NCG613 – Assignment 1

Machine Learning (KNN) for House Price Prediction

Rahul Jadhav (17250785)
Date: 11 April 2018

Contents

1. Introduction

Wikipedia defines **Machine Learning** (referred as ML going forward) as field of computer science that uses statistical techniques to give computer systems the ability to learn with data, without being explicitly programmed.

Using '**sklearn'**, one of the most popular ML libraries in python, we are interested in **predicting house prices** in London based on the location (easting/northing) and the area of house (in square meters).

We are building a model using K Nearest Neighbor (referred as **KNN** hereafter) algorithm on the dataset containing details of 1405 houses. We are going to use predictions from this model to produce 3D plots of house prices for specific floor areas. This writing is all about outlining the techniques we have followed for prediction and describing the results we get from 3D plots.

2. Techniques Used

Rescaling data

In case of data where scale of a predictor is misleading with respect to scale of other predictor, we should rescale/normalize the data. In our example, easting/northing are in range of 100 thousand while floor area ranges from 25 to 269. This is done to avoid effect of easting/northing on floor area when we calculate distance for KNN algorithm (which is explained in the document later). We are using method of **z-scores** for rescaling in which for a predictor value we subtract mean from it and divide the difference by standard deviation to get the z-score.

Choosing algorithm

ML problems are basically categorized as supervised, unsupervised and semi-supervised. In case of supervised problems, from the data provided, we know what is to be predicted (i.e. labels). This is not the case with unsupervised problems where algorithm generates classes (by grouping statistically close observations) based on given data. In 'House Price Prediction' problem, we already know what is to be predicted (i.e. house price), thus this is a **supervised ML** problem.

We know that supervised learning got 2 flavors: classification (for data with labels as classes i.e. discrete labels) and regression (for data with continuous labels). In our case, house price is a continuous label and hence we must go for **regression**.

Now, as we have decided to go for regression, we need to decide the algorithm. Multiple Linear Regression (MLR) or KNN. MLR is a parametric approach which assumes a strong linear relationship between predictors and response. However, KNN is non-parametric approach

which doesn't assume a linear relationship as in MLR, being more flexible approach. While setting asking price for a house, sellers usually tend to have a look at prices of same sized houses on sale in the vicinity oh house being sold. Based on these facts, we have decided to go for **KNN** as we are not sure if linear relationship exists between predictors and response.

KNN & it's tuning parameters

The main idea behind nearest neighbor method is to find **predefined number (K)** of observations closest in distance to new observation and predict the label from those K observations. **Distance** comes in 2 forms as Euclidian or City Block. Euclidian distance between 2 points is the straight-line distance while City Block distance is sum of distance travelled in x direction and y direction (plus distances travelled in each of other directions, in case of dimensions > 2). We also need to choose **distance averaging method** as either uniform weighted mean or distance weighted mean.

Here, K, Distance Metric and Averaging Method used are called as **tuning parameters** for KNN.

For supervised ML, the goal is to find some function f(X) which gets us closest to the actual response (y). This function depends on data on which it is trained (training data) and tuning parameters. Tweaking tuning parameters alter how close f(X) goes to y. The aim is to identify combination of these tuning parameters which gets us closest.

The general method of evaluating this is called Cross-Validation (CV) which uses train and test datasets formed by splitting available data. The scoring function measures how close to actual response our f(x) gets. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are commonly used scoring functions. We have used **MAE for CV** with grid search in our example which carries out a full cross-validation search, rather than using a pre-defined set of tuning parameters. We are also able to see **best estimators** (as below) chosen by our model for prediction.

Summary without pipeline:
Nearest Neighbors: 13
Minkowski P: 1
Weighting: distance
MAE Score: 26.49

Summary using pipeline:
Nearest Neighbors: 13
Minkowski P: 1
Weighting: distance
MAE Score: 26.47

ML Pipelines

Entire process starting from rescaling raw data to make it usable by ML algorithm, training ML algorithm, and finally using the output from algorithm for predictions is called as ML pipeline.

