Maciej Legas 2031545

CSCM35 Coursework: Real Estate Investment

Introduction

The problem given is to define and solve a data mining problem that involves a given dataset of 34,857 properties listed or sold between September 2016 to February 2018, with their prices and details. The proposed definition of a problem to solve is that we would like to be able to predict house prices of houses that we have the same details as the houses in the dataset, by basing on the prices of the houses from the given dataset. A solution of this problem can help with solving other relevant issues, such as finding houses that are priced too low and perhaps should be invested in, or finding what would be the more or less right price to list a house for. To solve this problem, we propose the use of linear regression models, alongside some data pre-processing techniques used on the dataset, to train and fit a model that will be able to give an approximate price given the details of the property as input.

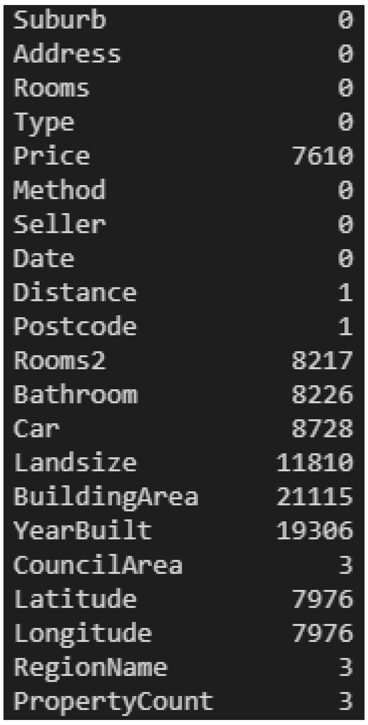
Method and experiment

Data pre-processing

To achieve best results via the use of regression, it is recommended that each of the column of the dataset is cleaned and correctly processed in order to decrease the noisiness and any incompleteness of the collected data.

Missing values

As shown below on Figure 1, the given dataset has been found to have a lot of columns suffering from the lack of data – the worst offender, *BuildingArea*, does not have data for nearly 60.5% rows of the entire dataset, which will have to be dropped.



```
Suburb              0
Address             0
Rooms               0
Type                0
Price            7610
Method              0
Seller              0
Date                0
Distance            1
Postcode            1
Rooms2           8217
Bathroom         8226
Car              8728
Landsize        11810
BuildingArea    21115
YearBuilt       19306
CouncilArea         3
Latitude         7976
Longitude        7976
RegionName          3
PropertyCount       3
```

Figure 1: Amount of missing values for each column.

While we can use the simplest solution and drop the rows containing no values for a chosen column in the case of the dependent variable *Price*, as we need correct values due to it being the target value for a good estimation and hope to offset information loss with prediction accuracy in this case, the better way for the independent, continuous values, such as *Rooms* or *Car*, that do not lack information for a severe amount of rows, is to replace the missing values with the median value of nearby properties (*Suburb*/*RegionName*) where possible, dropping the rows if we are unable to find such values. Although this operation will raise the variance of a model and might slightly skew our data, it is still more beneficial than removing a great proportion of the dataset by just dropping all of the rows with missing values, which would leave us without the rest of the information from other columns [1]. Such filling of values should also happen before dropping the rows which miss the value of *Price*.

Type conversion

There are quite a few values, such as *Bathroom*, *Car*, *YearBuilt* and *PropertyCount* that have an improper data type of a float. After filling in the values, they will be converted to integers to have correct values, as otherwise we might have 0.5 of a bathroom in a property. *Date* will also need to be converted to an integer to be used in regression.

Encoding of categorical values

An issue to resolve is how to deal with the categorical values. The proposed solution is to use one-hot encoding, as simply replacing the categories with numbers can affect weighting of the parameters [2]. However, one-hot encoding quickly grows the number of fields in size, which means that we need to remove columns with an exceedingly large amount of unique categorical values. *Suburb*, *Address* and *CouncilArea* are such columns, as they contain 344, 26737 and 33 unique categorical values. Since we have other geographical data to preserve the relation between price and place, these columns can be safely dropped.

Moreover, there are over 388 categorical data entries under the *Seller* value. Although we do not have any other data that could preserve the information given by this value, we can fortunately bin the value counts and just store in which group was the seller responsible for this property (e.g. “Under 100”).

Correlation matrix

Once we have the data correctly processed, we create a correlation matrix to determine the correlating independent variables and find out the correlation coefficients. By taking the correlation coefficients into consideration from the correlation matrix, we are able to determine which independent variables should we drop from later usage in the regression models.

