Department of Economics

Prof. Damian Kozbur

Machine Learning for Economic Analysis

WRITE-UP

FOR THE REPLICATION PROJECT ON

BUSINESS ANALYTICS PROJECT IN MACHINE LEARNING

BASED ON DUBOCANIN, BÜHLMANN & OFFERINGA (2019)

University of Zurich[UZH]

Paolo Neri	paolo.neri@uzh.ch	16-729-287
Gabriele Brunini	gabriele.brunini@uzh.ch	20-742-219
Maxime Vandierendounck	maxime.vandierendounck@uzh.ch	16-730-137

January the 20th, 2021

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DDML	Double-Debiased Machine Learning
ML	Machine Learning
MSE	Mean Squared Error
OLS	Ordinary Least Squares
XGBoost	Extreme Gradient Boosting

1. INTRODUCTION

All through history, housing booms and busts have often been detrimental to both financial stability and the real economy. Many significant episodes of banking distress have been associated with boom-bust cycles in property prices. IMF research shows that boom-bust patterns in house prices preceded more than two-thirds of the nearly 50 systemic banking crises in recent decades. In contrast to housing cycles, boom-bust cycles in stock prices are much less likely to trigger systemic banking crises (Zhu, 2014).

Even when housing busts do not have a sizeable financial stability impact, they can affect the real economy. Research shows that recessions in OECD countries are more likely given a house price bust. Such recessions also tend to be much more in-depth and generate more unemployment than normal recessions. In short, there is abundant evidence that housing cycles can be a threat to financial and macroeconomic stability. Hence it is crucial to keep an eye on current housing market developments to keep them going through another boom-bust cycle (Zhu, 2014).

Housing booms have different characteristics across countries and periods. What is expected is that it very often damages financial stability and the real economy when the bust comes. The tools for containing housing booms are still being developed. The evidence on their effectiveness is only just starting to accumulate. Nevertheless, we need to have some predictive and analytical tools to detect market failures and promptly correct them (Zhu, 2014).

In this research project, we wanted to replicate the work of Dubocanin et al. (2019). The data describes Melbourne's properties where specific characteristics related to geography, the seller, and the building help us better understand price heterogeneity among multiple observations. We know the benefits of precise predictions as housing markets, in general, can affect the real economy. For example, the United States housing bubble of 2008 arguably worsened the entire world's subsequent financial crisis. The whole process was complex, interconnected, and vast—and it was all underpinned by appreciating home prices.

The goal of this project is twofold. On the one hand, the prediction models provide a more efficient way to predict house prices, mainly using features regarding the building's characteristics and geographical location. By having a working model that predicts house prices, we can keep an eye on current housing market developments to keep them from going through another boom-bust cycle.

On the other hand, we have implemented the double debiased machine learning (DDML) method. This step is essential, as this novel method allows us to obtain better coefficients for our predictors of interest by using more information in our estimation procedure. It could lay the basis for inference and, therefore, a step toward future research on this topic. We limited ourselves to report the obtained coefficients for this project's scope and compare them with standard OLS instead of drawing causal statements. We emphasize the implementation of Machine Learning (ML) models for economic purposes.

We developed the work as follows: **Section 2** explores the dataset. **Section 3** describes the ML techniques, while **Section 4** presents the results. Finally, **Section 5** discusses potential improvements for further research.

2. DATA ANALYSIS AND PRE-PROCESSING

The dataset has 34'857 observations related to a house in the region of Melbourne. Looking at the dataset structure, we notice that *Price*, the outcome variable, is explained by the other 20 features presented in this section. **Figure 1** describes the distribution of the dependent variable *Price*. In part (a) of the figure, one can notice that housing prices are highly skewed towards the left, which means that the data distribution is far from Gaussian. Therefore, we decided to analyze by rescaling the predicted variable *Price* using a logarithmic scale, so that data is now normally distributed. The result of that transformation can be seen in figure (b).