3. What do results say?

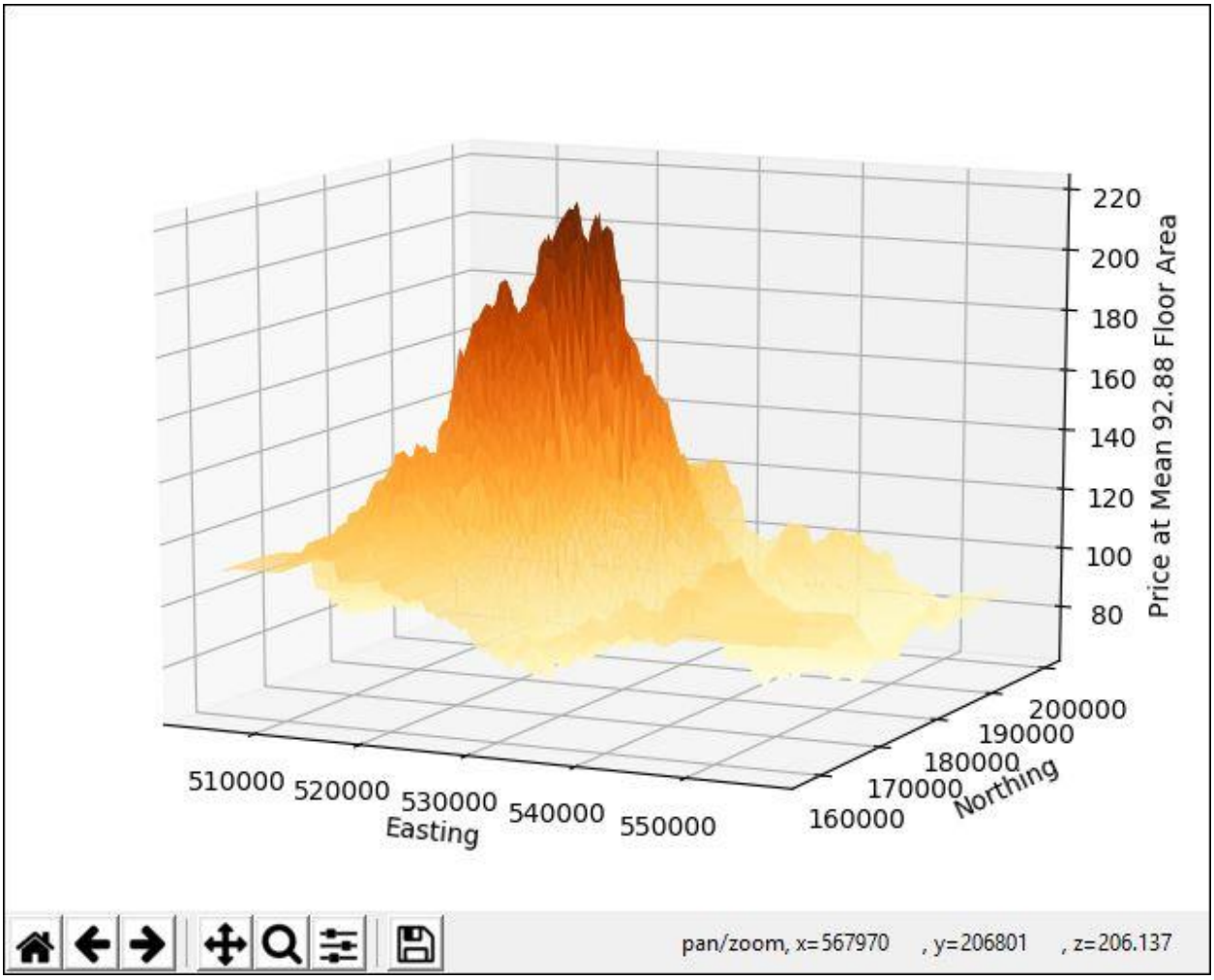
We have used model trained using Cross Validated KNN algorithm to plot & predict house prices for average floor area, for area of 75 sqm and for area of 125 sqm.

To visualize and interpret following plots precisely, we need to adjust the **azimuth and elevation** appropriately. Otherwise, it becomes difficult to interpret only based on what we see with default settings. For example, the peak in plot ideally should have maximum price, but it won't be the case if we look at plot from different angles and altitudes. Plotting in python allow us to change the view to visualize plot most appropriately.

Plots drawn below show house prices for locations in London with mentioned floor area in square meters. Those area are 92.88 sqm (average floor size), 75 sqm and 125 sqm.