Outliers and regression models

Outliers, by having a direct effect on the mean and standard deviation of a variable [3], can affect the line of best fit of a regression model. In the worst case scenario, it may create an association between variables which would not exist without the outlier. The Figures 2 and 3, shown below, show just two box plots of columns of the dataset, although many more variables have the same issue in this dataset. Currently, what seems to be the main drive in the field of regression is to leave the outliers as they are and use robust regression models instead, which are used to overcome the problem of invalid initial assumptions of a dataset, such as the lack of outliers [4].

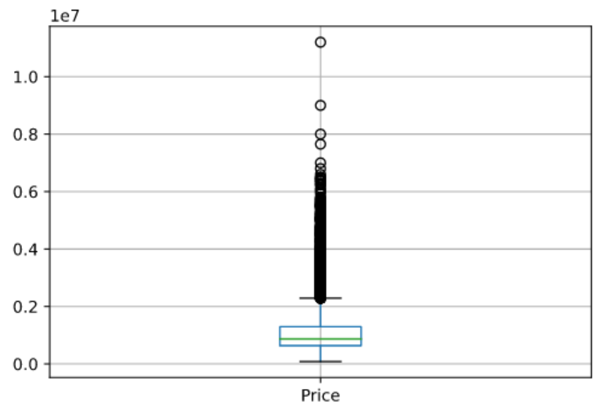


Figure 3: Price outliers.

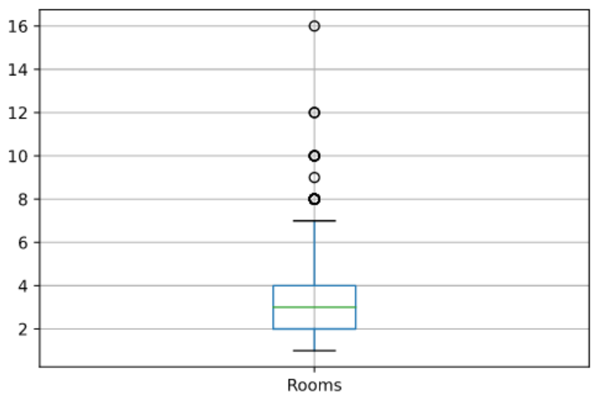


Figure 2: Rooms outliers.

Therefore, the experiment made in this work will be to evaluate and compare a number of robust regression models, such as Huber, RANSAC or Theil-Sen Regression, with standard regression models, such as Ridge, Lasso or Linear regression, on this *House Price Dataset*. Scikit-learn implementations of these models are going to be used for this aim.

Evaluation of regression models

The used regression models will be evaluated on a number on standard metrics. The chosen metrics will be the R^2 score, Mean Absolute Error (MAE), and Mean Squared Error (MSE) and Maximum Residual Error (MRE). The dataset is going to be partitioned into two parts – 80% of it will be used for training each model, while 20% will be used as a validation set.