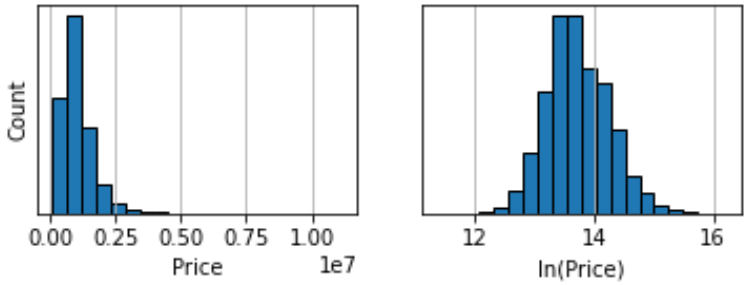

Figure 1: Distribution of Price (a) and ln(Price) (b).

Firstly, the location of missing values is detected and is described in **Table 1** in the **Appendix**. Secondly, the dependent variables are investigated in a more detailed manner. One notices three different sub-categories: seller-related variables, building-related variables, and geographical variables.

The first sub-category, the seller-related variables, encompasses the following features *SellerG*, *Method*, and *Date*. *Seller G* has 388 unique values and gives information about the building's seller, while *Method* describes the selling-method. Date enables the further description of the dataset in a temporal context. **Figure 2** describes the distribution of sold houses between March the 9th of 2016 and October the 3rd of 2018. Since it is assumed that these variables do not impact *Price*, they are discarded.

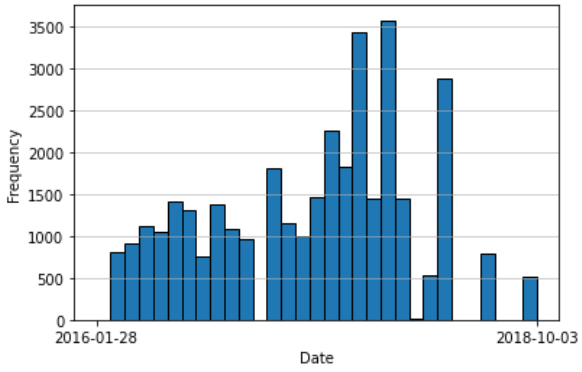

Figure 2: Sells' Temporal Distribution.

The second sub-category corresponds to the seven building-related variables: *Bedroom2, Rooms, Bathroom, Car, Land Size, Building Area, Types,* and *Year Built*. **Figure 3** shows the correlation coefficients between the numerical variables of this sub-category and *Price*. One notices that *Bedroom2* and *Rooms* are highly correlated. Further, *Bedroom2* was collected through web-scraping and, therefore, has more missing values. Further, *Rooms, Bathroom,* and *Building Area*'s variables display the highest correlation coefficients with *Price*.

Figure 3: Correlation Matrix for Building-Related Features

The distribution of different *Types* is investigated in **Figure 4**. The categories belonging to this feature are more detailly described in **Table 2** in the **Appendix**. Indeed, type "*u"* seems to be the cheapest on average while type "*h"* looks more expensive than type *"t"*.

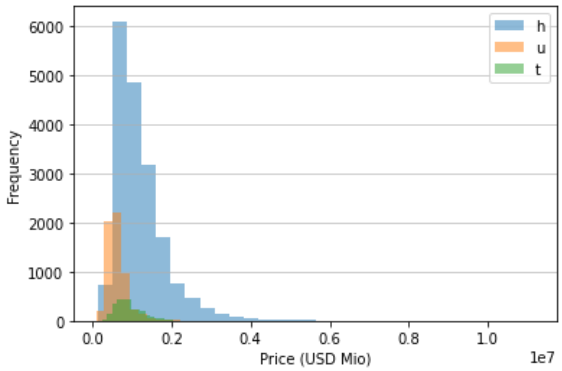

Figure 4: Distribution of Price given Types

The last sub-category represents the nine geographical features: *Address, Region Name, Council Area, Suburb, Postcode, Latitude, Longitude, Distance,* and *Property Count.* However, *Latitude* and *Longitude* allow us to locate the houses and situate where the more expensive buildings are (see **Figure 5**). This step further emphasizes the importance of *Region Name, Council Area, Suburb* and *Postcode*.

