Bachelor's Thesis

UNIVERSITEIT VAN AMSTERDAM

The Impact of Feature Extractions in Automated Valuation Models

Thesis Advisor: Felipe Dutra Calainho

Author: Seung Woo Hong

Student Number: 10879420

Faculty of Economics and Business

Business Administration

Specialization: Finance

Submission Date: 30th, June 2021

Table of Contents

Abstract

With regards to the importance of real estate valuation, this study aims to observe the change in prediction performance in Automated Valuation Models (AVMs) in accordance with the application of machine learning algorithms. This study mainly focuses on analyzing the impact of applying feature extraction techniques to AVMs. The feature extraction methods we had focused on the experiments were based on Principle Component Analysis (PCA), K-Means, Autoencoder methodologies. For the experiment, this research used the Russian real estate dataset provided by Sberbank. This dataset was suitable for our experiments with feature extraction as it originally contained more than 300 features. First, this paper conducted a comparative analysis of prediction performance by repeating the training and testing process 37 times. Next, this study confirmed the effect of feature extraction using PCA, K-Means, and Autoencoder on the linear regression model through regression analysis by testing the significance of the added coefficients. This paper finds the results that the most appropriate feature extraction was different depending on the model and feature selection method used. The alternative hypothesis was supported in PCA and K-Means feature extraction, and the alternative hypothesis was not supported in Autoencoder feature extraction.

1. Introduction

Real estate is regarded as an asset with high investment value due to the increasing value of today's continuous demand, beyond the real estate's nature as a residential space (Manganelli, 2015). Historically, investing in real estate has provided slightly lower returns than common stocks and has a relatively low level of risk (Reilly and Brown, 2000). Therefore, changes in real estate prices have a great impact on a wide range of classes, from individuals who demand housing stability to companies that own real estate as assets. Along with the trend, various investments from individuals to institutions are being made in the real estate market. This study aims to provide applicable information that can be used for investment portfolio planning by estimating the value of real estate using the Automated Valuation Models (AVMs) methodologies based on machine learning algorithms. The performance change of the model according to various feature extraction methodologies will be analyzed.

The field of artificial intelligence has experienced repeated ups for a long time. The impact of artificial intelligence has been underestimated in the past due to difficulties in application to complex real-world problems, limitations in computational speed, and the lack of evolutions in machine learning algorithms (Buchanan, 2005). However, today, a large amount of data can be collected through various systems including search engines, artificial intelligence has evolved into a self-learning machine by analyzing big data through advanced algorithms. With these developments, artificial intelligence has been introduced throughout the industry to efficiently solve problems.

The real estate market is being discussed as one of the fields with a high possibility of using big data (Kok and Koponen, 2017). This characteristic has made the investment researchers to move on

to use machine learning based AVMs from traditional hedonic price model. Traditionally, a manual data selection and collection of meaningful features are necessary because the manual processing by conducting various social surveys assists to predict real estate prices. Today, implementing AVMs using machine learning algorithms allows data with higher dimensions to be considered at a time. However, previous researchers have emphasized the importance of selecting the right features for the modeling while using machine learning algorithms (Guyon & Elisseeff, 2003; Chandrashekar & Sahin, 2014). From this point of view, this study aims to extract meaningful information from excluded variables in automated feature selection process.

The structure of this paper is as follows. In theoretical framework, a concept and previous research of the classic hedonic price model and modern approaches of AVMs will be discussed. The focus will be describing feature selection and extraction in terms of dimensionality reduction. In the method section, this paper will discuss the process of designing and evaluating house price prediction models with various methods to answer our research question: *How do different feature extraction methods affect the performance of price prediction in AVMs?* Using the data processed with various feature extractions as input, the comparison of performance on both linear and non-linear regression models will be analyzed. Finally, this paper will discuss the contribution and limitation of our findings and how further studies can be improved.

2. Theoretical Framework

2.1. The Hedonic Price Model

The hedonic price model is a method of estimating the implicit price in a product group from the characteristics of a product. The hedonic price model was devised for the purpose of estimating the economic value of qualitative changes in consumer goods. The hedonic price model presupposes that the value of a good is determined by the characteristics of the good. The characteristics of goods are constituent elements of goods, and since they are not traded individually, they are traded as a bundle. Therefore, the price of each character cannot be observed directly. In the hedonic price model, the price of a good is determined by the price and quantity of the characteristics contained in the good, and the price of these characteristics is called the hedonic price (Court, 1939). Therefore, the hedonic function is referred to as the regression equation from individual characteristics.

The main limitation of the hedonic price model lies in the applicability of model due to its data-consuming nature (Chau & Chin, 2003). Chau and Chin (2003) argued that the model itself cannot be directly transferred in some places where the required data for the model is not collectible. Limsombunchai (2004) has pointed out issues regarding model specification procedures, multicollinearity, independent feature interactions, heteroscedasticity, non-linearity, and outlier data points might hinder the prediction performance.

