CSI 5387 Data Mining and Concept Learning

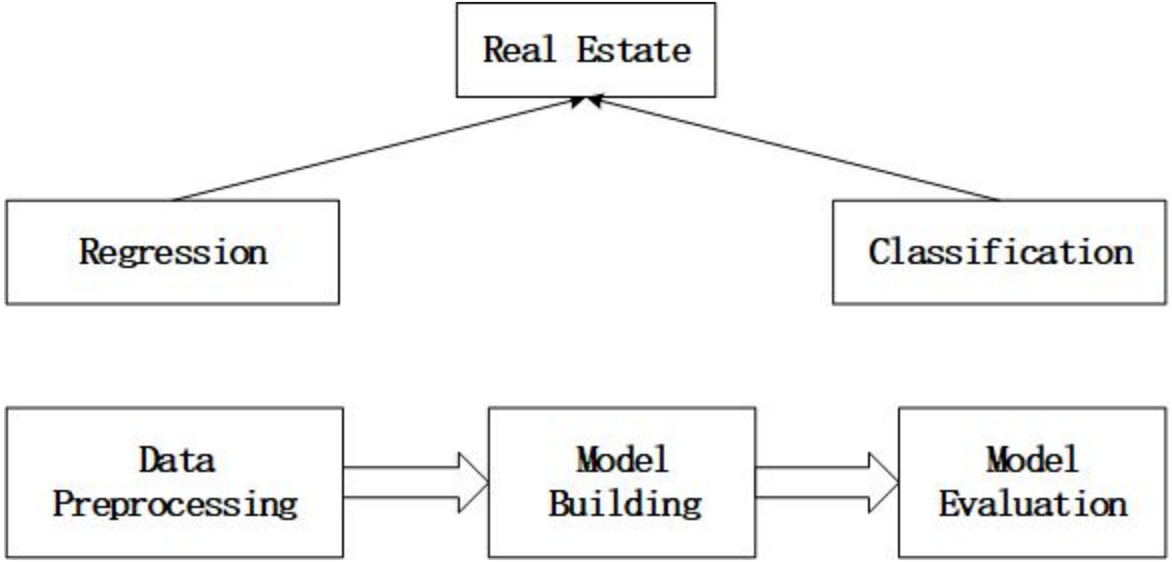
Project

Group 6 Real Estate

Presented by:
Ao Peng
Lingfeng Zhang
Ning Wang
Zhihe Huang

GitHub Link: https://github.com/AaPaul/CSI-5387_Data_Mining

Introduction



Outline

- Introduction+dataset description
- Data Pre-processing
- Outlier Analysis
- Conventional Regression+Evaluation
- Neural Network(MLP) in regression
- Classification(bins)+Evaluation

Introduction

Dataset: Real Estate

X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

Data Pre-processing

- Missing values detection: no missing values in this dataset

```
X1 transaction date
False    414
Name: X1 transaction date, dtype: int64

X2 house age
False    414
Name: X2 house age, dtype: int64

X3 distance to the nearest MRT station
False    414
Name: X3 distance to the nearest MRT station, dtype: int64

X4 number of convenience stores
False    414
Name: X4 number of convenience stores, dtype: int64

X5 latitude
False    414
Name: X5 latitude, dtype: int64

X6 longitude
False    414
Name: X6 longitude, dtype: int64

Y house price of unit area
False    414
Name: Y house price of unit area, dtype: int64
```