Results and discussion

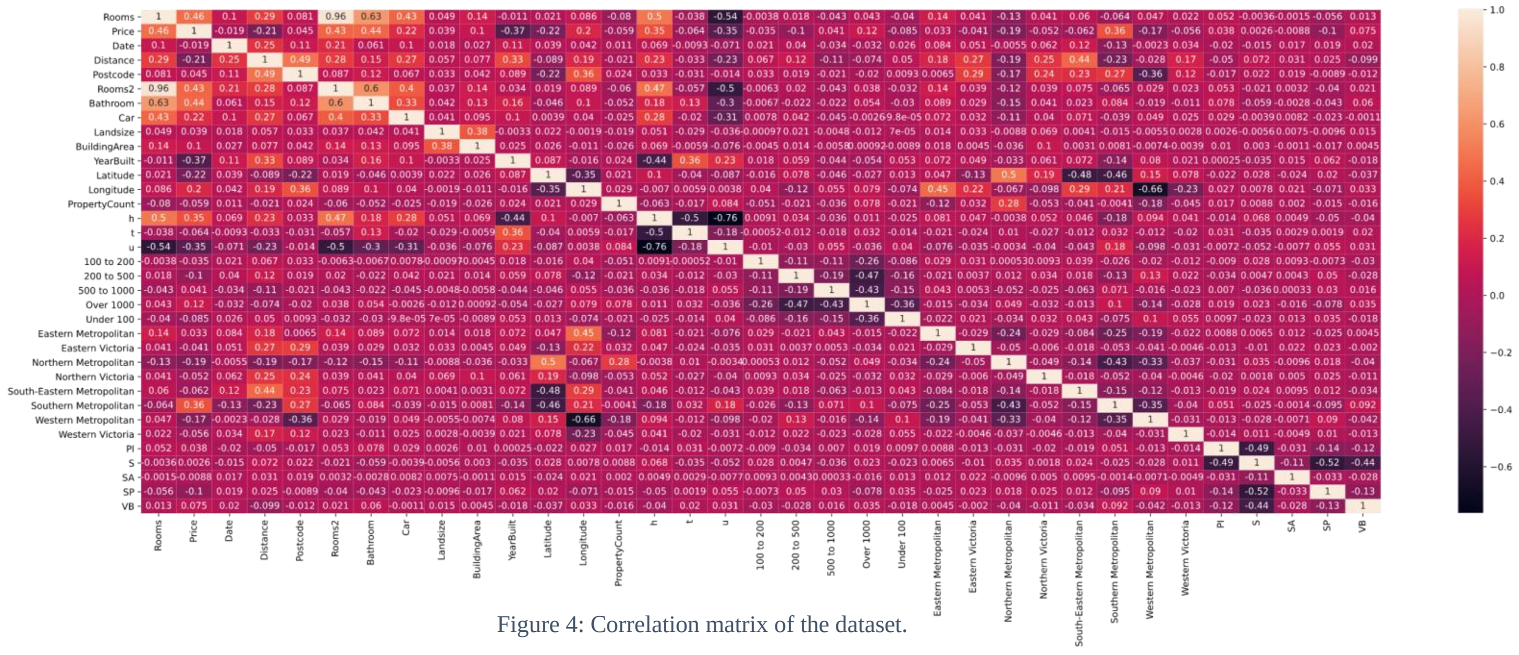


Figure 4: Correlation matrix of the dataset.

After performing the previously mentioned data pre-processing, a correlation matrix was made, containing the new one-hot columns, as shown above on Figure 4. By taking the correlation coefficients from the correlation matrix into consideration, we were able to determine which independent variables should we drop from later usage in the regression models. *Rooms2*, while having a similar correlation coefficient as the *Rooms* value to *Price*, is mainly a duplicate of *Rooms* and therefore was chosen to be dropped as it contained missing values in contrast to *Rooms*. The transaction methods, currently one-hot encoded and previously stored in the *Method* value, were also unnecessary as they did not seem to affect the price at all, therefore they also were chosen to be dropped. Although *Latitude* and *Longitude* values showed weak correlations, they were nominal variables. This means that while they might have been helpful, usually the highest priced properties in towns are in a circular, centre area, which is most likely also true for this dataset. If we treated the correlation of those values linearly, we would most likely be introducing bias into this dataset, therefore we also chose to drop those values. Furthermore, *Date*, *Postcode*, *Landsize*, *BuildingArea* and *PropertyCount* were estimated to be safe to drop as they did not contribute any significant correlation to the value of *Price.*

Once we have dropped the chosen values of the dataset, we split the dataset into two parts and fitted the following models in the following order: Linear Regression, Lasso Regression, Ridge Regression, Huber Regression, RANSAC Regression and Theil-Sen Regression. The value of the alpha parameter for Lasso and Ridge regressions has been set to 0.01. The maximum iteration value for every regression model except Linear, Huber and RANSAC has been set to 10,000. We also collected the MSE, MAE, R^2 score and the MRE for each model evaluated. A table of those results is shown below. The amount of decimal points for MSE, MAE and MRE will be cut to two to allow more clarity, and five in the case of the R^2 score and time taken.

Type of regression model	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	R^2 score	Maximum Residual Error (MRE)	Time taken
Linear	163471276942.30	268721.21	0.58272	4968261.5	0.05143 seconds
Lasso	163471856360.96	268724.38	0.58272	4968200.36	44.19279 seconds
Ridge	163440866140.06	268251.74	0.58280	4968194.03	0.01915 seconds
Huber	182166878119.92	263794.62	0.53500	5817767.36	7.17674 seconds
RANSAC	216665518948.41	274487.01	0.44694	6090272.0	0.45216 seconds
Theil-Sen	200297289199.80	297703.02	0.48872	5200308.41	17.78719 seconds

Table 1: Metrics of the evaluated models.

We have plotted the predictions of each model against the true *Price* values of the validation as well, which are shown below as Figures 5, 6, 7, 8, 9 and 10.