2.2. Automated Valuation Models

In recent years, with the spread of the digital economy, a 'big data' environment in which a lot of information and data are produced around us has arrived, making it possible to utilize more diverse features in price-prediction models. From such movements, Automated Valuation Models (AVMs) have been actively used for predicting house prices. In practice, modern AVMs combine the complex use of linear regression, machine learning, neural networks, and other mass appraisal techniques.

In the past, most of the real estate price prediction models were based on regression equations (Valier, 2020). However, linear models might show their weakness to aspects such as outliers, non-linearity, discontinuity, and fuzziness (Selim, 2009). To overcome potential limitations, recent studies have implemented various machine learning algorithms. Limsombunchai (2004) was able to improve predictive power compared to the hedonic price model in estimating the house price in New Zealand through an artificial neural network model. Other machine learning models (tree-based, SVM, and neighbors) showed superiority over traditional models on the Scopus real estate dataset (Valier, 2020). Empirically, modern AVMs complement the disadvantages of existing hedonic price models which do not reflect the nonlinear relationships of characteristics and house price (Limsombunchai, 2004).

2.3. Supervised Machine Learning Regression Models

In this part, this paper is going to analyze tree-based supervised regression models that can be used as base algorithms of AVMs based on the finding of related works of literatures.

2.3.1. Decision Tree

A Decision Tree is an algorithm that automatically finds rules in data by creating tree-based classification rules (Song and Ying, 2015). Due to the nature of intuitiveness and easiness to understand, The Decision Tree algorithm has been widely used across several disciplines. Although the decision tree methodology can be used in both classification and regression modeling, the disadvantage of the decision tree is that the decision tree methodology cannot predict continuous values outside the range of the training data (Song and Ying, 2015). To solve the overfitting problems and their shortcomings in Decision Tree regression, an ensemble methodology was introduced. The ensemble methodologies aim to construct stronger learners through voting from multiple weak learners (Dietterich, 2000; Sagi and Rokach, 2018; Zhou, 2009). The ensemble algorithm is divided into bagging, boosting, and stacking according to the way they produce weak learners (individual trees).

2.3.2. Bagging

The bagging (Bootstrap Aggregating) models randomly create subsets from a given dataset and vote the results from weak learners trained from small datasets. The advantages of using bagging lie in the randomness and generalization (Oshiro et al., 2012). Random Forest, as a representative

algorithm of bagging, has a fast execution speed and shows high prediction performance in various areas (Qi, 2012). While Random Forest has been widely used in different fields, choosing the right number of trees to be used is still a challenge (Oshiro et al., 2012).

2.3.3. Boosting

While the bagging algorithm is focused on making a generalized model, boosting aims at solving a difficult problem to solve. The boosting algorithm iteratively updates several weak learners and optimizes it by weighting the incorrectly predicted data. Unlike bagging, boosting contains full instances per step in the training procedure (Quinlan, 1996). Bagging and boosting may have performance differences depending on the situation (Oshiro et al., 2012). Since subsets of training datasets are not generated while training in bagging, boosting can have an overfitting issue having a chance of affected by noise in a dataset (Oshiro et al., 2012).

2.4. Feature Selection in Machine Learning

The curse of dimensionality is a phenomenon in which the dimension of the data increases, the distance between the data increases, the sparsity of the empty space increases, and the performance of the model decreases (Indyk and Motwani, 1998). To solve this problem caused by using raw data of many dimensions, feature selection is critical in machine learning. Feature selection is a process of selecting variables to compose a model. By selecting useful features from data, it improves the chance to increase the predictive power and generalization performance of the model (Guyon and Elisseeff, 2003). Feature selection methods are largely classified into wrapper methods, filter methods, and embedded methods.

First, in wrapper methods, subsets of features are made and continuously tested through model prediction. The wrapper is a method to select the subset that shows the best performance in terms of prediction accuracy (Chandrashekar and Sahin, 2014). The disadvantage of the wrapper method is that it continuously trains and finds the optimal subset, therefore time and cost are very high (Khalid et al., 2014). Nevertheless, the feature selection is a desirable method for model performance because this method ultimately finds the best feature subset.

In feature selection using the filter methodology, the relevance between features is measured. Instead of giving the best feature subset, this method informs the user how much influence each feature has by giving a feature rank. Therefore, although the feature selection method may not provide the best feature subset for the model, but provides the additional information for feature selection (Sánchez et al., 2007). Also, all ranked features are considered independent variables.

Finally, embedded feature selection is a method that combines the advantages of filter and wrapper methodologies (Lal et al., 2006). Also, this method inserts feature selection into the machine

learning algorithm itself (Lal et al., 2006). Embedded methods learn features that contribute to the model's accuracy (Lal et al., 2006).

2.5. Feature Extraction

While feature selection chooses the best subset of the original pool of features, feature extraction transforms and generates new features from the original features (Gangurde, 2014). Feature extraction can be viewed as finding a set of vectors that represent the characteristics of the data in a reduced dimension (Lee and Landgrebe, 1993). By mapping and extracting features to another space that can better describe the features implicitly, feature extraction allows extracting of latent factors which were difficult for existing features to recognize. If features are extracted in the right manner, the newly reduced feature set summarizes most of the information from the original feature set.

