HOUSE PRICE PREDICTION USING

POLYNOMIAL REGRESSION & MULTI-LAYER PERCEPTRON

1. Introduction

Buying my own house has always been my dream. Therefore, in this report, I try to apply what I learn from the course Machine Learning (ML) to predict the house price in the capital region of Finland. The house price information can be useful for real estate agents and for people who have plan to buy house in this region (i.e., application domain).

The ML problem is discussed in detail in Section 2. Section 3 has several parts: first I describe the dataset; next, I discuss about two ML models; finally, I explain the data for training-validation-testing. In section 4, the results of ML models are discussed. Finally, section 5 contains a discussion about the performance of the final ML model as well as the possible future developments.

2. Problem Formulation

The question I try to resolve using ML 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.

3. Methods

3.1. Dataset

3.1.1. 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 information about other areas is not included in the dataset. The original data can be downloaded from the database of Statistics Finland [1].

The original dataset has 23607 rows, and 16127 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 has 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 numerical 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].

3.1.2. 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.

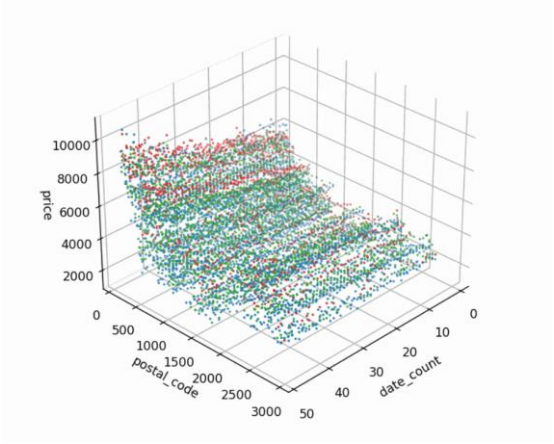

Figure 1 – Data Scatterplot
(Red: 1 room – Green: 2 rooms – Blue: 3 rooms)

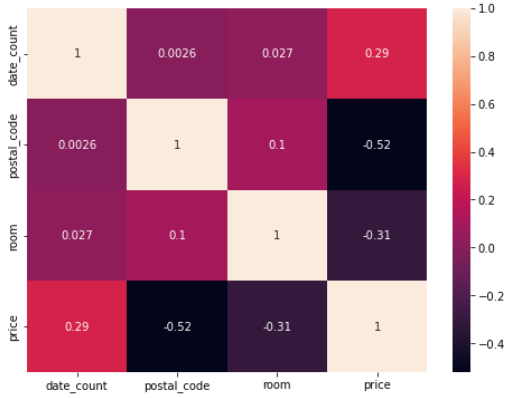

Figure 2 – Correlation Matrix between features and label

After visualizing the data with scatterplot (figure 1), as well as analyzing the correlation matrix (figure 2), all of three properties show some correlations with the house price. As a matter of fact, these properties are easy to obtain, so it makes sense if 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].

3.2.Polynomial Regression Model

According to figure 1, 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 orders from 2 to 10 are used for training and predicting the price.

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

$$\mathcal{H}_{\text{poly}}^{(n)} = \{h^{(\text{w})} : \text{R} \rightarrow \text{R} : h^{(\text{w})}(x) = \sum_{r=1}^{n} w_r \, x^{r-1} \text{ , with some } w = (w_1, \ldots, w_n)^T \in \text{R}^n \}.\text{ [5]}$$

Different orders of polynomials (degree = 2, . . ., 10) are fitted to the training set by minimizing the mean squared error (loss) (for short, MSE) [5]. The MSE is calculated by examining all datapoints and obtaining the average squared difference between the predicted labels and the real labels.

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - h^{(w)}(x^{(i)}))^2 \text{ .[5]}$$

This loss function is used for the regression fitting because it is normally used in polynomial regression [5], and it is also the default loss to use for regression problems [6].

3.3.Multi-Layer Perceptron Model

Beside 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 (i.e., features and label).

Neural networks are 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 [7]. An example of a basic neural network structure is shown below in figure 3 [8].

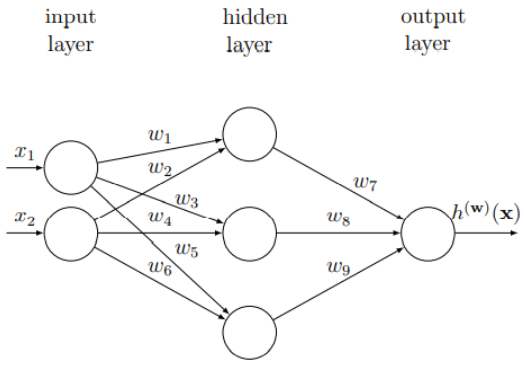

Figure 3: A basic neural network with two-elements input layer and output layer, and a single hidden layer with 3 hidden units

Figure 3 shows the neural network representation of a predictor $h^{(\mathrm{w})}(\mathrm{x})$ which maps the input (feature) vector $\mathbf{x} = (x_1, x_2)^T$ to a predicted label (output) $h^{(\mathrm{w})}(\mathrm{x})$. This neural network defines a hypothesis space consisting of all maps $h^{(\mathrm{w})}(\mathrm{x})$ obtained from all possible choices for the weights $\mathbf{w} = (w_1, \dots, w_9)^T$. To show the complex non-linear mapping, MLP utilizes activation functions at each of its calculated layers [9]. There are several activation functions, and the rectified linear unit (ReLU) is chosen because it is the most common one, and it is also computationally simple [10]:

$$ReLU(x) = \max(x, 0)$$

MLP models are fitted to the training set by minimizing the MSE. As described in section 3.3, the MSE is calculated by examining all datapoints and obtaining the average squared difference between the predicted labels and the real labels [5].