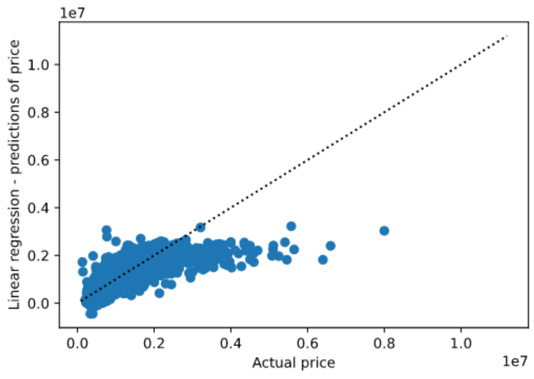


Figure 5: Linear regression predictions against actual prices.

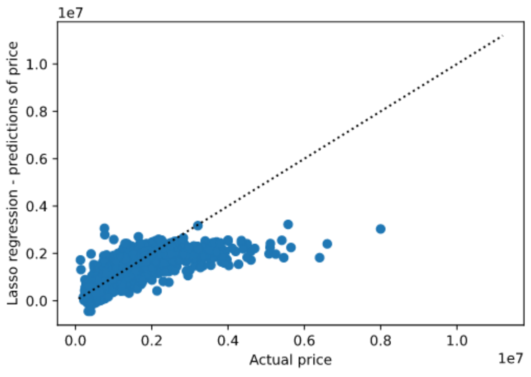


Figure 6: Lasso regression predictions against actual prices.

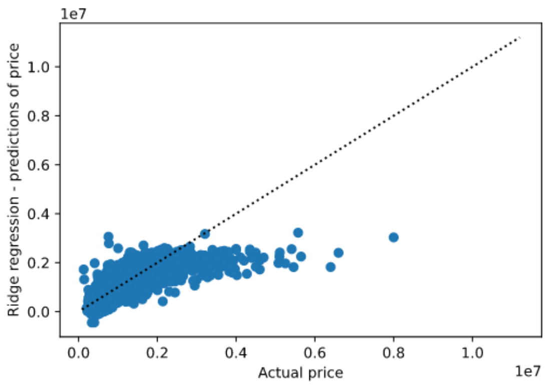


Figure 7: Ridge regression predictions against actual prices.

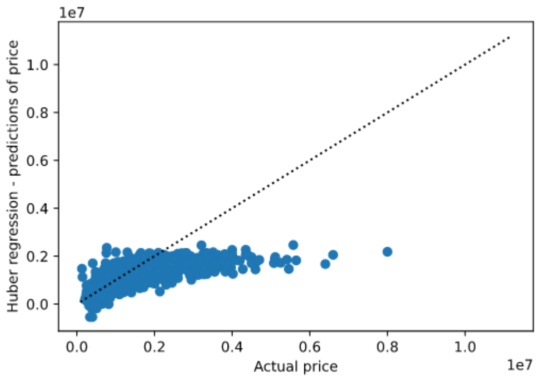


Figure 8: Huber regression predictions against actual prices.

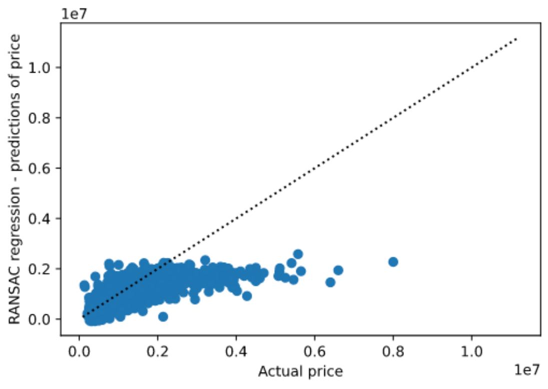


Figure 9: RANSAC regression predictions against actual prices.

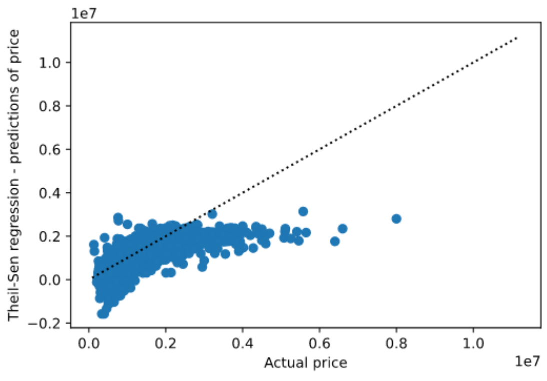


Figure 10: Theil-Sen regression predictions against actual prices.

