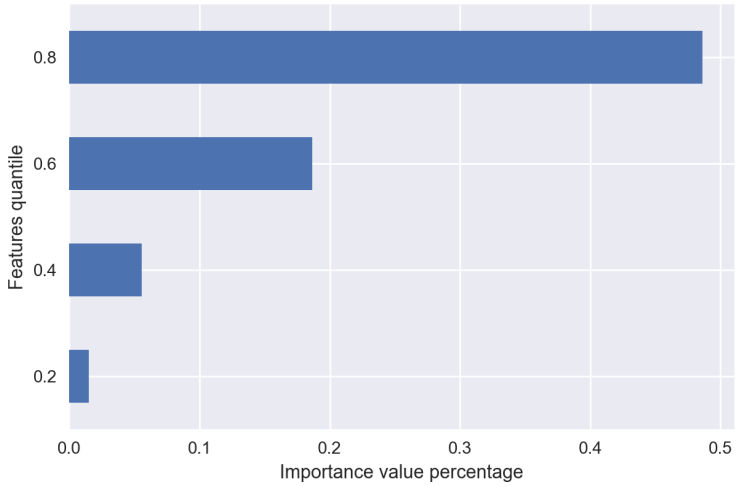


Figure 4: Feature Importance Analysis.

## 2.3 Algorithms and Techniques

In this project two tasks were addressed, the first is a regression task that corresponds to a function approximation, or a prediction on a single variable. The other task is classification, which in this project works as an extension of regression using the range of the function. It is possible to set an arrangement of "classes" that represents a value in the continuous space of the range of the function. This process is called discretization and it is used in granular computing.

To perform these tasks in this project, a Multilayer perceptrons were used.

Artificial Neural Nets are part of state of the art classification and regression tasks.

The next parameters can be tuned in both training processes:

- Number of classes to predict on.
- Number of epochs
- Batch size
- Learning rate
- Optimizer function
- Loss function

The next parameters can be tuned in both models:

- Number of hidden layers
- Model architecture.
- Activation functions

## 2.4 Benchmark

On the Kaggle competition, there is plenty of example projects, mostly based on a variety of ensemble methods. A XGBoost approach[4] was used as a reference as it is a common way to handle the problem.

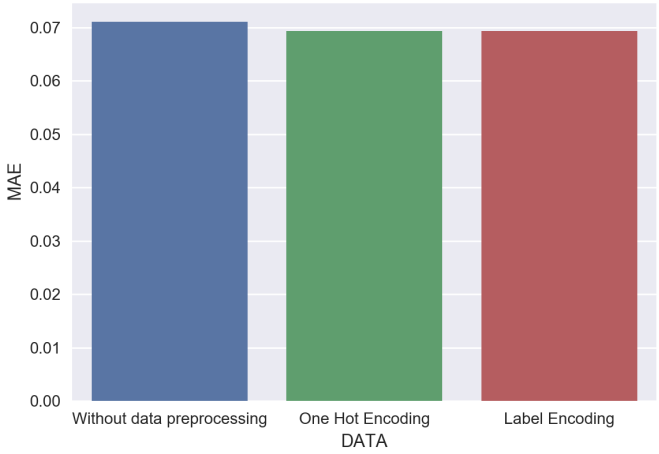


Figure 5: MAE using differents preprocessed datasets.

# 3 Methodology

## 3.1 Data Preprocessing

Preprocessing data is a non-trivial task, and in many machine learning approaches, it could increase performance. The method used to preprocess data is as follows:

1. Fill blank spaces with the mean of each feature.
2. Parse the transaction date into a continuous value.
3. Parse all boolean features based on strings, like ('Y', 'N') into 1.0 and 0.0.
4. Compute feature importance to determine some feature dropping ratio.
5. Parse categorical data:
    - One Hot Encoding[5].
    - Label Encoding[6].
6. Scale all features on a range between [0.0, 1.0].

For data discretization:

- Compute minimum and maximum of the target variable.
- Scale the target variable between zero and the number of classes.
- Fill an array of zeros with length equal to the number of classes.
- Apply the scaled target number to correspond to the index and change the value at that index position to 1.

It is possible to adjust the following parameters in the preprocessing stage:

- Allowance importance feature rate for dropping columns.
- Type of encoding, Label or One Hot Encoding.

## 3.2 Implementation

The implementation of this project is based on two stages:

- Generating preprocessed datasets.
- Training models.

