Comparing Linear Models and Instance-Based Learning for Finnish House Price Prediction

Aalto University

1 Introduction

House price/value estimation is a crucial part of selling and buying apartments. Machine learning (ML) can be used to ease the valuation process when buying or selling a house. For example, a price/value prediction can be used to avoid buying houses with unreasonably high listing prices. Vice versa, the predictions can be used to identify good deals.

In this machine learning project, we are applying regression models for the task of predicting Finnish house prices. The goal is to find out whether good predictive performance can be obtained with basic linear or instance-based methods. More precisely, we are comparing linear regression, ridge regression (regularization), and k-nearest neighbor (kNN) regressor for the task.

This project report is divided into six separate sections. Section 2 discusses the problem formulation, defining data points, features, and labels. Then, section 3 takes a more detailed look at the dataset used for training the models. It also discusses the chosen ML models and loss functions. In section 4 we consider the results of the ML models, and section 5 continues by communicating the key findings obtained during the project work. Finally, section 6 is reserved for the bibliography.

2 Problem Formulation

The house price prediction is modeled as an ML problem where **data points** represent real estate sales in Finland. Each data point is characterized by **features** describing the real estate, namely apartment area (m^2), apartment age (years), postal code (943 different postal codes), apartment floor, building condition (1=worst, 5=best), lot ownership (own or rental), and building type (flat, row house, or detached house). The quantity of interest (**label**) for a data point is the house price (€/m^2). So, the mentioned real estate properties are used to predict the real estate price/value.

3 Method

The **dataset** consists of house sales in Finland from the past 12 months. It is a unique, scraped dataset from asuntojen.hintatiedot.fi/. The scraped dataset contains ~40 000 data points. 70% of the data points (~28 000) have been used for learning a hypothesis (training) for predicting the label (house price).

The **ML models (hypothesis spaces)** used in the project are linear regression, ridge regression (regularized regression), and k-nearest neighbors regressor. Linear regression uses the following linear hypothesis space (as defined in Ch. 3 of mlbook.cs.aalto.fi)

$$\mathcal{H} = \left\{h^{(w)}: \mathbb{R}^n \to \mathbb{R}: h^{(w)}(x) = w^T x\right\}.$$

When applying regularization (ridge regression) we are shrinking the hypothesis space of the previously formulated hypothesis space of linear regression. As you will later see, the initially applied linear regression will be dramatically overfitting to the data. To cope with this problem, ridge regression shrinks the hypothesis space and is able to better generalize to unseen data. The hypothesis space of regularized regression $\widehat{\mathcal{H}}$ can be considered equivalent to a subset of the linear regression hypothesis space $\mathcal{H}$

$$\widehat{\mathcal{H}} \subset \mathcal{H}.$$

In the kNN regressor, the hypothesis space is defined by the dataset. As in all instance-based methods, we use instances (training data points) to define a hypothesis.

The **loss function** for learning the linear regression, and comparing the training and validation errors is the basic (mean) squared error loss (as formulated in Ch. 2 of mlbook.cs.aalto.fi)

$$\mathcal{L}\big((x, y), h\big) \coloneqq \big(y - h(x)\big)^2.$$

The motivation for using squared error loss is that it is generally considered a good first choice loss function for regression models (Ch. 2 of mlbook.cs.aalto.fi). It can also be easily implemented with ready-made libraries, such as scikit-learn. The loss function for training the ridge regression is a combination of mean squared error loss plus L2 regularization penalty. The loss can be formulated as follows

$$\mathcal{L}\big((x, y), h\big) + \lambda\mathcal{R}(h) \coloneqq \big(y - h(x)\big)^2 + \lambda\big||w|\big|^2.$$

Again, for the instance-based method kNN, there is no formulated loss function for learning. Instead, the training data is searched to obtain k instances with minimum Euclidean distance to given data points.