Having the results collected and visualized, we can proceed with their evaluation and discussion. The best R^2 score, which measures "how well unseen samples are likely to be predicted by the model, through the proportion of explained variance" [5], has been achieved by the Ridge regularisation regression model, with a value of 0.58280 (58.28%). Surprisingly, the standard linear regression model was not far from the top result, achieving a R^2 score of 0.58272 (58.272%). Considering how many columns of the dataset had missing values, achieving a result this high is quite satisfactory, as we were expecting the best R^2 score to be around 0.40 due to the lack of data cleanness of the dataset. If we take a look at the large MSE, MAE and MRE values of each regression model, we can see that we have obviously introduced a lot of variance due to filling out the missing values with the median. While we could boost the R^2 score and lower the variance by simply dropping all of the rows that contained missing values during the data pre-processing stage, our regression models would most likely be overfit and therefore be unprepared to predict a house price for any unseen data that might have been in the dropped rows. However, since all of the standard linear regression models have achieved similar results in terms of the metrics, this shows that the initially addressed data cleanness issue of the dataset has not been completely resolved. Moreover, comparing the time results of each algorithm has been strapped of its usability as most of the more advanced models have failed to converge due to the unclean data of the dataset, models such as Lasso or Theil-Sen.

Interestingly, robust regression models performed far worse than the standard, non-robust linear regression model when we compare the metrics of MSE, MAE, R^2 and MRE. Such a result might point to a possible characteristic of this dataset: that the outliers are needed in this data, and giving less weight to outliers might have done more harm than good to us in this experiment. This is most likely explained by the relatively small size of the dataset and the unequal distribution of classes in categorical data columns, due to which we might need each data point that we can use, regardless of whether it is an outlier.

If we investigate the plots, we can see that there is a common problem amongst all of the regression models, which is the inability to predict houses of a price value higher than 3 million. A quite surprising discovery, and drawback at the same time, was to find that one of the regression models, Theil-Sen Regression, actually predicted negative values for some of the house prices.

Conclusion

In our experiment, we have evaluated a total of six (multi) linear regression models: Linear, Lasso, Ridge, which represented the non-robust category, and Huber, RANSAC and Theil-Sen, which represented the robust category of regression models, as we have found the dataset to suffer from a number of outliers in most of its columns. The evaluation was done on the *House Price Dataset*, on which we firstly used a number of data pre-processing techniques. Our experiment managed to find an acceptable linear regression model for general use, with it being the Ridge regression model with an R^2 score of 0.58272 (58.272%), and the robust models to fall behind the standard non-robust models. While this R^2 score makes the model reasonable to use for solving our problem and therefore for predicting house prices, using the linear models from this experiment in real-life applications would most likely still require to verify the prediction on neighbouring, similar properties. However, as stated in the previous section, this drawback is mainly drawn from the problems of the data in the dataset. To address this issue, either other data pre-processing techniques could be used or more data could be collected to improve the accuracy of the regression models.

One of the drawbacks of our approach was the significant reduction of geographical data. For better use of the geographical data such as *Postcode*, *Longitude* and *Latitude*, the data could be initially clustered using techniques such as k-means or gaussian mixture models to gather more information about what types of houses are based in a geographical cluster, which could boost our regression models.

To avoid the problem of negative value predictions as seen on the plot of Theil-Sen regression model predictions, log of *Price* could be used instead, which should allow the model to base on the distribution of *Price* rather than its exact values [6]. This could perhaps make a positive impact on enlarging the range of predictions of the models.

References

[1] S. Saha, "Here's Why You Shouldn't Drop Missing Values", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/few-reasons-to-not-drop-missing-values-575a8d2b6a41. [Accessed: 04 May 2021].

[2] N. Chauhan, "From Data Pre-processing to Optimizing a Regression Model Performance", *KDnuggets*, 2019. [Online]. Available: https://www.kdnuggets.com/2019/07/data-pre-processing-optimizing-regression-model-performance.html. [Accessed: 04 May 2021].

[3] K. Grace-Martin, "Outliers: To Drop or Not to Drop", *The Analysis Factor*. [Online]. Available: https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/. [Accessed: 04 May 2021].

[4] Moeedlodhi, "How Outliers Can Pose a Problem in Linear Regression.", *Medium*, 2020. [Online]. Available: https://medium.com/swlh/how-outliers-can-pose-a-problem-in-linear-regression-1431c50a8e0. [Accessed: 04 May 2021].

[5] "3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation", *Scikit-learn.org*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score. [Accessed: 04 May 2021].

[6] RegressForward, "Getting negative predicted values after linear regression", *CrossValidated*, 2015. [Online]. Available: https://stats.stackexchange.com/questions/145383/getting-negative-predicted-values-after-linear-regression. [Accessed: 04 May 2021].