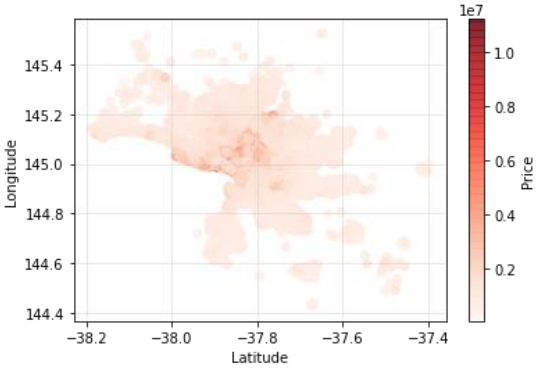

Figure 5: Price in relation to geographical Coordinates

Finally, the relationship between *Distance* from the Central Business District and *Price* is displayed in **Figure 6**. A weak negative correlation between these variables is observed.

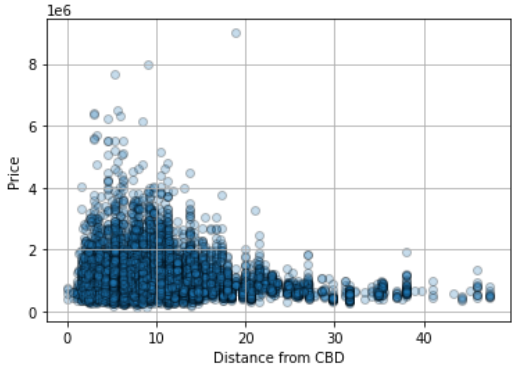

Figure 6: Relationship between Distance from the Central Business District and Price

After displaying and analyzing each predictor's correlation to the y variable, we decided to remove some features that do not seem to have any predictive power (*Method, SellerG, Address*). Additionally, we removed the highly correlated features to other variables (*Suburb, Postcode, Bedroom2)*. This reduces the explanatory part of the dataset to 15 columns.

Some feature engineering has been implemented on the dataset, namely, we created a variable representing the number of years since the house was built, calculated as:

```
df_imputed['houseAge'] = 2020-df_imputed['YearBuilt']
```

Then, using the *"day of year"* variable, which informs on when the house was sold, we created the *season* variable. This is done in order to investigate whether there is some correlation between the housing price and the season:

```
for i in df_imputed['day_of_year']:
    if i in spring:
        season = 'spring'
    elif i in summer:
        season = 'summer'
    elif i in fall:
        season = 'fall'
    else:
```

```
        season = 'winter'
    list.append(season)
```

Further, we have decided to not merely drop the missing values from the dataset, so to not lose too many records. Instead, when the missing value is part of a continuous feature, we imputed it using the column's mean, while if the encountered missing value is categorical, for which taking the mean does not have much sense, we imputed the most frequent category among all the feature levels. The only records that we decided to drop are the ones having missing housing price. This pre-processing step is a reduction in the number of records from the initial 34'857 to 27'247. We compared this result with the pre-processing step of dropping all the rows with missing values from the dataset, which result in a reduced dataset having only 8'887. This variation in number of rows can affect the trained ML models' overall accuracy and can, therefore, worsen the overall economic analysis.

In the last step, the dataset is split into two parts:

- *Training* (80% of data) having 21'797 records
- *Testing* (20% of data) having 5450 records.

3. MODELS AND METHODOLOGY

In the prediction assessment phase, the MSE is the main evaluation criteria of each analyzed model's accuracy due to its simplicity and execution speed. As a consequence, one can look at the MSE formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \quad (\quad _{Yi} - \hat{f}(x_i))^2$$

3.1 Linear Regression-Based Models

3.1.1 Methodology

The first models represent different flavours of linear regression models. Firstly, the Ordinary Least Square base model is used. This is perceived as the lower bound in terms of performance. Secondly, two regularization techniques, Lasso and Ridge, are implemented on the baseline model. By doing so, the hyperparameter t is used to increase the flexibility of the linear models. The standard OLS model results in an MSE of 0.0709. This result is used as a baseline to assess the accuracy of the additional models presented in **Section 3.**

A 10-fold cross-validation procedure was conducted to find the optimal penalty parameter t that minimized the mean squared error (MSE) across the training sample.

3.1.2 Lasso

Lasso regression is a shrinkage method that adds a penalty term to a standard OLS regression minimization problem. This ML technique aims to prevent overfitting by reducing model complexity. The advantage is that the lasso penalty can shrink parameters to zero, thus pruning the model of variables, which it considers non-essential. In other words, Lasso has a variable selection property (James et al., 2013).