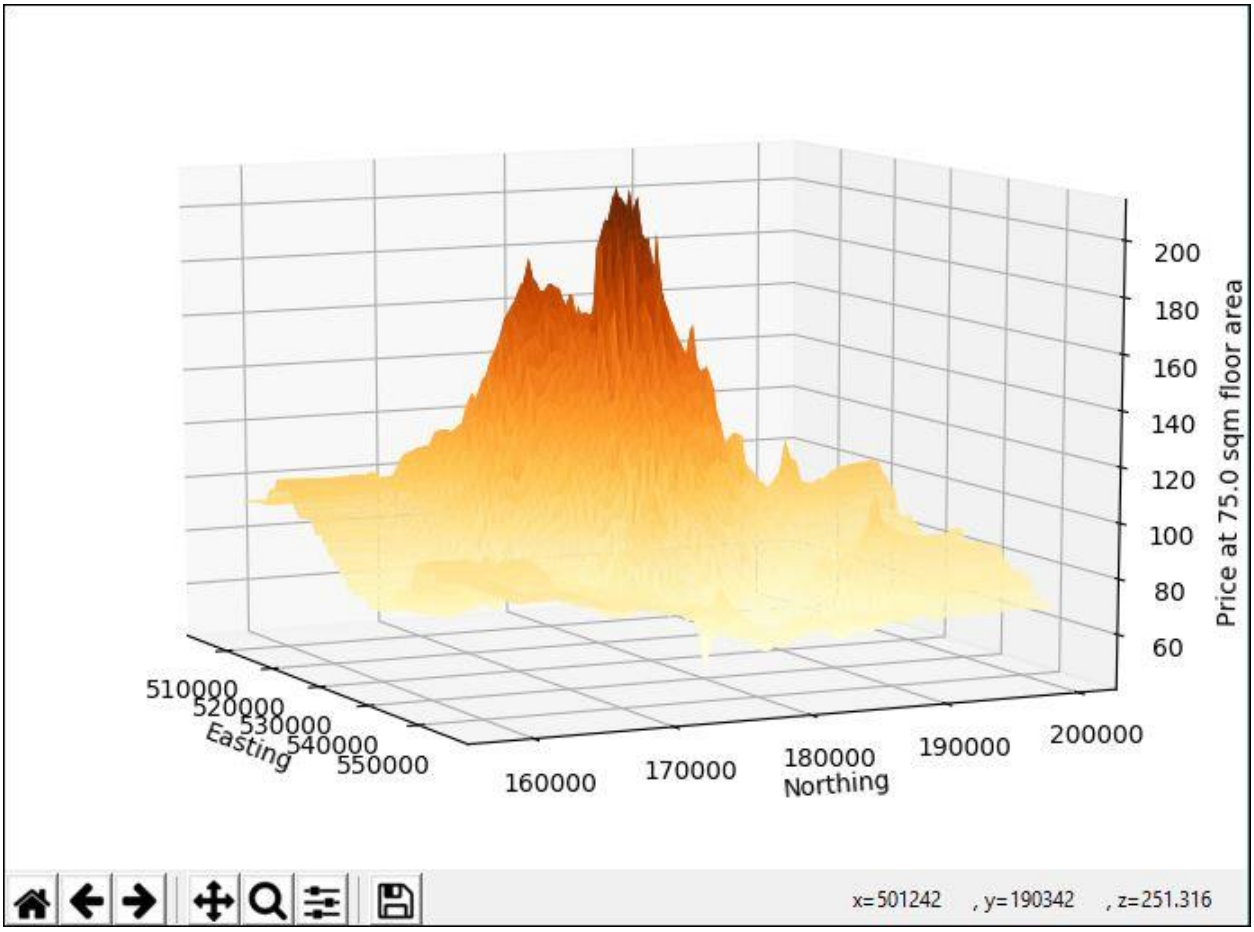
We notice that higher prices are associated with **dark colored peaks** in the 3D plots. Most important general trend is that, house prices tend to be **higher in central parts** of London, which is quite normal in case of most of the cities around the globe.