MSE is used because it is the default loss to use for regression problems [6], and it can give straightforward results comparison with the Polynomial Regression model above.

3.4.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 does not increase the training time or require heavy

computation resource like other methods (e.g., k-fold). Ratio 60-20-20 is used as it is a common ratio in ML [11], and 60% of the dataset provides enough data (4488) for training the ML models.

4. Results

The MSE calculated for training and validation data sets with different models are shown below in figures 4 and 5. 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).

	Degrees	poly_train_errors	poly_val_errors
0	2	821490.503063	846820.311068
1	3	637939.605455	648206.247382
2	4	626373.987684	635870.998087
3	5	95662280.854722	96425132.205230
4	6	274930837.219780	263704712.775417
5	7	294065621.260557	333872433.938662
6	8	1704356375.412804	1884669945.958006
7	9	1697826848.241478	1869154814.749877
8	10	2006505754.955699	2511204270.958496

Figure 4: Train-validation loss plot (degree 2 to 10)

	Layers	mlp_train_errors	mlp_val_errors
0	(60, 45, 30, 15)	633043.109809	641569.921510
1	(45, 30, 15)	630065.726971	644601.738248
2	(45, 30, 15, 15, 10, 5, 5)	626645.625087	634530.581429
3	(45, 30, 15, 15, 15, 5, 5, 5)	630583.833791	633860.409823
4	(15, 15, 15, 15, 10, 10, 10, 5, 5)	615811.643512	630782.464190
5	(15, 15, 15, 15, 10, 10, 10, 5, 5, 5)	602398.919553	609150.950916
6	(15, 15, 15, 10, 10, 10, 10, 5, 5, 5)	638207.166010	651879.265246
7	(15, 15, 15, 10, 10, 10, 5, 5, 5)	627281.120319	619793.170464
8	(15, 15, 15, 15, 5, 5, 5, 5)	617701.521773	626043.844166

Figure 5: Train-validation loss report (degree 2 to 4)

One can observe that 4th order polynomial has lowest loss (626K for training and 636K for validation). For higher order polynomials, the loss starts to increase. The 4th order polynomial model also shows a stable performance with no overfitting or underfitting.

For MLP models, the best neural network construction is 10-hidden-layer with 15 neurons per layer for layer 1 to 4, 10 neurons each layer for layer 5 to 7, and layer 8 to 10 each has 5 neurons unit (15, 15, 15, 15, 10, 10, 10, 5, 5, 5). The model has MSE of 602K for training and 609K for validation.

From the train-validation loss tables above, MLP is a little bit better than Polynomial Regression. Therefore, MLP model with 10 hidden layers (15, 15, 15, 15, 10, 10, 10, 5, 5, 5) is chosen as the final model. This model is tested with the test data, and it has a testing MSE of **606061,709.**

5. Conclusion

The best ML method for the ML problem of this report is a MLP model with 10 hidden layers (15, 15, 15, 15, 10, 10, 5, 10, 5, 5), and the final testing MSE is **606061.709**, which is quite close to the training error and a little smaller than the validation error. This consistency is a good sign of no overfitting or underfitting with the ML model. Therefore, the final test error is a great result. However, there are many improvements that can be made to achieve better results.

The dataset on Statistics Finland has only three raw features with numerous missing-data datapoints. The lack of features makes the feature selection process restricted. The relationship between the features and label being not very clear (not so correlated) also limits the model result.

Especially for the MLP model, the results can be improved by trying different learning rate values, different optimizers for the training process, maybe other activation functions like LeakyReLU. Another promising approach is GridSearchCV. Due to restricted computational resource, the idea could not be explored in this report but there could have been some interesting improvements to the MLP models as well as some better results with GridSearchCV method.

Note (Optional Reading): To justify the features choice again, other 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 (4^{th} order polynomial with three features) and MLP model. This is another concrete reason to support the features choice at the beginning. More information about the comparisons can be found in "Polynomial Regression + Multi-layer Perceptron" file [12].

6. References

[1]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/Machine%20Learning%20Models/Data%20Visulization%20and%20Exploration.ipynb

[5] Course book, chapter 3.2

[6] 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.

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

[8] Course book, chapter 3.11

[9] Activation function in MLP: https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron

[10] ReLU advantages: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

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

[12] Polynomial Regression + Multi-layer Perceptron file: https://github.com/tamdnguyen/House-Price-Prediction/blob/main/Machine%20Learning%20Models/Polynomial%20Regression%20%2B%20Multi-layer%20Perceptron.ipynb

7. 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 method can be found in "Machine Learning Models" folder