Paris Housing and Swimming Pools

Tyler Cambron, [tcambron@bellarmine.edu](mailto:tcambron@bellarmine.edu)

Oct. 20, 2022

I chose to use a dataset containing housing information throughout the city of Paris. The housing information contains many relative attributes that can affect whether or not a house has a swimming pool. Between the different machine learning regression models, the best model I could choose for this dataset is decision tree classification.

This dataset holds important aspects of housing. The square meters of the house is known as an integer, and could be useful to find out how much space there is for a potential pool. There are integers for number of rooms, floors, city code (zip code), the range from the city (the higher the range, the more exclusive the neighborhood), the year made, the number of previous owners, and the number of guest rooms. The square footage of the basement, attic, and garage in meters. There are boolean values for having a yard, pool, storm protector, and storage room. The price of the house is a float value. There is a categorical value for whether a house is “luxury” or “basic”.

The machine learning models I tested to find the best fit were logistic regression, support vector classification, decision tree classification, and random forest classification. The boolean value for having a pool is being predicted by these models. Logistic regression is predicting the boolean value using the logistic equation: *log(p/1-p)*. Support vector classification is predicting using linear or non-linear vectors formed by the observed data. Decision tree classification uses a tree structure by determining classification using different nodes that split into two differently classified children; it continues until properly classified. Random forest classification creates many different randomly bagged trees, and each tree classifies through the decision nodes into the classified nodes. The decision tree model is the best model for this data.

This dataset of Paris housing has 18 columns and 10,000 entries.

| **Variable Name** | **Data Type** |
| --- | --- |
| squareMeters | Integer number of square meters. |
| numberOfRooms | Integer number of rooms in the building. |
| hasYard | Integer (0, 1) if it has a yard. |
| hasPool | Integer (0, 1) if it has a pool. |
| floors | Integer number of floors in the building. |
| cityCode | Integer zip code. |
| cityPartRange | Integer number of the range from the city. The higher the more exclusive. |
| numPrevOwners | Integer number of previous owners. |
| made | Integer year made. |
| isNewBuilt | Integer (0, 1) if it is newly built. |
| hasStormProtector | Integer (0, 1) if it has a storm protector. |
| basement | Integer number of basement. |
| attic | Integer number of square meters in the attic. |
| garage | Integer number of square meters in the garage. |
| hasStorageRoom | Integer (0, 1) if it has a storage room. |
| hasGuestRoom | Integer (0, 1) if it has a guest room. |
| price | Float price of house. |
| category | Object “Luxury” or “Basic” |

Variable “category” needs to be replaced with dummies, and so it now appears as integers 0 or 1 in columns “category\_Basic” and “category\_Luxury”. Afterwards I used a standard scaler to scale the data, especially for the column “price”. Once my values were prepared, I was able to split my data into “X” and “y”. “X” is all values except “hasPool”, and “y” is the values of “hasPool”.

I start by splitting “X” and “y” into each of its own training model and testing model. The testing model size is 20% (0.2) of the original, and the training model is 80% (0.8) in return.

I used Google Colab to write the python notebook, and the libraries used by python are Pandas 0.18.1, Matplotlib 1.5.3, and SKLearn 0.16.1. The pandas was used to read and handle the csv file in which I imported the housing data from. Matplotlib was used to show the ROC curve on a graph. SKLearn provided all of the machine learning and data handling I needed such as “train\_test\_split”, “StandardScaler”, and all of the machine learning models tested.

The results of a root mean square error in each of the models are:

| **Model** | **RMSE** |
| --- | --- |
| Logistic Regression | 0.6186 |
| Support Vector Regression | 0.6148 |
| Decision Tree | 0.6099 |
| Random Forest | 0.6164 |

The results show that the lowest root mean squared error (RMSE) is from decision tree classification. The lowest RMSE is the best option for a model because it is the closest to predict the value with the least amount of error. The worst model is logistic regression as the RMSE is the highest, making the model predict with the most amount of error.

References

*scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation*. (n.d.).