House price for average (92.88 sqm) floor area



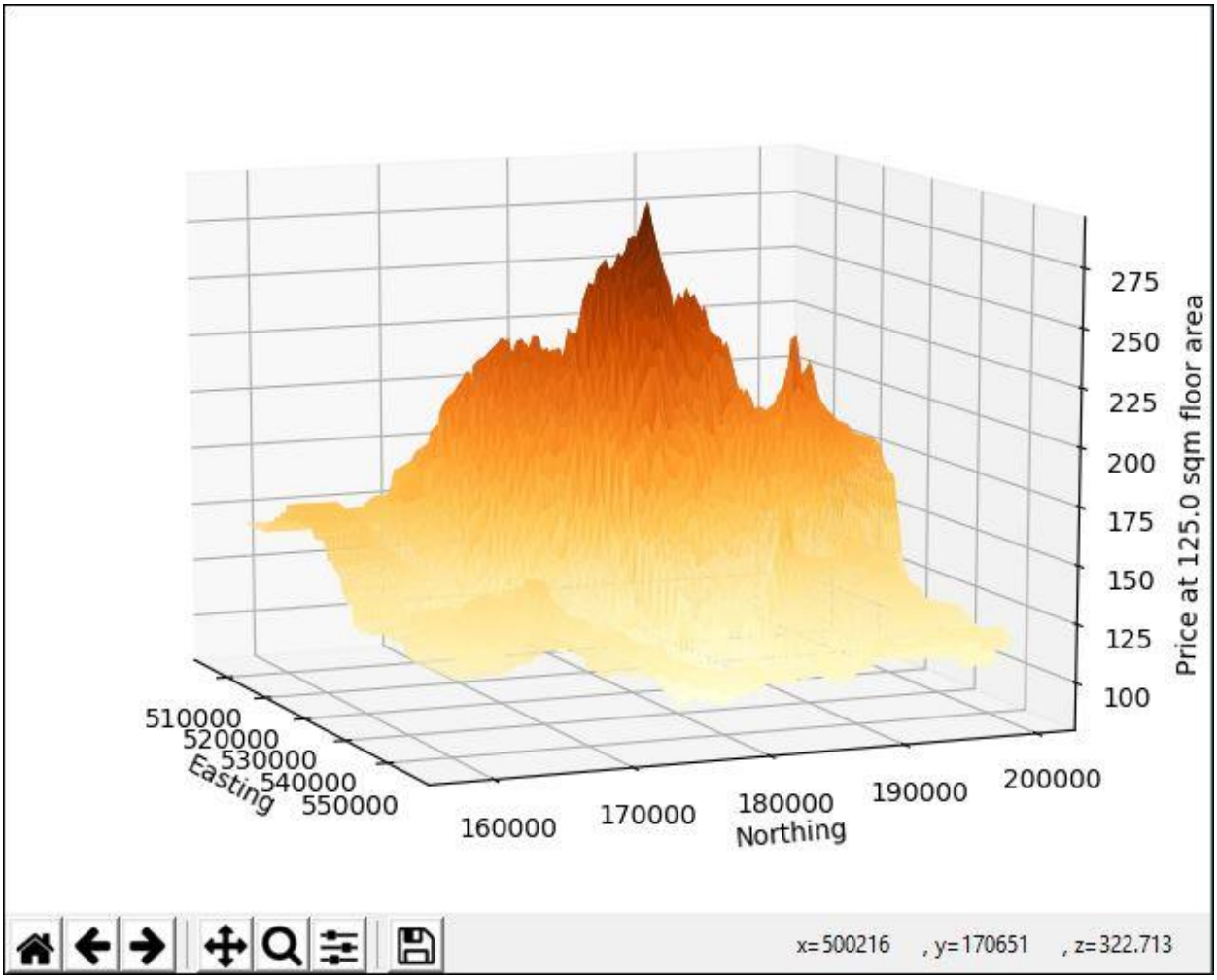
From our data, average floor area for house is 92.88 square meters. Considering 92.88 square meters as an area, above plot shows predicted house prices in thousand GBP (on z axis) for given location (easting x axis and northing on y axis). The basic trend we observe is that, prices are highest **(200K+)** for locations with easting between range 530000 to 539000 and northings greater than 155000.

House price for floor area of 75 square meters



For house floor area of 75 square meters, above plot shows predicted house prices in thousand GBP (on z axis) for given location (easting x axis and northing on y axis). We observe same trend as those for houses with average area, with the exception of few houses with easting in between 546000 to 546800 and northing between 155900 to 156150, are predicted to cost in the range of 190K to 206K.

House price for floor area of 125 square meters



For house floor area of 125 square meters, above plot shows predicted house prices in thousand GBP (on z axis) for given location (easting x axis and northing on y axis).

Overall, more number of houses (in addition to those mentioned in location ranges for 75 and mean areas) are predicted to fetch higher prices in this case. It is normal, as area in this case is 125 square meters. The house with easting as 548090 and northing as 157509 is predicted at **308K GBP** (which is not clearly visible in above setting). Thus, 3D plots are great for understanding overall trend, but we need to observe them carefully to come to the conclusion.

Last but not the least, if we add more observations to the database and rerun the model, it will learn from date as a whole and update itself accordingly without human intervention. In this case, we will get different house prices apparently even closer to actual ones. This is the beauty of machine learning. Generally, **more relevant data** we have**, more accurate** the model performs.

4. Appendix

Python code used for house price prediction explained above is attached here for reference. This code will be shared through email (if required).

hprice.py