Principal Components Analysis (PCA), as the most used linear dimensionality reduction technique, reduces the dimension by extracting the principal component, representing the results by using the correlation that exists between variables (Wold and Geladi, 1987). PCA finds the axis of the data with the highest variance and reduces the dimension to that axis, which is the principal component. Because the direction of a vector with the largest variance is the direction with the largest correlation between features and the direction containing the most information, the experiment can assume that effective dimensionality reduction occurs using PCA (Lever et al., 2017). However, limitations of PCA must be considered: the assumption of PCA presupposes a linear relationship between variables (Huang & McAvoy, 200). It can also be problematic when applying PCA to large datasets (Elgamal & Hefeeda, 2015).

Autoencoder is a nonlinear dimensionality reduction technique using artificial neural network structure to learn latent variables (Kramer, 1991). Autoencoder's neural network architecture in which the input layer and the output layer are the same, and then a hidden layer lies in the middle. Parameters of the hidden layers are optimized by an objective function that minimizes the reconstruction error of the input of the encoder and the output of the decoder (Hinton & Salakhutdinov, 2006). As a result, for dimensionality reduction purposes, only the network corresponding to the encoder part from the trained Autoencoder model passes the original data to obtain reduced data.

Clustering is mostly used to find hidden patterns of data where no label is available (Xu and Wunsch, 2005). Clustering aims to find a new set of categories by grouping similar instances into the same group and assigning them to different groups if instances are not similar (Rokach & Maimon, 2005).

From the above theoretical backgrounds, hypotheses for this research are formulated as below:

$H_{PCA:0}$*: There is no effect in using a PCA feature extraction on model performance.*

$H_{PCA:1}$*: There is a positive effect in using a PCA feature extraction on model performance.*

$H_{KMeans:0}$: There is no effect in using a K-Means-based feature extraction on model performance.

$H_{KMeans:1}$: There is a positive effect in using a K-Means-based feature extraction on model performance.

$H_{AE:0}$: There is no effect in using an Autoencoder feature extraction on model performance.

$H_{AE:1}$: There is a positive effect in using an Autoencoder feature extraction on model performance.

Where $H_{PCA:0}$ is the null hypothesis and $H_{PCA:1}$ is the alternative hypothesis in PCA feature extraction. $H_{KMeans:0}$ is the null hypothesis and $H_{KMeans:1}$ is the alternative hypothesis in K-Means feature extraction. $H_{AE:0}$ is the null hypothesis and $H_{AE:1}$ is the alternative hypothesis in Autoencoder feature extraction.

3. Methods

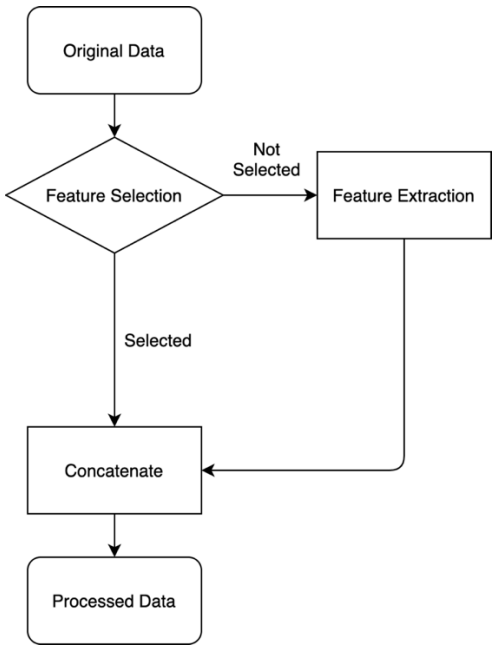


Figure 1: Data Processing Flow

The data processing flow is shown in Figure 1. The experiment was designed to observe the change in model performance according to applying various feature selection and feature extraction methods. To discover the effect, we use three baseline models: linear regression, Random Forest (as a representative of bagging ensemble model), and LightGBM (as a representative of boosting ensemble model). The three models were trained and compared on the original dataset including all features and 12 datasets in which vectors were concatenated with 20 common features selected by three selection methods and vectors reduced in dimension by four extraction methods. Detailed information on the 13 datasets for comparative analysis is presented in Table 1.