The formula for the Lasso estimate looks as follows, where $t \geq 0$ is a tuning parameter:

$$(\hat{\alpha}, \hat{\beta}) = arg\ arg \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \right\} \quad subject\ to \quad \sum_j |\beta_j| \leq t$$

It has been shown that ridge regression and Lasso have advantages and disadvantages when it comes to different scenarios. Lasso works best when there is a small number of large effects, but it does best when there is a number of moderate-sized effects, and ridge regression does best when there is a large number of small effects (Tibshirani, 1996). The selection of an adequate penalty parameter t is of significant importance for Lasso. When t is equal to zero, then the lasso predictions are equal to those from OLS (James et al., 2013). The 10-fold cross-validation procedure was conducted to find the optimal penalty parameter lambda, and the plotted results are shown below.

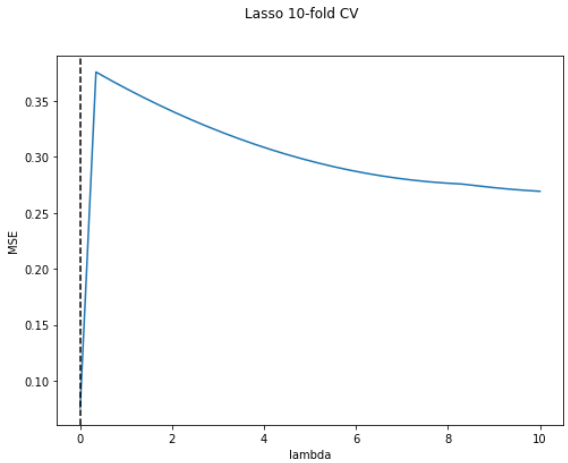

Figure 7: Relationship between MSE and the chosen lambda (t) values for Lasso Regularization

The cross-validation procedure suggested that the optimal t is zero, as the range for lambda was set to be between 0 and 10, and the optimal lambda ended up being 0. The evolution of the *MSE*, given different t levels, and the optimally chosen t indicated with a dotted line, is shown in **Figure 7**. As expected, the final *MSE* result is precisely the one we obtain using an OLS estimator, since the regularization parameter is set to zero (final *MSE = 0.0709*).

3.1.3 Ridge

We test another shrinkage method, namely: Ridge regularization. The difference lies in how the penalty term is calculated and thus changes how parameters shrink. The penalty shrinks them towards zero

without actually setting them equal to zero. Hence, in contrast to lasso regression, ridge regression includes all regressors in the final model.

The following formula presents the minimization problem for the ridge regression estimation, where $t \geq 0$ is a tuning parameter (James et al., 2013):

$$(\hat{\alpha}, \hat{\beta}) = arg\ min \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \right\} \quad subject\ to \sum_j \beta^2 \leq t$$

The 10-fold cross-validation procedure was conducted to find the optimal penalty parameter lambda, and the plotted results are shown below.

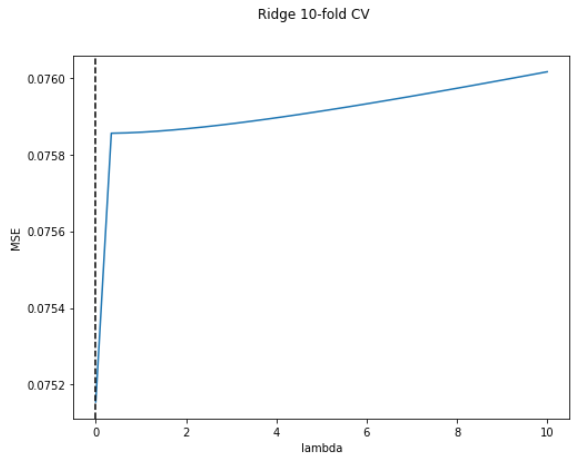

Figure 8: Relationship between MSE and the chosen lambda (t) Values for Ridge Regularization

As the image shows, the cross-validation procedure suggested that the optimal t was again 0. The evolution of the *MSE*, given different t levels for regularizing the Ridge regression model, and the optimally chosen t of zero indicated with a dotted line is shown in **Figure 8**.