uOttawa

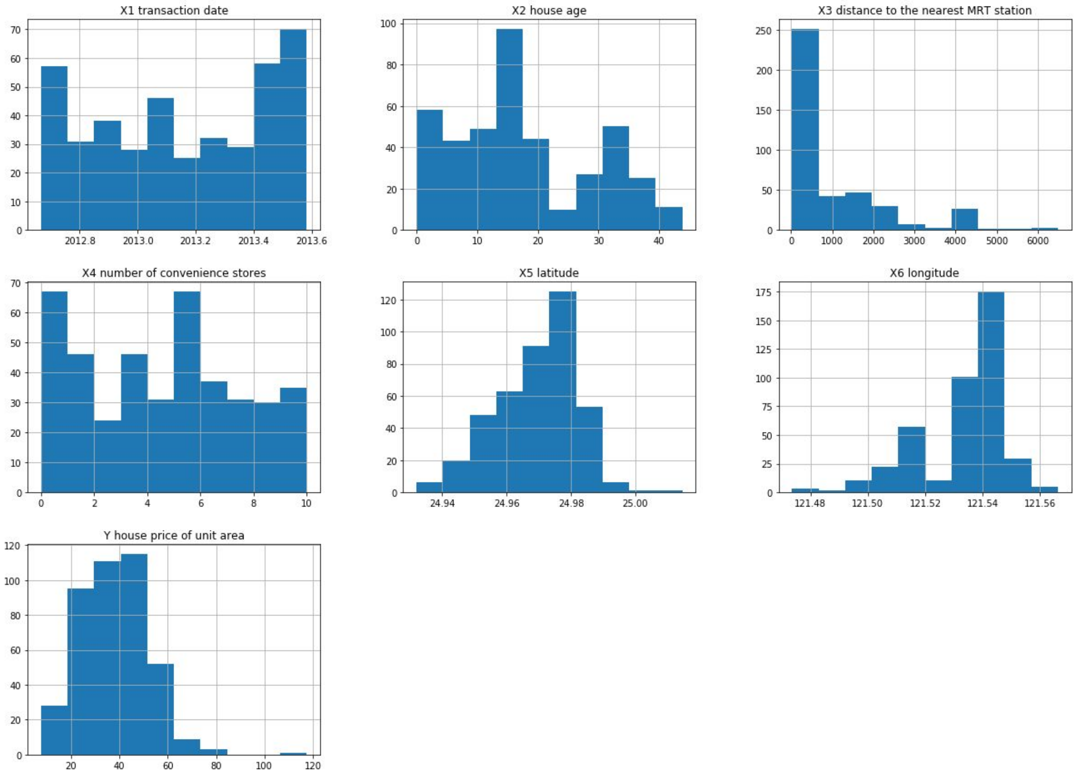
Data Pre-processing

- Brief information (statistics)

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	2013.148953	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	0.281995	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	2012.666667	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	2012.916667	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	2013.166667	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	2013.416667	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	2013.583333	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000