Dataset		# Features
Original Dataset		376
Feature Selection	**Feature Extraction**	
Wrapper Method	No Feature Extraction	20
	PCA	28
	K-Means	28
	Autoencoder	28
Filter Method	No Feature Extraction	20
	PCA	28
	K-Means	28
	Autoencoder	28
Embedded Method	No Feature Extraction	20
	PCA	28
	K-Means	28
	Autoencoder	28

Table 1: Datasets used for training

3.1. Dataset Description

	Train Dataset	Test Dataset
# Original Instances	30,471	7,662
# Processed Instances	30,101	7,662
# Original Features	291	291
# Processed Features	376	376
# Deleted Features	92	92

Table 2: Statistics of the dataset

	Sale Price (Train Dataset)
Mean	7,123,035.27
Std. Deviation	4,780,111.32
Min	100,000
25%	4,740,002
50%	6,274,411
75%	8,300,000
Max	111,111,112

Table 3: Statistics of the sale price (target variable)

In this section, this paper used Sberbank Russian Housing Market dataset to demonstrate the impact of different feature selection and extraction methods on housing price regression models (Sberbank, 2017). The Sberbank dataset originally contains 291 features describing individual transactions. The training data contains 30469 instances from August 2011 to June 2015, and the test set contains 7662 instances from July 2015 to May 2016. To handle missing values in the training dataset, features that contain more than 1,000 missing values were deleted, and remaining instances with missing values were also removed. Eventually, the experiment processed categorical features with one-hot representation. The result obtained a dataset of 377 features. Detailed information of data processing in train and test dataset is shown in Table 2 and Table 3.

3.2. Feature Selection Methods

Our experiment aims to solve the curse of dimensionality problem while delivering more information in the modeling process by effectively extracting and indirectly using features excluded from the selection process.

3.2.1. Wrapper Method

Recursive Feature Elimination (RFE) (Guyon et al., 2002) was used for the wrapper feature selection and implemented with the Scikit-learn library. The step parameter was set to 10, which then 10 features were removed at each iteration. For the external estimator, the base model set in each experiment was included. The process was repeated recursively to gradually remove less important features, leaving only the number of features specified in the experiment, in our case 20.

3.2.2. Filter Method

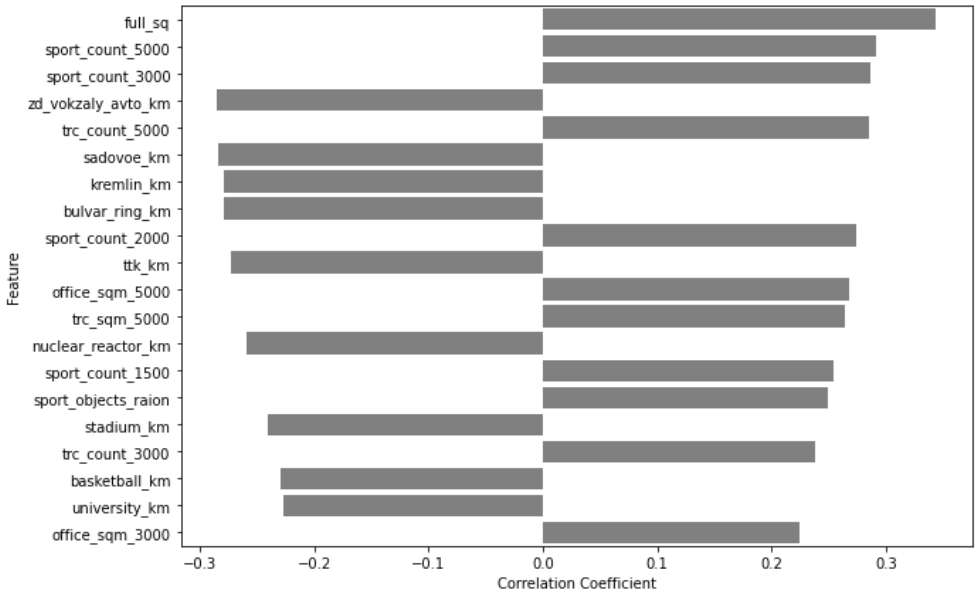


Figure 2: Correlation coefficients of selected variables (Top 20)

In the filter method, a feature with a high correlation coefficient is used after finding the correlation between features using a statistical measurement. In our experiment, the correlation coefficient with the target variable was calculated, and 20 items were selected and used in the order of greatest absolute value. Figure 2 shows the distribution of the correlation coefficient.