The second stage process it was defined in a base model as it is shown below in Listing 1:

```
class ZillowBaseModel():
    def __init__(self):
        pass
    def run_default(self):
        pass
    def set_dataset(self, n_classes=100, train_size=.7, test_size=.5):
        pass
    def build_model(self):
```

```
    pass
def set_optimizer(self, learning_rate=.1):
    pass
def train_model(self, training_epochs=1000, batch_size=32, logs_path=None):
    pass
```

Listing 1: Base model on `zillow_project/data.py`

The sequence of actions is as follows:

1. Load the preprocessed dataset and set features and the target variable.
   (a) In the classification approach, the target variable is also discretized into an array of certain length using the next linear formula:

$$new\_y = \frac{size\_array}{max\_y - min\_y} \times (original\_y + min\_y)$$

   Where $new\_y$ is the transformed target variable which is the index of the array, and $min\_y$ and $max\_y$ are the minimum and maximum values respectively of the original value.
   (b) In the regression approach it was used the same variable without modifications.
2. The next step is build the model of the neural net, as it is shown in Table 3 for both tasks.
3. Next the optimizer and the loss function is defined. In both models it was used the Mean Squared Error as loss function, and Adam method as optimizer.
4. And finally the model it is trained, logging MAE on the training and validation sets.

| Layer | Dimension | Activation Classification | Activation Regression |
|---|---|---|---|
| Dense Layer | 4096 | Relu | Relu |
| Dropout | Prob(.2) | None | None |
| Dense Layer | 4096 | Relu | Sigmoid |
| Dropout | Prob(.2) | None | None |
| Dense Layer | 2048 | Relu | Tanh |
| Dropout | Prob(.2) | None | None |
| Dense Output | Regression(1) - Classification(100) | Sigmoid | None |

Table 3: Classification and Regression MLP Model

## 3.3 Refinement

At the beginning of the project, one of the biggest challenges was to define the preprocessing stage. Doing exploratory visualization, shown evidence of a significant number of missing values, as was illustrated in Figure 3. A feature-relevance analysis was performed to address this issue, by removing all 20% less relevant features. Without having significant improves, in comparison with the standard dataset. In the second stage, finding the right parameters for the MLP was essential to get better results. Changing the learning rate, to an exponential decay was the most relevant change. With higher learning rates the model could not improve, as it is shown in Figure 6.

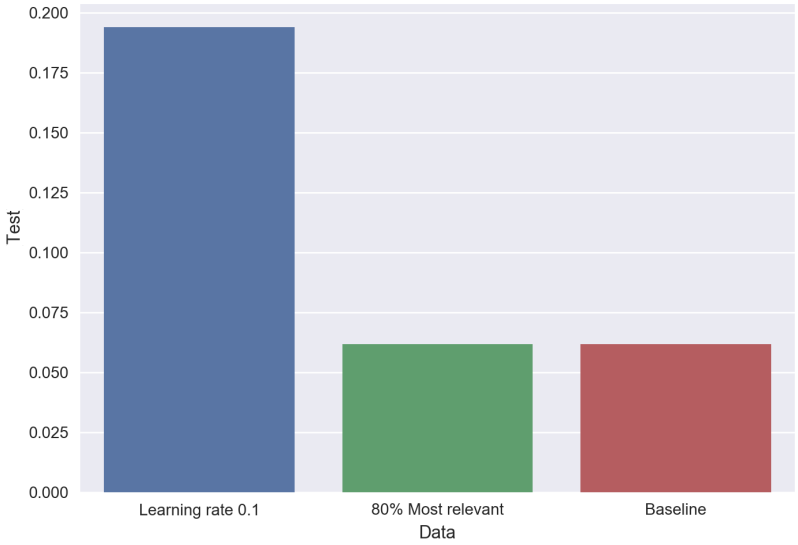


Figure 6: MAE on the Classification Model using Label Encoding. The Baseline and 80% Most Relevant models use exponential decay and the start learning rate of 0.01.

# 4 Results

## 4.1 Model Evaluation and Validation

In Table 3 was shown, the model architecture it was defined with three dense layers and a final dense layer without activation function, for the regression task. In the classification model, a dense layer with sigmoid activation it was used.

The next hyperparameters there were used because they performed the best.

- Start learning rate of 0.01 and an exponential decay of rate 0.96 with decay steps of 1,000.
- Discretization size was of 100 classes.
- The number of epochs was 100.
- The loss function was the mean squared error.
- The Optimization function was Adam.
- The batch size was 64.

A test set it was used, to verify the robustness of the results. And it has a close performance compared to the train set.

## 4.2 Justification

In comparison with the benchmark, both models have a better performance. And the classification model even had a better performance than the regression model, as it can be seen in Figure 7.

The final solution based on classification it is a non-traditional way of solving this problem. Having a better performance than an ensemble method it is interesting, and also how reducing the precision of the target function could result in a better overall performance.