3.2 Random Forest

Random forest (RF) is a very popular supervised algorithm used for both classification and regression tasks. It grows and combines multiple decision trees to create a forest. A decision tree can be understood as a flowchart that draws a distinct pathway to a decision or outcome by starting at a single point and then branching off into two or more directions. Each branch offers different possible outcomes, incorporating various decisions and chance events until an outcome is achieved.

The logic is that multiple uncorrelated models – i.e., the individual decision trees – perform much better as a group than they do alone. The key here lies in the fact that there is a low (or no) correlation between the individual models – that is, between the decision trees that make up the larger Random Forest model. Hence, while individual decision trees may produce errors, most of the group will be correct, moving the overall outcome in the right direction.

The most convenient benefit of using random forest is its ability to, by default, correct for decision trees' inconvenient habit of overfitting to their training set. Using the bagging method and random feature selection when executing a random forest almost wholly resolves the overfitting, leading to accurate predictions. Furthermore, even if some data is missing, Random Forests usually maintain their accuracy. After carefully tuning the hyperparameters of the RF model, we tested the model using the test set and obtained an *MSE* score of 0.04376, which represents a wide improvement with respect to basic regression analysis models: the MSE decreased by 2.7 per cent.

3.3 KNN

In contrast to linear regression, which represents a parametric model, K-nearest neighbours (KNN) regression is a well-known non-parametric model. Parametric models, and linear regression, in particular, have many advantages over non-parametric models: they easy to fit, have simple interpretations, and it is possible to test for statistical significance. However, one of the main disadvantages of parametric models is that there are strong assumptions regarding the form of $f(X)$. In case that the real relationship is far from linear, non-parametric models may be a viable alternative (James et al., 2013).

Since it is plausible that the true relationship between *Price* and the other features is not linear, we have performed KNN regression to have a more flexible approach. The model works as follows:

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_0} y_i$$

For an arbitrary prediction point x_0 and a chosen value K, the model operates in two steps. The first step consists of identifying the K training observations that are nearest to x_0. N_0 denotes these observations. In a second step, $\hat{f}(x_0)$ is determined by computing the average of N_0. The optimal value of K relies on the bias-variance trade-off. For instance, $K = 1$ implies that the accuracy of the training sample is going to be 100%. KNN now takes on the form of a step function. By increasing the value for K, a smoother fit can be achieved. A small value for K implies low bias but high variance. With an increasing value for K, the smoothness increases, and as a consequence, it may cause bias. A cross-validation procedure can be applied in order to determine the optimal value. One of the main problems that can occur when performing a KNN regression is known as the "curse of dimensionality." When p in a p -dimensional space is large, the K nearest observations may be far from the actual test observation. The consequence would be a poor fit (James et al., 2013). We set the value of K to 5 and used the Minkowski distance metric $p = 2$ to equal the Euclidean Distance.

The resulting *MSE* obtained from the KNN regression is 0.08933, not far from the baseline method's performance (slightly worse). This result is probably due to the above-mentioned "curse of dimensionality," or the possibility that the real relationship is indeed linear. We think it seems reasonable to assume that other non-parametric models might perform better than KNN regression, so we tried another approach: XGBoosting regressor.

3.4 XGBoosting Regressor

XGBoost stands for "Extreme Gradient Boosting," where the term "Gradient Boosting" originates from Friedman (1999). Like random forests, gradient boosting is a set of decision trees, a series of sequential steps designed to answer a question and provides probabilities, costs, or other information, which are then aggregated in some way.

The two main differences between the two techniques are:

1. How trees are built: random forests build each tree independently, while gradient boosting builds one tree at a time. This ensemble model works in a forward stage-wise manner.
2. Combining results: random forests combine results at the end of the process (by averaging or "majority rules") while gradient boosting combines results along the way.

In particular, XGBoost employs some precautions that make it exceptionally successful, particularly with structured data similar to the dataset we are using in our economic analysis (XGBoost, 2016). The particularities to be noted are:

- XGBoost computes second-order gradients, i.e., second partial derivatives of the loss function (similar to Newton's Method), which provides more information about gradients' direction and how to get to the minimum of our loss function. While regular gradient boosting uses our base model's loss function to minimize the overall model's error, XGBoost uses the second-order derivative approximation.
- Advanced regularization (L1 & L2) is already implemented in XGBoost, which improves model generalization.