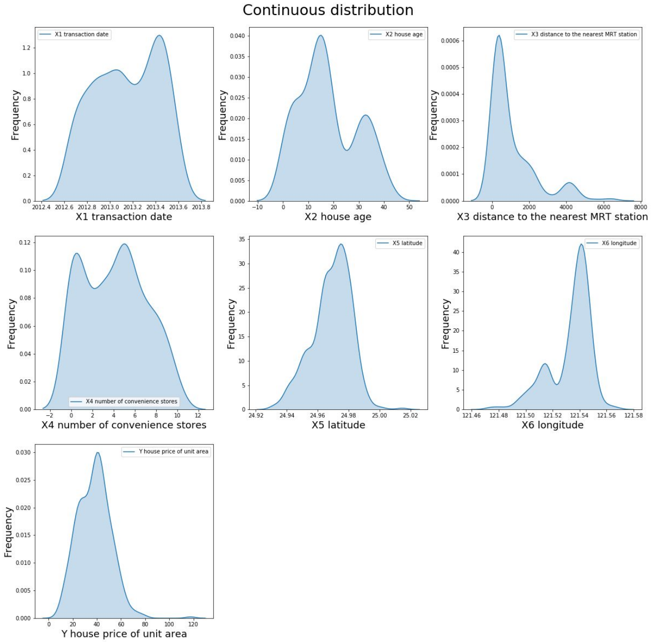
Data Exploration

Histogram: Visualize all numeric data and their distributions



Density Plot

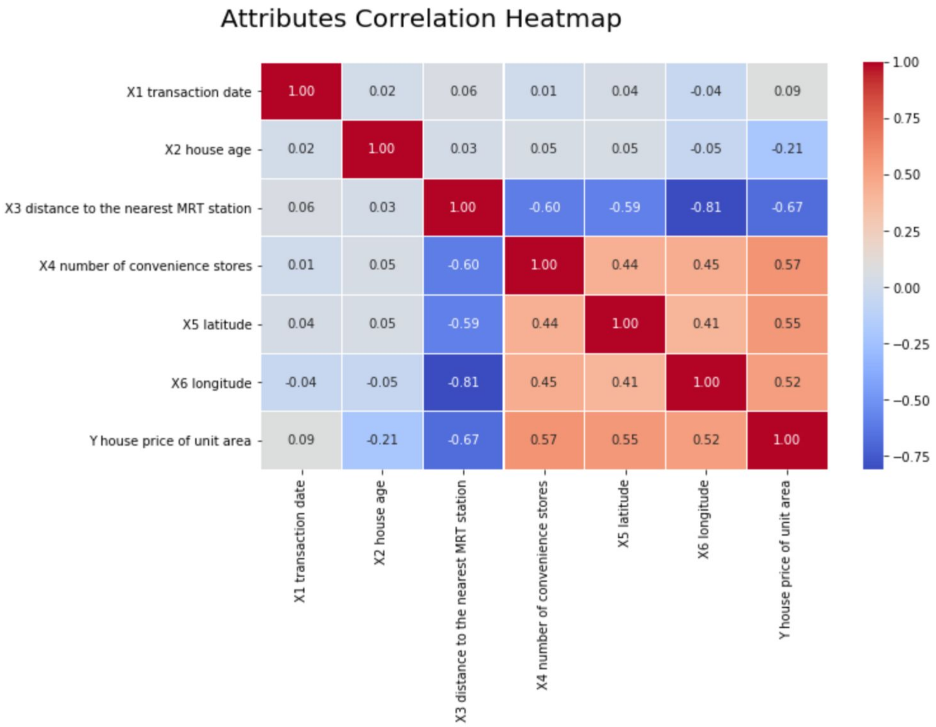
- Density plot is another method that can work well in understanding how the data is distributed for one attribute



uOttawa

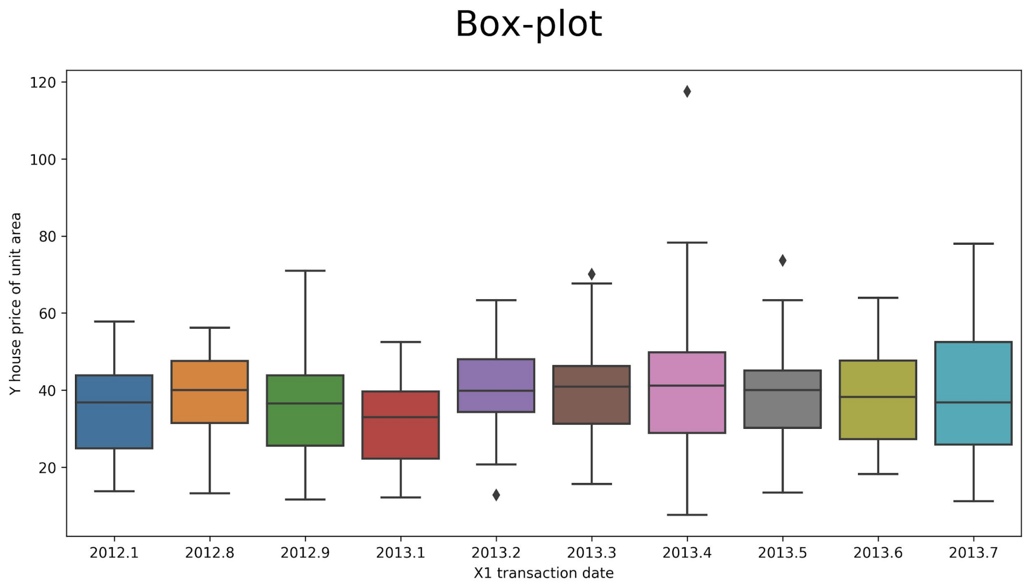
Data Exploration

3. Heatmap: shows the correlation between every two attributes



Outlier Analysis

- Box-plot
  It is a way of effectively depicting groups of numeric data and it is easy to know the quartilevalues and also potential outliers