3.2.3. Embedded Method

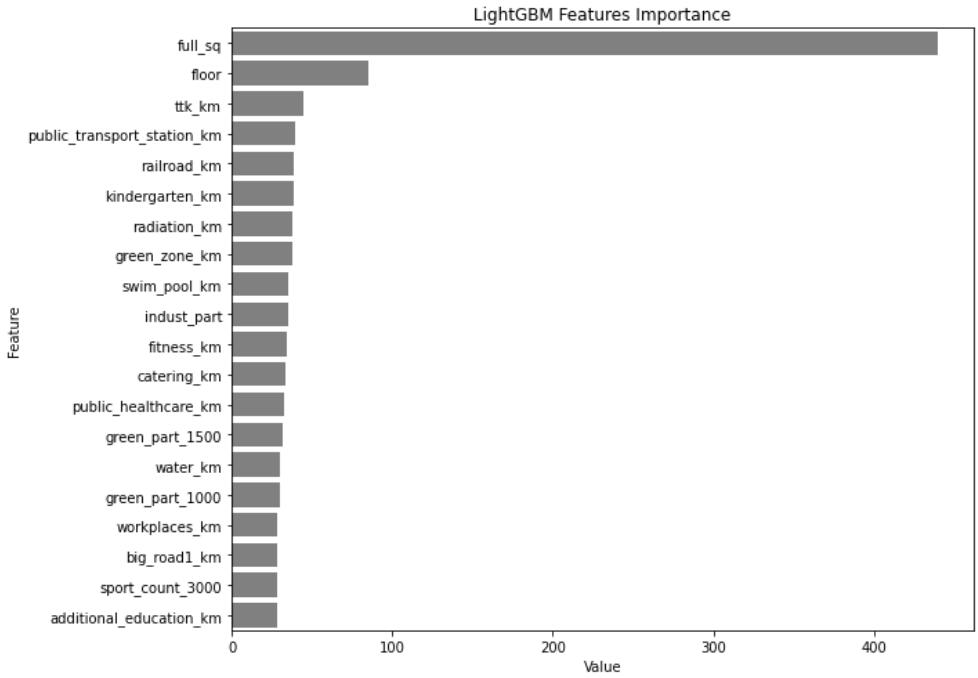


Figure 3: LightGBM feature importance of selected variables (Top 20)

To select features according to the embedded method, this study trained a tree-based model to find features that contribute to the model's accuracy. This research used the LightGBM (Ke et al., 2017) as a base model and trained the regression model including all features to calculate the feature importance. Figure 3 shows the top 20 features with their importance calculated from the built-in method of LightGBM.

3.3. Feature Extraction Methods

3.3.1. Principal Component Analysis

The researcher extracted latent vectors that can represent data well in low dimensions using the most representative dimensionality reduction technique which is Principal Component Analysis (PCA). PCA transforms samples in a high-dimensional space into a low-dimensional space without

linear correlation by finding new axes orthogonal to each other while preserving the variance of the data as much as possible.

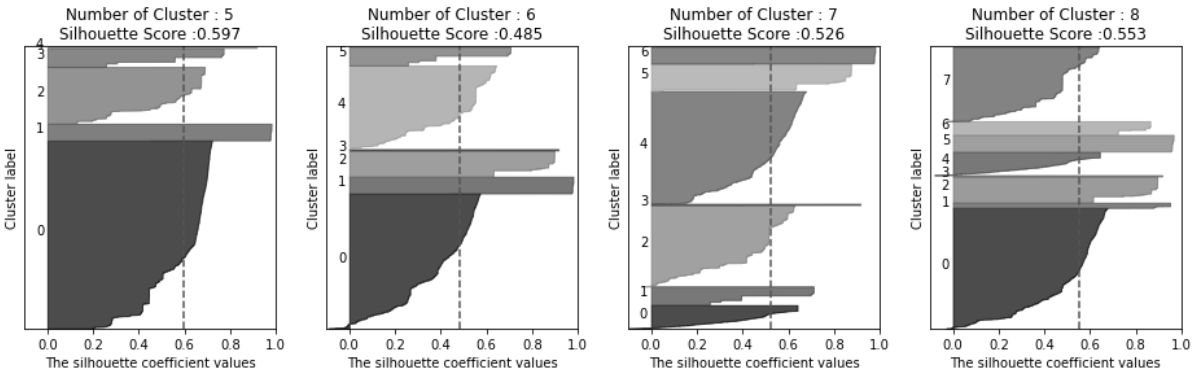
3.3.2. K-Means

In this experiment, the study assume that a cluster to which instance of data is assigned can provide compressed information of data. The assigned cluster values of features that are excluded in the selection process are combined with the selected features to be used for modeling.

In the K-Means clustering process, the initial centroid is randomly selected. For each data point, this clustering process is designated as the cluster to which the nearest centroid belongs (Poteraş et al., 2014). The K-Means algorithm is being optimized by minimizing the J function of the expression below:

$$J\left(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K\right) = \frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

Where $c^{(i)}$ denotes the cluster number that each data of $x^{(i)}$ were assigned, and this value becomes one of the number of clusters. μ_K is the centroid information of a cluster, and $\mu_{c^{(i)}}$ denotes the center point for the cluster to which the data of $x^{(i)}$ belongs to. The J function is the square of the distance between $x^{(i)}$ and $\mu_{c^{(i)}}$, and the objective of the K-Means algorithm is to minimize the distance between these two points.