In our economic analysis, XGBoost achieves an *MSE* of 0.03946, which is the best-achieved score among the ML models we trained by far, improving on the OLS model *MSE* by more than 3 per cent! So we can say that, in this case, a non-linear model is indeed performing the regression task achieving a lower error.

3.5 Double-Debiased Machine Learning

For the double-debiased ML (DDML) technique, we closely followed the exercise session's procedure, but we did not include any instruments in our analysis. Additionally, we have not transformed the independent variable *price* in logarithmic scale as we did in the prediction task, since the step distorts the interpretation of our coefficients and the analysis' results are generally less interpretable.

It is sufficient to have a high number of features to compute the DDML procedure. First, we split the sample into a main and an auxiliary sample. In the second step, we used the auxiliary sample to estimate the coefficients that are later needed to compute the double-residualized OLS estimator with the primary sample.

Because the sample was split to compute the whole procedure, we performed 100 iterations for each coefficient and afterwards took the median value. This would lead to more stable coefficients for the variables of interest. Regarding the sample split, it was possible to implement a probabilistic approach by computing the split using a binomial function, given that we had a large sample. The split's proportions resulted in being close and significantly reduced the time it took to perform the DDML procedure:

`split = numpy.random.binomial(1, p, len(D))`, where `len(D)` is the dataset length.

In **Section 4**, we present the results showing the comparison between the coefficients resulting from standard OLS and those obtained through DDML, all shown in **Table 2**.

4. RESULTS

Table 1 recapitulates the results of the models presented in **Section 3**.

Model	Parameter	MSE	Model	Parameter	MSE
OLS	-	*0.0709*	**KNN**	# of neighbours = 5	*0.08933*
Ridge	*t = 0*	*0.0709*	**Random Forest**	n_estimators=1500 max_depth=12, min_samples_split=6, min_samples_leaf=5	*0.04376*
Lasso	*t = 0*	*0.0709*	**XGBRegressor**	n_estimators=6000, learning_rate=0.01 max_depth=4 max_features='sqrt' min_samples_leaf=15 min_samples_split=10	*0.03946*

Table 1: Results

The table shows all the tested models, which have been accurately tuned to pick the best hyperparameters for the dataset on which they are trained. We observe a lower *MSE* value using the XGBRegressor model, being trained using 6000 estimators, a learning rate of 0.01, max_depth of 4, minimum sample leaf of 15 and a maximum sample split value of 10. The result suggests that for our economic analysis, the model can predict better than the baseline models.

As already mentioned, for the KNN model, one can see that the MSE is slightly higher than the one obtained with Ridge, Lasso, and OLS. It is experimentally proven that, for this dataset and the

regression task at hand, KNN does not necessarily have to be favoured over linear regression due to the disadvantages of such a non-parametric model (James et al., 2013).

In **Table 2**, the coefficients of the variables of interest are reported. These were found using DDML with three different ML techniques (Lasso, Ridge, and Boosting), and we compare them with the coefficient that resulted from a standard OLS regression.

Variable of Interest	Lasso	Ridge	Boosting	OLS
Rooms	381048.10	377832.56	261628.34	310737.95
Distance	-5628.86	-36134.82	3424.49	-33165.16
Bathroom	158450.03	131258.65	181235.26	141660.05
Car	66950.43	50417.46	50163.77	43513.17
Landsize	-2.23	-0.94	-0.04	3.58
Building Area	-1563.81	-1268.80	-410.22	35.08
House Age	3158.55	2960.23	4377.57	3656.22

Table 2: Coefficients provided by the DDML procedure with different ML methods compared to standard OLS

It gets evident that the coefficients for the variable of interest all point in the same direction, except for *Distance*, *Landsize*, and *Building Area*. When looking at the reported coefficient for *Distance*, we see that using Boosting for DDML results in a completely different coefficient. Given that *Distance* measures the Distance to the business district, it makes sense to assume that the real effect of *Distance* on *Price* is negative. We see that OLS yields completely different results regarding the differences between the coefficients on *Landsize* and *Building Area*.

