House Price Prediction using polynomial regression & multi-layer perceptron

1. Introduction

In this report, I try to predict the house price in the capital region of Finland (i.e., Helsinki, Espoo, Vantaa, Kauniainen) using two machine learning methods: Polynomial Regression and Multi-layer Perceptron.

The machine learning problem is discussed in detail in section 2 Problem Formulation. Section 3 Methods describes the dataset, data analysis, and two machine learning models. In section 4 Results and section 5 Conclusions, I present the results and conclusions of the tests performed.

1. Problem Formulation

The question I try to resolve using machine learning is: Given the basic information about an apartment (i.e., date of buying, postal code, number of rooms), what is the approximate price of that apartment?

The datapoints in the problem are the apartments. The apartments are characterized by several traits: the time when the apartment price was estimated, the postal code, and the number of rooms. Since all these three traits can be obtained easily, they will potentially be used as features for one datapoint. The label is the price of the apartments at the corresponding time.

To be more specific about the type of data, the time when the apartment price was estimated is in “string” format, which is a quarter of a year (for example, 2010Q1, which means the first quarter of 2010). The postal code is also “string” of number (for example, 00100). The number of rooms in the original dataset is also “string” (for example, Blocks of flats, one-room flat). The price of the apartment, which is the label, is integer.

1. Methods
   1. Dataset
      1. Data Preprocessing

For the data of this project, there is a record of the average prices of old dwellings in housing companies and numbers of transactions by postal code area from the first quarter of 2010 until the last quarter of 2021. Since my interest is solely in the price of the apartments in capital region (i.e., Helsinki, Espoo, Vantaa, Kauniainen), the number of transactions and information about other postal code areas are not included in my dataset. The original data can be downloaded from the database of Statistics Finland [1].

The original dataset has 23607 rows, and 16127 out of 23607 rows do not have information about the price. It is indeed a large portion of the whole dataset but if these rows are not used, there are still around 7000 rows with adequate information about features and labels. Eventually, the cleaned dataset contains 7480 datapoints with full information, which is sufficiently enough for the ML methods used.

The next step is converting the information of the features into some types of integer value that can be used for NumPy [2] arrays and for the ML model. For the time when the apartment price was estimated, the information is transformed into an integer which indicates the number of quarters counted from the beginning of 2010 (for example, “2010Q1” will be 1, “2011Q1” will be 5). Both the postal code and the number of rooms are converted into their corresponding integers (for example, “00100” changes to 100; “Blocks of flats, one-room flat” turns into 1). More details about data processing can be found in the “Data Processing” file [3].

* + 1. Data Visualization and Exploration

Intuitively, a house price can be determined by various factors: size, location, number of bedrooms, construction year, etc. In the original and processed dataset, there are only three properties of the apartments: the time when the apartment price was estimated, the postal code, and the number of rooms. These properties can highly be potential features.

|  |  |
| --- | --- |
| Chart, scatter chart  Description automatically generated  Figure 1 – Data Scatterplot (Red: 1 room – Green: 2 rooms – Blue: 3 rooms) | Chart, treemap chart  Description automatically generated  Figure 2 – Correlation Matrix between features and label |

After visualizing the data with scatterplot (figure 1), as well as analyzing the correlation matrix (figure 2), all of three properties show some correlation with the house price. As a matter of fact, these properties are easy to obtain, so that it makes sense they are chosen as the features. Therefore, these three properties are chosen as the features for the ML model. More details about data visualization and exploration can be found in the corresponding file [4].

* 1. Polynomial Regression Model

According to the data scatterplot, one can see that the house price follows increasing curved pattern. A chosen ML model is expected to show the relationship between the features and the labels in this pattern. Polynomial regression is a simple approach, and it can effectively show this curvilinear relationship between features and label. Therefore, polynomials with order from 2 to 10 are used to train and predict the price.

The hypothesis space of a polynomial regression model is constituted by polynomial maps.

[6]

Different orders of polynomials (degree = 2, . . ., 10) are fitted to the training set by minimizing the mean squared loss [6]. The mean squared loss is calculated by examining all datapoints and obtaining the average squared difference between the predicted labels and the real labels.

.[6]

This loss function is used for the regression fitting because it is normally used in polynomial regression [6], and it is also the default option in the used scikit-learn package [7].

* 1. Multi-Layer Perceptron Model

Besides the polynomial regression model, a multi-layer perceptron (MLP) model is also explored in depth. MLP is chosen because of its ability to implicitly detect complex nonlinear relationships between dependent and independent variables. Therefore, I want to use MLP method to better deal with the complex relationships between features and label, and to see if this model can yield better results than the polynomial regression.

