MLND Capstone Proposal

Identifying Property Bargains

Yavor Paunov

December 30, 2017

1 Domain Background

Machine learning has already been used in the real-estate business, on platforms where buyers and sellers meet. Services such as Zillow in the United States and Booli in Sweden currently show the estimated value of the houses listed on their platforms. Having this point of reference is valuable for both buyer and seller. As a seller it's important to set a price that is not too low, while as a buyer the goal is to avoid overpaying. This in turn, leads to a more efficient housing market. This problem is particularly interesting to me as someone planning to shop for a home in the not too distant future.

For this project, I will focus on helping a potential house buyer find a bargain using ensemble learning. This technique combines the output of multiple models to obtain a more accurate prediction than any of them individually.

2 Problem Statement

The focus will be on helping a potential buyer pay as little as possible for their new house. To that end I will develop a command-line application which gives a prediction for the final sale price of a given house.

This will help a potential user of the service identify houses where the asking price is low compared to the predicted price, and in the end buy a cheaper home. Due to the available data, reliable predictions will only be available for houses in King County, Washington. Another limitation is time — the data limits us to a specific time period. However, given another similarly structured dataset, it would be fairly straightforward to obtain predictions for a different location and time.

3 Datasets and Inputs

The "House sales in King County" dataset will be used for the project. The dataset contains data for house sales in the King County area in Washington for a period of about one year starting in April 2014. The following features are present in the dataset:

id the unique id of the sale

date the date when the house was sold.

price the final selling price in US dollars.

bedrooms the number of bedrooms.

bathrooms the number of bathrooms.

sqft_living the living area in the house in square feet.

sqft_lot the area of the whole lot on which the house is built in square feet.

floors the number of floors in the house.

waterfront whether the house is built on a waterfront (1 or 0).

view the quality of the view from the house, integer values ranging from 0 to 4.

condition the physical condition of the house, integer value ranging from 1 to 5.

grade the overall grade of the house in terms of construction and design, integer values ranging from 1 to 13.

sqft_above the size of the living area above ground (living area minus basement), in square feet.

sqft_basement the size of the basement, in square feet.

yr_built the year the house was built.

yr_renovated the year the house was renovated.

zipcode the zipcode of the area where the house is located.

lat the geographical latitude of the house.

long the longitude longitude of the house.

sqft_living15 the average house square footage of the 15 closest houses.

sqft_lot15 the average lot square footage of the 15 closest houses.

4 Solution Statement

Ensemble methods will be the focus in my approach towards the solution. More specifically, the various implementations of bagging and boosting in **scikit-learn** will be explored. The final model will be decided after experimentation, and taking into consideration the data, as well as where we stand in regards to the bias-variance trade off when using a weak learner. For example, if we consistently get a model with high bias and low variance with a weak learner, perhaps switching to boosting would be appropriate.

5 Benchmark Model

A linear regression model trained on the same data as the actual predictor will be used as benchmark. A well tuned non-linear predictor should be able to outperform this basic predictor. The metric used to measure the performance of the benchmark model will be the same as the one for the solution: a coefficient of determination.

6 Evaluation Metrics

The metric used for this will R^2 — the coefficient of determination between the actual prices and those predicted by the model. The formula for the metric as implemented in **scikit-learn** is

$$R^2(y, \widehat{y}) = 1 - \frac{\Sigma_{i=0}^{n_{samples}-1}(y_i - \widehat{y}_i)^2}{\Sigma_{i=0}^{n_{samples}-1}(y_i - \overline{y})^2} \tag{1}$$

where $\overline{y} = \frac{1}{n_{samples}}\Sigma_{i=0}^{n_{samples}-i} y_i$.

The highest possible value and best score is 1.0, indicating predictions identical to the actual values. The score can be negatives as well, in case the model is sufficiently bad.

7 Project Design

7.1 Tech stack

The project will be built using **Python 3.6**, and will make use of **scikit-learn** for training and evaluating the model, as well as obtaining predictions, while **numpy** and **pandas** will be used for reading and preparing the data.

7.2 Process

The first step will be exploring the data and investigating how it needs to be preprocessed. For example, the zipcode feature might be useful since it could give an indication on the value of the house. However, it is a categorical variable with many possible values. Therefore, it needs to be encoded in a way that ensures dimensionality doesn't explode. At this stage we can identify what features will end up being used for training the model.

The next step is to train the linear model which will be used as a benchmark. This will be used as the baseline.

Next, a few different options of ensemble models will be compared and the best performing one chosen for tuning. The selected model will then be fine tuned using grid search to ensure we get score that is as high as possible on the test data.

7.3 Implementation

The final result of the project will be two command line scripts, one for training and one for predictions.

The training script's input will be a **csv** file containing data about already completed house sales. In our case, this will be the "House Sales in King County" dataset. The data will then be prepared and fed to the previously decided ensemble regressor's **fit** method, with the resulting method being persisted on disk.

The prediction script will take data about a single house as input, read the persisted model, and output a predicted price.