It is hard to say why these (in part) significant differences among the coefficients are observed. Probably it is because the DDML procedure heavily depends on how each ML method works, and as a consequence, some ML techniques may be more suitable than others. Another possible reason may be that we computed 100 iterations of each DDML procedure and consequently took the median to determine the coefficients since only one iteration led to even more significant differences. More iterations may result in smaller differences.

Table 5 (see **Appendix**) provides results of the same ML models trained using a different dataset, namely: every row that includes a missing value, is dropped. This results in a dataset having only 8'887 rows, since many records have missing values and are therefore not included in the dataset. We performed hyperparameter tuning on those models as well: the results are carefully reported in the table.

5. DISCUSSION AND CONCLUSION

In this project, we aimed to replicate Dubocanin et al. (2019) and further emphasized several ML techniques to perform predictions using the *MSE* as our accuracy metric. In **Section 2**, we investigated the raw dataset to understand which features could significantly drive the outcome variable *Price*. An Exploratory Data Analysis was executed to facilitate further feature engineering and pre-processing (e.g., transforming into logarithmic scale). **Section 3** describes the six chosen models – three linear and three non-linear ones – and highlights the main drawbacks and advantages. All of them are tuned to optimize the result on the out-of-sample set, the test set. **Section 4** presents the results, and clearly shows that XGboost model achieves the best performance on our dataset. These results can be used to closely monitor the market's health in general, and to better predict market bubbles and likely bursts, so to apply macroeconomic measures to adjust it.

Even though our models have accurate and consistent results, some further improvement is still possible. First, we decided to drop observations if the outcome variable is missing and impute the missing explanatory variables by their mean (if they are numerical) or by the most frequent category (if they are categorical). This technique might not be optimal since it directly impacts the distribution of specific features. Further implications about the missing values can be emphasized.

Additionally, the raw dataset has not been extended to include other significant variables. Since we know the observations' geographical and temporal characteristics, potential additional features affecting *Price* could have been added to the dataset. For example, the local unemployment rate or other macroeconomic variables could have been used.

Lastly, we only characterize the accuracy of our models using the *MSE* value. This metric selection could have been expanded by AIC, BIC, or Adjusted R-Square values.

Furthermore, we got a first insight on how ML techniques can be applied to topics for which economists usually care about inference. DDML could be applied to various economic topics, and it was interesting to see how such a method could revolutionize economic analysis conduct in future works.

BIBLIOGRAPHY

Dubocanin, N., Bühlmann, J.E., & Offeringa, P. (2019). *Business Analytics Project in Machine Learning*.

Friedman, J. (1999). *Greedy Function Approximation: A Gradient Boosting Machine*. Annals of Statistics, 29(5).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R.* Springer.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288. Retrieved from https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x

Tianqi Chen, Carlos Guestrin (2016). *XGBoost: A Scalable Tree Boosting System*

Zhu, M. (2014, June 5). *Housing Markets, Financial Stability and the Economy*. International Monetary Fund. https://www.imf.org/en/News/Articles/2015/09/28/04/53/sp060514

APPENDIX

Suburb	0	**Bathroom**	8226
Address	0	**Car**	8728
Rooms	0	**Landsize**	11810
Type	0	**BuildingArea**	21115
Price	7610	**year built**	19306
Method	0	**CouncilArea**	3
SellerG	0	**Latitude**	7976
Date	0	**Longitude**	7976
Distance	1	**Region name**	3
Postcode	1	**Propertycount**	3
Bedroom2	8217	-	-

Table 3: Location of Missing Values

h	villa, cottage, house, semi terrace
u	duplex, unit
t	townhouse

Table 4: Description of Types

Model	Parameter	MSE	Model	Parameter	MSE
OLS	-	*0.05866*	**KNN**	# of neighbours = 5	*0.0532*
Ridge	*t = 0.2592*	*0.05866*	**Random Forest**	n_estimators=1500 max_depth=12, min_samples_split=6, min_samples_leaf=5	*0.03777*
Lasso	*t = 0.2592*	*0.05866*	**XGBRegressor**	n_estimators=6000, learning_rate=0.01 max_depth=4 max_features='sqrt' min_samples_leaf=15 min_samples_split=10	*0.02976*

Table 5: Results using another variation of the dataset