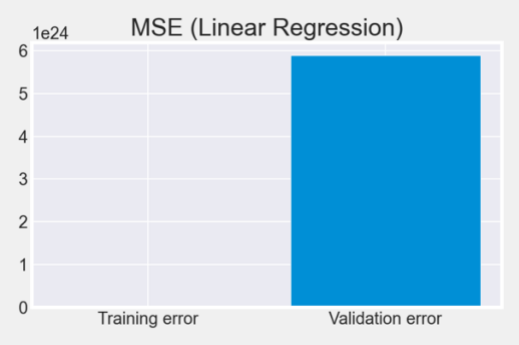
After choosing the best model based on validation error, the resulting model is evaluated on previously unseen data (test set). The coefficient of determination, R^2 (R-squared) is used as the loss function for testing the generalization. It was chosen because it is able to represent how well the chosen model fits the unseen data. It can also be easily implemented with scikit-learn.

The **data is split** into training, validation, and test sets with a single split. 70% of the data is used for training, 20% for validation, and 10% for testing. The data is shuffled randomly before doing the splits.

4 Results

After building the initial model (linear regression), we got the following results for training and validation errors.

Figure 1. Training and Validation errors of Linear Regression



As figure 1 showcases, the validation error of ≈ 5.9e24 is way larger than the training error ≈473 847 that can't be even seen on the plot. This indicates that the model is overfitting to the data. To combat this problem, the next model built was a regularized regression with L2 penalty (ridge regression). Figure 2 showcases the results obtained from the ridge regression.

Figure 2. Training and Validation errors of Ridge Regression

After tuning the hyperparameter λ, a tremendously better validation error of ≈572 000 was obtained. Now also the training and validation errors are close to each other. This suggests that regularization successfully combats the overfitting problem of the linear regression, and thus gives us a much better predictor map.

To compare the linear predictors' results to instance-based learning, the kNN regressor was tuned next. The following figure (3) communicates the results obtained from kNN.

Figure 3. Training and Validation errors of kNN Regressor

The kNN regressor obtained its best results with the number of neighbors (k) being 14. With a validation error of ≈1 640 000 the kNN performed clearly better than the overfitting linear regression. However, the validation error of ridge regression ≈572 000 is still much better compared to the best results of kNN. Thus, based on the validation error, ridge regression was chosen to be the **best model**. To measure the ridge regression's ability to generalize, R^2 measures were calculated for the training set. The obtained R^2 value is ≈0.77. This suggests that 77% of the variance in the labels was successfully explained with the features. All of the results are collected into the following table.

Table 1. All Results

MODEL	*Linear Regression*	*Ridge Regression*	*kNN Regressor*
Training MSE	473 850	485 730	1 530 000
Validation MSE	5.9e24	572 000	1 640 000
Testing R^2		0.77	

5 Conclusions

Our **initial goal** was to find out whether a good predictive performance could be obtained with simple regression models. We were also interested in comparing linear models and instance-based learning for the task. Both of these tasks were completed successfully. As ridge regression was found out to be the best model, the linear modeling approach ended up being superior to instance-based learning for the given task. The ridge regression was able to balance the training and validation errors, working as a solution to the overfitting problem of the linear regression model. Also, an R^2 value of 0.77 (on the training set) indicates that 77% of the variance in the labels was explained. This can be considered as a good result for a such simple model.

Despite obtaining good results with the ridge regression, we **do not consider the results to be optimal**. As only three models were built, it is highly unlikely that one of them would end up being an optimal hypothesis space for the given task. Thus, we see there is certainly room for improvement.

For **further improvement**, other hypothesis spaces could be modeled for the given task. For example, more advanced models like gradient boosting and neural networks would likely result in better outcomes. Also feature selection could be tried out to see if it could improve the results.

6 Bibliography

Jung, Alexander. "Machine Learning. The Basics.", 2021. https://github.com/alexjungaalto/MachineLearningTheBasics.