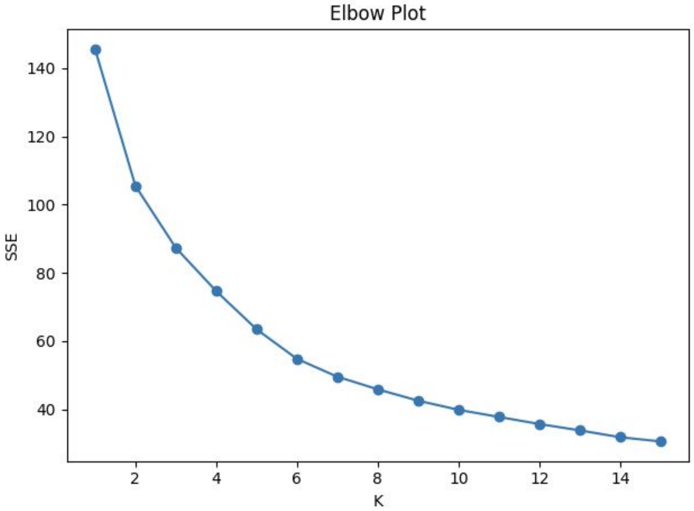


Outlier Analysis

- Elbow Method

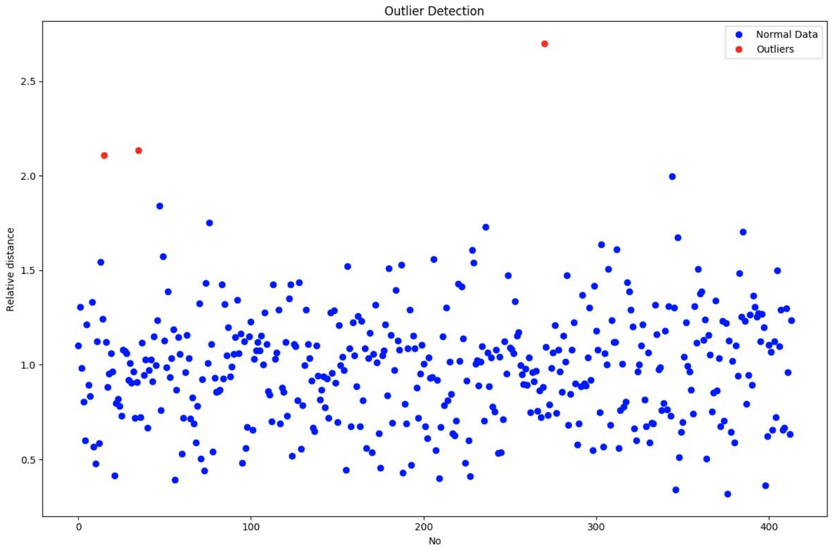
  Use min_max method to standardize the data.

  From the plot, we choose 8 as the K value.



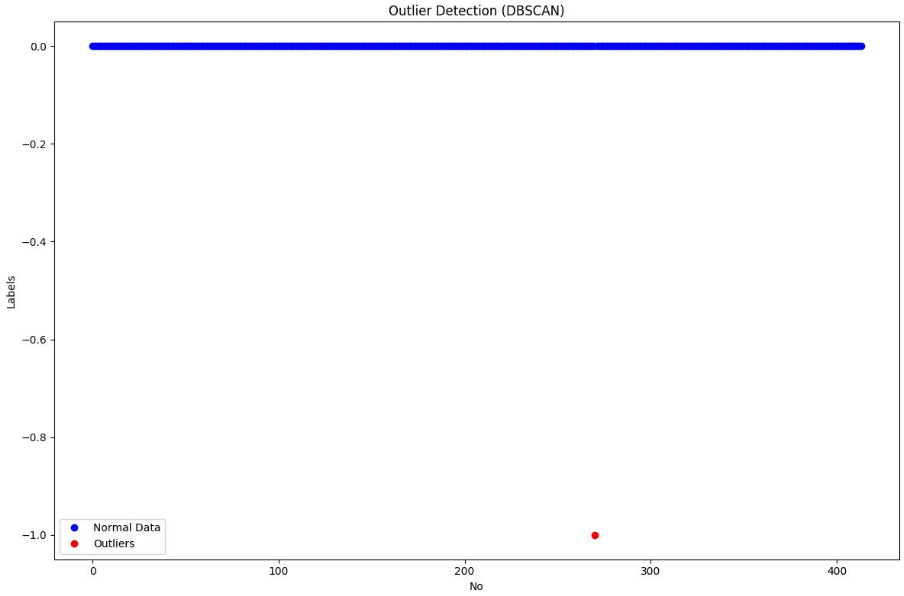
Outlier Analysis

- Set threshold=2, K=8 and do the itreation 100 times
- Create KMeans model and fit data



Outlier Analysis

- DBSCAN
  Only one point is judged as the outlier which is almost the same to the results of KMeans.