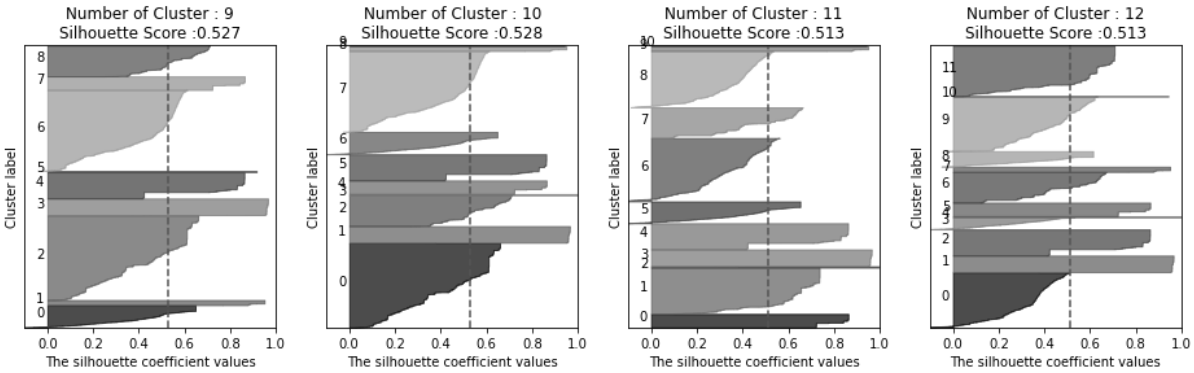


Figure 5: Silhouette analysis for K-Means clustering

In cluster analysis, selecting the right number of clusters is an important and challenging task (Wang, 2010). Although the analysis undergoes difficulty to accurately evaluate the performance of a model due to the nature of unsupervised learning, silhouette analysis is used as an effective method to evaluate the performance of clustering (Craenendonck and Blockeel, 2015). The silhouette analysis shows how efficiently the distances between each cluster are separated. The silhouette score has a value between 0 and 1, and the closer to 1, the better the clustering is evaluated. Figure 4 demonstrates the visualization of silhouette analysis for K-Means clustering on train data with the number of clusters range between 5 and 12. The experiment selected the number of clusters to 8 according to the visualized data in Figure 3. The silhouette score was the highest when the number of clusters was set to 5, but the experiment uses the number of clusters as 8 where the silhouette score was the second-highest and the clusters were more balanced.

3.3.3. Autoencoder

The Autoencoder compression model consists of encoder and decoder parts, where the encoder network provides a lower dimensional latent vector. The mini-batch is encoded through FC (356, 128, ReLU) – FC (128, 64, ReLU) – FC (64, 8) then decoded through FC (8, 64, ReLU) – FC (64, 128, ReLU) – FC (128, 356). The dropout ratio used in the network is 0.2. In the encoding process, only the encoder network is used to return the 8-dimensional latent vector.

Our Autoencoder instances are implemented by PyTorch (Paszke et al., 2019) and optimized by Adam (Kingma and Ba, 2015) with a learning rate of 0.0003. The number of training epochs and the size of mini-batches was 457 and 32, respectively.

3.4. Regression Models

3.4.1. Linear Regression

First, using the processed data, the research created a prediction model using the linear regression method, which is a basic model of machine learning. The formula for linear regression line can be written as follows:

$$\widehat{\mathrm{P}} = b_0 + b_1 \cdot Feature_{i1} + b_{i2} \cdot Feature_{i2} + \cdots + b_{i28} \cdot Feature_{i28} + \epsilon$$

Where $\widehat{\mathrm{P}}$ is the estimated price, ϵ is the error term, b_0 denotes the intercept, $b_1 \cdots b_{20}$, $Feature_{i1} \cdots Feature_{i20}$ denotes coefficients of each feature and 20 automatically selected features, respectively. Next, $b_{21} \cdots b_{28}$ is coefficients of each of 8 extracted features by different algorithms and $Feature_{i21} \cdots Feature_{i28}$ is 8 extracted features.

The optimal value of coefficients and intercepts of the regression formula can be found by minimizing the cost function. The cost function to minimize can be written as:

$$cost = \frac{1}{m}\sum_{i=1}^{m}\left(H\left(x^{(i)}\right) - y^{(i)}\right)$$

Where $H(x^{(i)})$ denotes the regression function, $y^{(i)}$ is the target value, and m is the number of data instances. The cost is defined as the average value of the difference between predicted value of random data x and the corresponding y when there are m different data sets.

3.4.2. Random Forest

The research used the random forest algorithm as another model. In the process of training the random forest, first, T training datasets are created through bootstrapping. Next, the study train T weak models and combine them using a mean or voting method. For this experiment, the Random Forest Regressor implemented in Scikit-learn was used. For the configuration setting of Random Forest, the default values provided by Scikit-learn were used.

3.4.3 LightGBM

LightGBM supplements for the slow training time problem of the existing boosting algorithm (Ke et al., 2017). In the process of training the LightGBM, first, the model makes the initial predictions. Then, the loss is calculated by the error with the actual value, sorted by loss, and the top N items are selected to store. Only a part of the remaining data is randomly sampled, and the data is resampled with a high gradient of 30% compared to the total data and a small gradient of 10% of the total data. Finally, a weight is applied to the reduced small gradient, and a weak predictor is created using the entire dataset,

loss, and weight for the reduced data, and added to the entire predictor set. For this experiment, the LightGBM Regressor implemented in LightGBM Python API was used. For the configuration setting of LightGBM, the default values provided were used.

