# Machine Learning - Real Estate CW

[1] Viktor Hatina - 40510222

[1] Edinburgh Napier University, 10 Colinton Rd, Edinburgh EH10 5DT, United Kingdom

**Abstract.**
Machine learning is a study of computer algorithms, which allow us to predict new outcomes based on historical input & output data. This is made possible through the process of training predictor variables in relation to the target variables, and developing models which accurately apply this knowledge to foreign data. This paper covers the basic and essential machine learning steps, along with other methods, that I have utilized in developing and identifying a model with the highest degree of accuracy in predicting housing price. This model is based on the Real_estate.csv dataset and my script covers: loading Real_estate.csv data, pre-processing data, splitting dataset to training and testing portions, training/fitting data, performing k-fold iteration, testing different models, identifying models and tuning parameters (hyperparameters) to achieve the highest degree of accuracy.

## 1 Introduction

Housing price can be affected by a wide range of factors, such as size, location, or age. In this project, I considered how variables like date of transaction, age, coordinate location, and distance to nearest store and metro correlate to housing price.

I utilized four regression models: Linear Regression, Lasso Regression, Ridge Regression, and Elastic Net Regression, to identify the model which most accurately predict housing price based on this set of variables.

## 2 Developing Models

### 2.1 Processing Data

It is crucial for data to be cleaned before it is sent for training. To make sure the real_estate.csv file was clean and had no missing values, I used panda's 'isna().sum' function.

Once the dataset is clean, it is important to consider the columns in the dataset, and eliminate those not relevant to the target variable. Keeping such data would increase the error between the actual and predicted output values and decrease the accuracy of the model.

I initially assumed to remove both “No” and “X1 transaction date” columns, as the row-number and date of transaction seemed to be variables which in no way reflected housing price. However, later realizing that a price is only relevant to the year it was recorded, I decided to place it back into the set of predictor variables.
While the value of the date itself posed as unnecessary data, together with the price, the date would help reflect the value of housing relevant to the specific year.

At this stage of developing my machine learning model, I realized that it is not only important to consider how the predictor value X affects the target value Y, but also the underlying insight derived from the relationship between the two variables.

As in this example, a single record of a transaction date may not be relevant in predicting the output Y, however, using it as a predictor with larger sums of data allows us to potentially identify a pattern in case during some period of time, prices for housing were unusually inflated.

### 2.2 Training and Testing

Satisfied with the variables in my dataset, I defined my X – predictor variables, and Y – target variables. Recognizing the models’ relatively average accuracy of 50-60% from previous tests, I decided to maximize the training set and split the data into 80% - training and 20% - testing samples. Despite the small testing sample, I was assured all data would eventually be tested on as our model was utilizing the ‘KFold’ method.

### 2.3 Performing K-fold cross validation

To display a clear visual of each iteration in the k-fold cross validation process, I created a loop that, together with other functions (get_score, mse, plot_model), displays: the actual vs predicted output values for both training and testing sets, the calculated average mean squared error with each fold, and the overall average scores for each individual model.

I later utilize the ‘cross_val_score’ function from the ‘sklearn.model_selection’ library to support my method’s results.

These results offer the mean square error and accuracy records for all models at default setting - absent of hyperparameter tuning. At this stage, Linear Regression is the most accurate model with a score of 58.51%.

### 2.4 Hyperparameter Tuning

At this stage I utilize the model_param_tuning function together with the GridSearchCV function to compare and identify the most accurate model and it is best performing hyper-parameters.

I chose numpy’s logspace linear range to iterate over potential alpha parameters and selected -10 as the starting value (despite requiring a positive alpha), as a means of identifying the ideal regularization strength for the model. The other parameter I

involved in this test was the normalization setting, which in all models turned out to perform better when set to 'True'.

# 3 Results

After performing k-fold cross validation with the optimal parameters for each model, these were the results:

| Model | Best Score | Best Hyper-parameter(s) |
|---|---|---|
| **Linear Regression** | 58.55% | 'normalize'=True |
| **Lasso Regression** | 58.63% | 'alpha'=0.000225, 'normalize'=True |
| **Ridge Regression** | 58.78% | 'alpha'=0.035938, 'normalize'=True |
| **Elastic Net Regression** | 58.71% | 'alpha'=0.000129, 'normalize'=True |

After reviewing the results, Ridge regression performed the best with a degree of accuracy 58.78% - only 0.07% from Elastic Net scoring 58.71%.

## 3.1 Conclusion

While It is hard to define the underlying reason for the best performing model with such miniscule variation in the accuracy results, I believe that Ridge Regression may have had highest accuracy as it is employs l2 regularization and has the ability to reduce model complexity by shrinking the effect of variables, without removing them. In this case, shrinking the coefficients to the predictor variables which aren't contributing to the prediction task (potential outliers), effectively reduces their impact and maintains a higher degree of accuracy for this model.

# References

1. scikit-learn. 2021. API Reference. [online] Available at: <https://scikit-learn.org/stable/modules/classes.html?highlight=sklearn%20linear_model#module-sklearn.linear_model> [Accessed 12 November 2021]