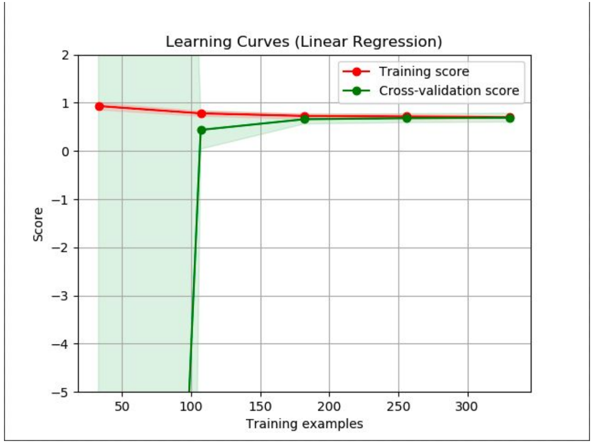


Outlier Analysis

- The number of outliers is **3** with the K-Means method, and **1** with the DBSCAN method while the total number of this dataset is **414**
- Lack of sufficient sample

Regression Models we used

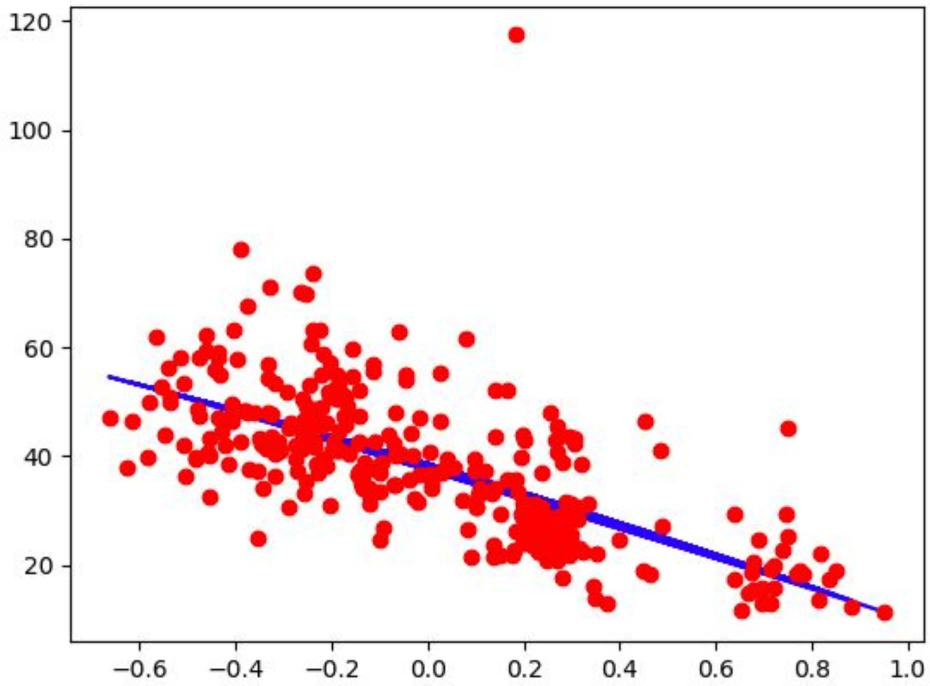
1.linear_regression_model
2.support_vector_regression()
3.KNN_regression()
4.gaussian_process_regression()
5.decision_tree_regression()
6.voting_regression()



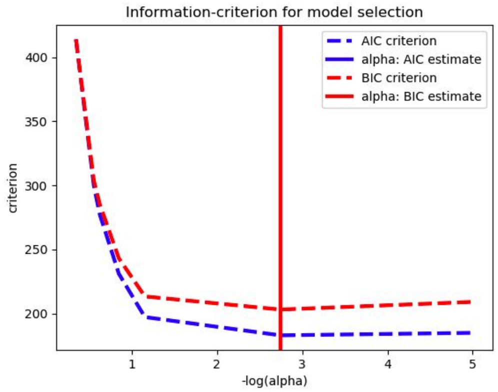
Various Linear Models:

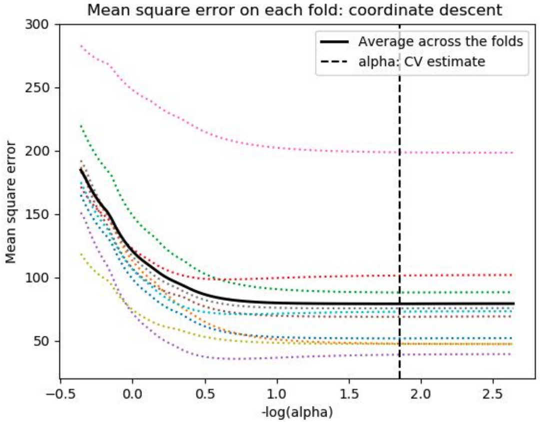
1.linear_regression_model_with_interaction()
2.Ridge_regression_model()
3.Ridge_cross_validation_model()
4.Lasso_regression()
5.Lasso_cross_validation_model()
6.Lasso_AIC()
7.Lasso_BIC()
8.Elastic_net_regression()
9.least_angle_regression()
10.Bayesian_ridgt_regression()
11.ARD_regression()
12.SGD_regression()
13.passive_aggressive_regression()
14.robust_regreesion()
15.Theil_Sen_robust_regreesion()
16.huber_robust_regreesion()
17.kernel_ridge_regression()

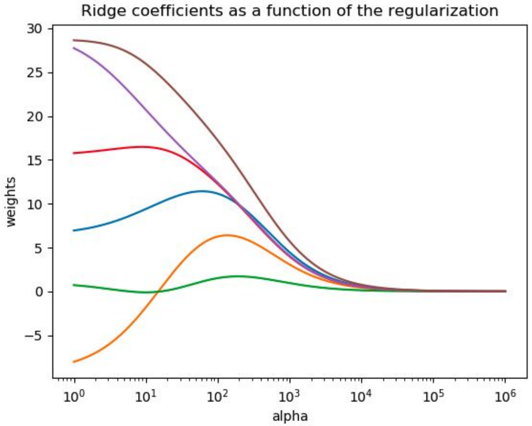
Regression Visualization with PCA

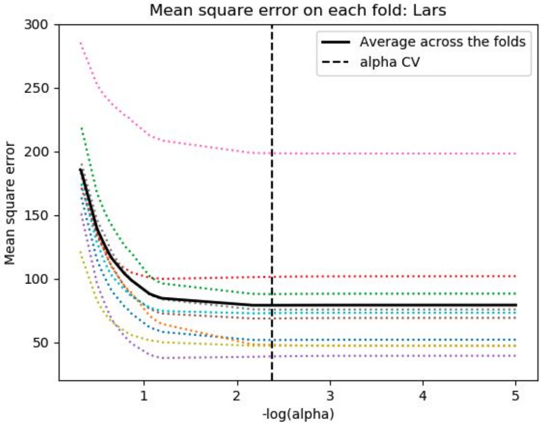


Hyper-parameter Analysis in some Linear Models







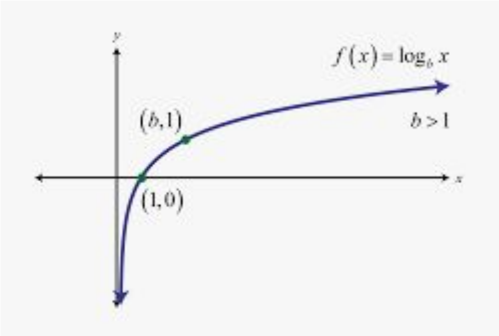


uOttawa

Regression Evaluation we used

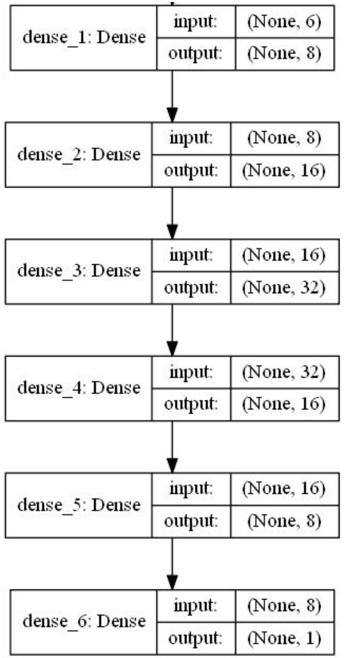
1.mean_absolute_error
2.➡️mean_squared_error
3.explained_variance_score
4.max_error
5.median_absolute_error
6.root_mean_squared_error
7.▶️R_squared
8.mean_squared_percentage_error
9.mean_absolute_percentage_error
10.mean_squared_logarithmic_error(MSLE)
11.root_mean_squared_logarithmic_error(RMSKLE)