3.5. Training Strategy

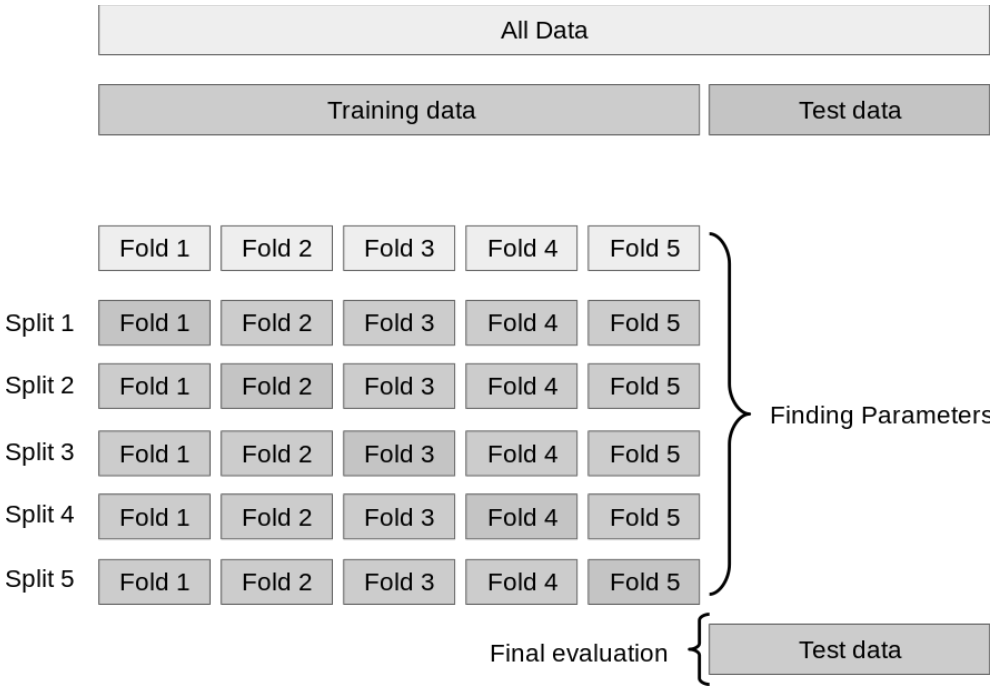


Figure 4: Explanation of K-Fold Cross Validation (Pedregosa et al.)

The K-fold cross-validation technique was utilized for model selection and error estimation. Figure 4 describes how cross-validation works in the training process. The model is trained K times on each split data, and the training status is monitored by averaging the scores for the evaluation metrics calculated K times. The number of K in this experiment was set to 5.

For the reproduction of the experiment, the value of all random seeds was fixed to 42.

3.6. Evaluation Methods

The experiment evaluated the model by uploading the prediction values for the test dataset with 7662 instances to Kaggle's system and calculating the official metric of the leaderboard, Root Mean Squared Log Error (RMSLE). The formula for calculating RMSLE is as follows:

$$RMSLE = \sqrt{\frac{1}{N}\sum_{N}^{i=1}\left(log(y_i + 1) - log(\hat{y}_i + 1)\right)^2}$$

Where N denotes the number of samples, y_i denotes the true value, and $\hat{y}_i$ denotes the predicted value. A model with a smaller RMSLE is a model with better estimation performance.

4. Results

4.1. Feature Selection

Table 4 lists the variables selected for each feature selection method which were reflected in the training dataset. The features selected by all three methods include 'full_sq', 'sport_count_3000', and 'ttk_km'. The set of features selected by the wrapper and embedded methods were most similar, and they contain the following variables: 'additional_education_km', 'big_road1_km', 'catering_km', 'floor', 'full_sq', 'green_part_1500', 'green_zone_km', 'kindergarten_km', 'public_healthcare_km', 'public_transport_station_km', 'radiation_km', 'railroad_km', 'sport_count_3000', 'swim_pool_km', 'ttk_km', 'water_km'.

Method	Subset of Selected Features
Wrapper Method	'additional_education_km', 'big_road1_km', 'catering_km', 'floor', 'full_sq', 'green_part_1500', 'green_zone_km', 'industrial_km', 'kindergarten_km', 'office_sqm_5000', 'oil_chemistry_km', 'power_transmission_line_km', 'public_healthcare_km', 'public_transport_station_km', 'radiation_km', 'railroad_km', 'sport_count_3000', 'swim_pool_km', 'ttk_km', 'water_km'
Filter Method	'basketball_km', 'bulvar_ring_km', 'full_sq', 'kremlin_km', 'nuclear_reactor_km', 'office_sqm_3000', 'office_sqm_5000', 'sadovoe_km', 'sport_count_1500', 'sport_count_2000', 'sport_count_3000', 'sport_count_5000', 'sport_objects_raion', 'stadium_km', 'trc_count_3000', 'trc_count_5000', 'trc_sqm_5000', 'ttk_km', 'university_km', 'zd_vokzaly_avto_km'
Embedded Method	'additional_education_km', 'big_road1_km', 'catering_km', 'fitness_km', 'floor', 'full_sq', 'green_part_1000', 'green_part_1500', 'green_zone_km', 'indust_part', 'kindergarten_km', 'public_healthcare_km', 'public_transport_station_km', 'radiation_km', 'railroad_km', 'sport_count_3000', 'swim_pool_km', 'ttk_km', 'water_km', 'workplaces_km'

Table 4: Selected Features

4.2. Feature Extraction

When the feature extraction method using PCA, K-Means, and Autoencoder was applied, the converted data according to the extraction method showed different vector representations in a latent

space. For visualization, the result performed three-dimensional extraction, instead of eight-dimensional extraction used in the main experiment. The results are included in the appendix (Figure 5, Figure 6, and Figure 7).