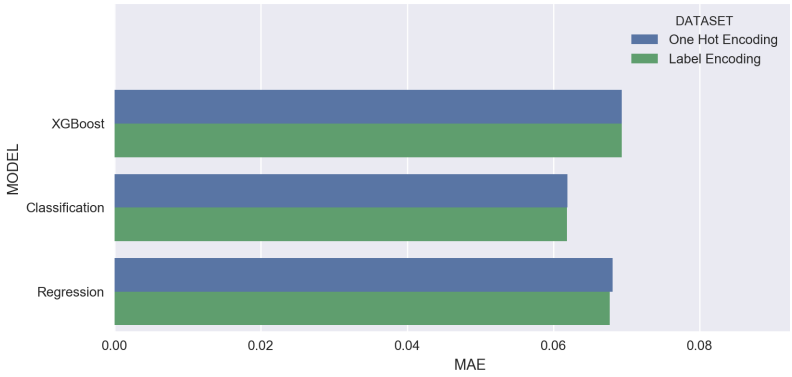


Figure 7: Comparison of the three models using Label and One Hot Encoding.

# 5 Conclusion

## 5.1 Free-Form Visualization

Figures of regression and classification tasks using LabelEncoder:

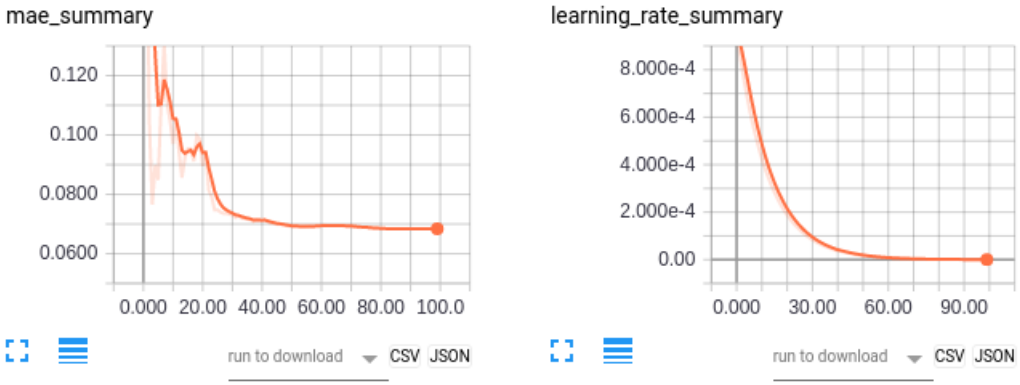


Figure 8: Visualization of MAE and the Learning Rate in Regression using TensorBoard.

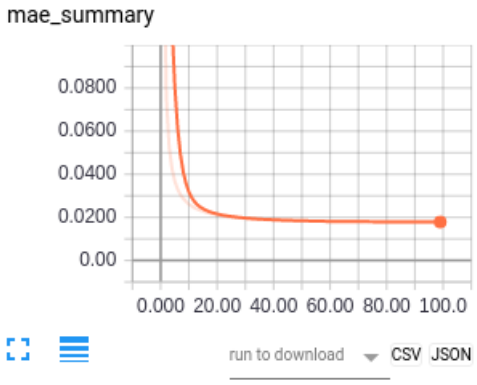


Figure 9: Visualization of MAE in Classification using TensorBoard.

## 5.2 Reflection

The process used for this project is summarized the next steps:

1. The data was downloaded.
2. The preprocessing method was performed.
3. The benchmark model using the preprocessed data was conducted.
4. A regression and classification models there were designed.
5. A TensorFlow application using the both models was coded.
6. Both models were tested and tuned.

In terms of complexity, selecting the right hyperparameters was the most difficult task, and defining a good policy for data preprocessing it was also complex.

The most impressive point was that a classification task could work better for regression than a regression model itself.

## 5.3 Improvement

To reduce the mean absolute error, it is possible to:

1. Increase the number of epochs.
2. Increase the number of classes for discretization.
3. Using CNN model transforming the data into an image format and design a deeper network.

Considering that the training and validation MAE there were mostly equaled, there is space for better results.

# References


[1] Zillow, “Zestimate.” [Online]. Available: https://www.zillow.com/zestimate/

[2] I. Zillow, “Real estate, apartments, mortgages & home values.” [Online]. Available: https://www.zillow.com/

[3] “Zillow prize: Zillow’s home value prediction (zestimate) | kaggle.” [Online]. Available: https://www.kaggle.com/c/zillow-prize-1

[4] M. Bober-Irizar, “Simple xgboost.” [Online]. Available: https://www.kaggle.com/anokas/simple-xgboost-starter-0-0655/code

[5] [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html

[6] [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html