Being a subset of ML methods, deep learning follows the basic ML principle: find a hypothesis map out of a hypothesis space (represented by neural networks) that minimizes a chosen loss on datapoints. Neural networks are called networks because they are typically represented by composing together many different functions, and the computed values create a network-like structure. MLP is the simplest type of a neural network, where each cell (neuron) is 'connected' to all the cells from the next layer, and only the next layer uses its value [9]. An example of a basic neural network structure is shown below in figure 5 [10].

Diagram, schematic

Description automatically generated

Figure 5: A basic neural network with two-elements input layer and output layer, and a single hidden layer with 3 hidden units

Figure 5 shows the neural network representation of a predictor which maps the input (feature) vector to a predicted label (output) . This neural network defines a hypothesis space consisting of all maps . obtained from all possible choices for the weights . Simply multiplying and adding to the initial value of x would not let us represent any complex non-linear functions, so we need to introduce a non-linearity to our network. It is done via activation functions in the hidden layer [9]. There are several activation functions, and I choose the rectified linear unit (ReLU) because it is the most common one and it can be easy to train:

MLP models are fitted to the training set by minimizing the mean squared loss [6]. As described in section 3.3, the mean squared loss is calculated by examining all datapoints and obtaining the average squared difference between the predicted labels and the real labels.

.[6]

This loss function is used because it is the default loss to use for regression problems [11], and it can give straightforward results comparison with the Polynomial Regression model above.

* 1. Train-Validation-Test Set Construction

The datapoints in the dataset are randomly divided into training, validation, and test sets with the ratio of 60%, 20%, 20% respectively. The motivation of using a single split is that the dataset has a large amount of data (7480), and this method will not increase the training time or require heavy computation resource like some other methods (e.g., k-fold). Ratio 60-20-20 is used as it is a fairly common ratio in ML problems [5], and 60% of the dataset provides enough data for training the used ML methods.

1. Results

The mean squared loss calculated for training and validation data sets with different polynomial models is shown below in figures 3 and 4.

|  |  |
| --- | --- |
| A picture containing chart  Description automatically generated  Figure 3: Train-validation loss plot (degree 2 to 4) | Text  Description automatically generated  Figure 4: Train-validation loss report (degree 2 to 10) |

The MSE of the models are high (hundreds of thousands), but it is not a huge problem since the features and the labels take large values (thousands). One can observe that 4th order polynomial has lowest loss, and for higher order polynomials, the loss starts to increase. Based on the result, polynomial regression model with degree 4 is the best model, and thus being chosen as the final polynomial model. The model shows a stable performance, and no overfitting or underfitting is visible.

To justify the features choice again, other polynomial regression models with only one or two features out of three are also analyzed. The training and validation loss of these models are much higher than the final polynomial model (4th order polynomial with three features). This is another concrete reason to support the features choice at the beginning. More information about the comparisons can be found in “Polynomial Regression” file [8].

1. Code file

All the code used to produce the results in the report can be found in this GitHub repository:

<https://github.com/tamdnguyen/House-Price-Prediction>

* The raw data and the processed data can be found directly through the link.
* The data preparation related files can be found in the folder “Data Preparation”.
* The data visualization and the code of the Polynomial Regression method can be found in “Model 1\_Polynomial Regression” folder

1. References

[1] [https://pxnet2.stat.fi/PXWeb/pxweb/en/StatFin/StatFin\_\_asu\_\_ashi\_\_nj/statfin\_ashi\_pxt\_  
112p.px/](https://pxnet2.stat.fi/PXWeb/pxweb/en/StatFin/StatFin__asu__ashi__nj/statfin_ashi_pxt_112p.px/)

[2] NumPy: <https://numpy.org/>

[3] Data Processing file: <https://github.com/tamdnguyen/House-Price-Prediction/blob/main/Data%20Preparation/Data%20Processing.ipynb>

[4] Data Visualization and Exploration file: <https://github.com/tamdnguyen/House-Price-Prediction/blob/main/Model%201_Polynomial%20Regression/Data%20Visulization%20and%20Exploration.ipynb>

[5] Train-Val-Test split ratio: <https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/>

[6] Course book, chapter 3.2

[7] scikit-learn: <https://scikit-learn.org/stable/>

[8] Polynomial Regression file: <https://github.com/tamdnguyen/House-Price-Prediction/blob/main/Model%201_Polynomial%20Regression/Polynomial%20Regression.ipynb>

[9] Course Assignment 7 - Artificial neural network (ANN)

[10] Course book, chapter 3.11

[11] Loss function for regression problem: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/#:~:text=with%20sample%20code).-,Mean%20Squared%20Error%20Loss,to%20use%20for%20regression%20problems>.