$$\text{MSLE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2.$$



Neural Network(MLP) in regression

Overview of the network



Input layer:
units:8, activation function:relu
1st hidden layer:
units:16, activation function:relu
2nd hidden layer:
units:32, activation function:relu
3rd hidden layer:
units:16, activation function:relu
4th hidden layer:
units:8, activation function:relu
Output layer:
units:1, activation function:linear

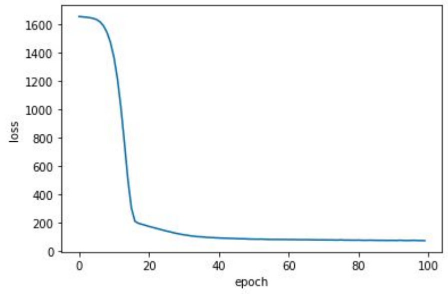
Neural Network(MLP) in regression

Backpropogation

Backpropogation is used to update the weights for each iteration.

Loss function: Mean squared error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$



Optimizer: Nadam

Hyper-parameter: epochs=300
batch_size=68
bias_used=True

uOttawa

Neural Network(MLP) in regression

Final Evaluation for testing

Mean squared logarithmic error (MLSE):

The best MLSE value is:

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(\log(\hat{y}_i+1)-\log(y_i+1))^2 = 0.052$$

Comparisions of regression

Models	MSLE
ARD Regression (Automatic Relevance Determination)	0.070
Bayesian Ridge Regression	0.069
Gaussian Process Regression	2.243
10-fold cross validation Ridge Regression	0.068
SGD Regression	0.069
Theil Sen Robustness Regression	0.197
Decision Tree Regression	0.065
10-fold cross validation Elastic Net	0.068
Huber Robustness Regression	0.064
KNN Regression (k = 3, uniform weight)	0.054
Kernel Ridge Regression	0.075
Lasso regression (Akaike information criterion)	0.070
Lasso regression (Bayes information criterion)	0.070
10-fold cross validation Lasso Regression	0.068
Lasso Regression (alpha=0.05)	0.068
Least Angle Regression (alpha=0.05)	0.068
Linear Regression (degree =3, bias=False, interception=False)	0.061
Passive Aggressive Regression	0.063
Ridge Regression (alpha=5)	0.068
Random Sample Consensus Robustness Regression	0.067
Support Vector Regression(kernel=linear)	0.066
Voting Regression	0.061
Multilayer Perceptron	0.052

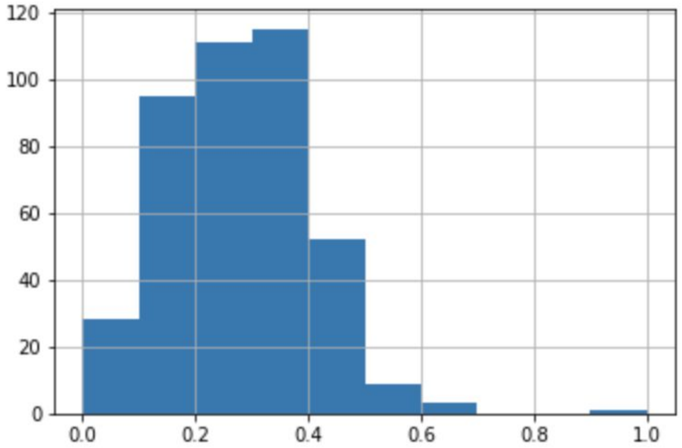
For each model, the hyper-parameters are fine-tunned.

Multilayer Perceptron is the best model in this task.

Classification

1. Min-max Standardization: All values are scaled to [0, 1].

2. Bining for Y house price per unit area:



0.0 ~ .2	low
.2 ~ .4	Medium
.4 ~ .6	Moderately High
.6 ~ 1.0	High

Binning Standard

Classification

3. Feature Selection:

1) Boruta:

['X2 house age', 'X3 distance to the nearest MRT station', 'X5 latitude', 'X6 longitude']

2) Tree-based:

['X2 house age', 'X3 distance to the nearest MRT station', 'X6 longitude']

Classification

4. Resampling due to unbalanced distribution:

1) Over_sampling: A random set of copies of minority class examples is added to the data

Class distribution of oversampling with train_set_boruta

[('High', 226), ('Low', 226), ('Medium', 226), ('Moderately High', 226)]

2) Under_sampling: Deletes some examples from the majority class

Class distribution of undersampling with boruta_set

[('High', 4), ('Low', 84), ('Medium', 123), ('Moderately High', 23)]

Classification

Now we've got 5 datasets:

Dataset	Total instance	Description
Original	414	No resampled and no feature selection
Ros_Boruta	904	ROS resampled and Boruta selection
Ros_Tr	904	ROS resampled and Tree-based selection
Renn_Boruta	234	RENN resampled and Boruta selection
Renn_Tr	219	RENN resampled and Tree-based selection

Dataset Description

ROS resampled: over-sampling
RENN resampled: under-sampling

Classification Models

1) Tree Model: DecisionTreeClassifier(decision tree algorithm)

2) Linear Model: LinearSVC(support vector machine algorithm)

3) Probabilistic model: GaussianNB(naive bayes algorithm)

4) Ensemble Model: Adaboost(boosting algorithm)

Evaluation for Classifiers: Friedman test

Friedman test: is designed for testing k algorithms against n datasets.

Evaluation Metrics: accuracy, precision, recall, **F1**, etc.

F1 score: combines precision and recall, both of which are insensitive about true negatives.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Evaluation for Classifiers: Friedman test

F1 score of models:

	DT	LSVC	GNB	ADB
Original	0.7952	0.7590	0.7108	0.8072
Ros_Boruta	0.8785	0.5525	0.7348	0.5856
Ros_Tr	0.8619	0.5304	0.6243	0.5470
Renn_Boruta	1.0000	0.8511	0.9149	0.5745
Renn_Tr	0.9545	0.8182	0.9318	0.6364

Evaluation for Classifiers: Friedman test

Ranks of F1 score:(k = 4, n = 5)

	DT	LSVC	GNB	ADB
Original	2	3	4	1
Ros_Boruta	1	4	2	3
Ros_Tr	1	4	2	3
Renn_Boruta	1	3	2	4
Renn_Tr	1	3	2	4
avg rank	1.2	3.4	2.4	3

We have $\bar{R} = \dfrac{1+k}{2} = 2.5$, $n\sum_j (R_j - \overline{R})^2 = 13.8$ and $\dfrac{1}{n(k-1)}\sum_{ij}(R_{ij} - \bar{R})^2 = 1.67$, so the Friedman statistic is 8.26 .

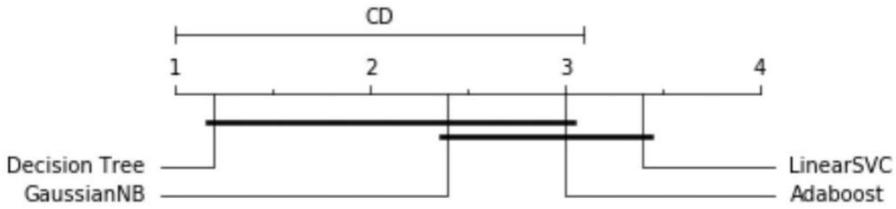
The critical value for k = 4, n = 5 at the α = 0.05 level is (α = 0.05, DOF = 3) 7.81473, so we reject the null hypothesis that all algorithms performs equally.

Evaluation for Classifiers: Nemenyi test

The critical difference in Nemenyi test is as follows:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}$$

where q_α depends on the significance level α as well as k, it is 2.569, so CD = 2.098.



CD diagram for the pairwise Nemenyi test

The performance of the top ranked algorithm "Decision Tree" is significantly better than the bottom one "LinearSVC" and slightly better than the other two.

References

1. D. (DJ) Sarkar, 2018, "The Art of Effective Visualization of Multi-dimensional Data", Retrieved from https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

2. V. Valkov, 2019, "Predicting House Prices with Linear Regression | Machine Learning from Scratch (Part II)", Retrieved from https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1

3. Flach, P., 2012, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data". Cambridge, pp.355-357.

Thanks!