4.3. Test Set Evaluation

Table 5 shows the private RMSLE score from Kaggle results obtained by fitting 13 datasets including the original dataset to three different base models.

Dataset		**Linear Regression**	**Random Forest**	**LightGBM**
Original Dataset		0.81583	0.33617	0.33011
Feature Selection	**Feature Extraction**			
Wrapper Methods	No Feature Extraction	0.41580	0.33605	0.32494
	PCA	0.41444	0.32817	0.32608
	K-Means	0.42679	0.33613	0.32515
	Autoencoder	0.41362	0.33017	0.32522
Filter Method	No Feature Extraction	0.40262	0.34416	0.32711
	PCA	0.42417	0.3311	0.32843
	K-Means	0.40315	0.34451	0.32861
	Autoencoder	0.40931	0.33175	0.33089
Embedded Method	No Feature Extraction	0.4133	0.33513	0.32532
	PCA	0.41622	0.32839	0.32506
	K-Means	0.40897	0.33503	0.32545
	Autoencoder	0.4151	0.33026	0.32838

Table 5: RMSLE Results

In all 13 datasets which were processed differently, the tree-based nonlinear models (Random Forest and LightGBM) significantly performed better than the linear regression model. Also, the boosting-based LightGBM performed better than the random forest which is using the bagging methodology. However, the difference in RMSLE between Random Forest and LightGBM was not significant that the decrease was in two decimal places.

Overall, the performance improvement through dimensionality reduction was significant. The effect of dimensionality reduction was most prominent when linear regression was used as a base model, showing a decrease to around 50%. In the linear regression model, the results showed the best performance when features were selected by the filter method, and in the Random Forest model, which gave the best performance when features were selected by the embedded method. Also, the LightGBM model showed the best performance when the wrapper method was used.

4.4. Hypothesis Testing in Linear Regression

OLS Regression Results			
Dep. Variable	y	**R-squared**	0.229
Model	OLS	**Adj. R-squared**	0.229
Method	Least Squares	**F-statistic**	357.8
Date	Sun, 27 Jun 2021	**Prob (F-statistic)**	0
Time	10:40:55	**Log-Likelihood**	-4.01E+05
No. Observations	24081	**AIC**	8.01E+05
Df Residuals	24060	**BIC**	8.01E+05
Df Model	20	**Durbin-Watson**	1.965
Covariance Type	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	2.50E+06	3.81E+05	6.549	0	1.75E+06	3.24E+06
x1	3.48E+04	636.886	54.626	0	3.35E+04	3.60E+04
x2	-1.01E+04	3321.967	-3.047	0.002	-1.66E+04	-3609.458
x3	5.05E+04	6646.467	7.6	0	3.75E+04	6.35E+04
x4	1.39E+04	1.42E+04	0.975	0.329	-1.40E+04	4.18E+04
x5	-8991.0207	5123.39	-1.755	0.079	-1.90E+04	1051.145
x6	-2.33E+06	1.09E+05	-21.283	0	-2.54E+06	-2.11E+06
x7	1.43E+06	9.53E+04	14.965	0	1.24E+06	1.61E+06
x8	6.27E+05	1.14E+05	5.513	0	4.04E+05	8.50E+05
x9	3.43E+04	1.27E+04	2.709	0.007	9473.191	5.91E+04
x10	2.73E+05	5.05E+04	5.395	0	1.73E+05	3.71E+05
x11	0.6212	0.052	11.931	0	0.519	0.723
x12	-0.5408	0.062	-8.692	0	-0.663	-0.419
x13	-1.22E+05	9854.283	-12.425	0	-1.42E+05	-1.03E+05
x14	-1.76E+04	1.39E+04	-1.272	0.204	-4.48E+04	9543.405

x15	6.49E+04	7011.227	9.251	0	5.11E+04	7.86E+04
x16	9692.4951	1.12E+04	0.864	0.388	-1.23E+04	3.17E+04
x17	1.75E+04	7402.026	2.367	0.018	3009.02	3.20E+04
x18	-615.572	2.06E+04	-0.03	0.976	-4.10E+04	3.98E+04
x19	-1.61E+04	7831.566	-2.054	0.04	-3.14E+04	-733.742
x20	-0.6019	0.099	-6.067	0	-0.796	-0.407

Table 6: OLS Regression Results – No Feature Extraction

To find out the effect of each feature extraction method, this paper focused on feature selection in the filter method and performed OLS regression analysis on each training dataset. Table 6 shows the results of OLS regression model using only 20 variables to which feature extraction is not applied. In this part, x1 to x20 represent the selected variables, and x21 to x28 represent the extracted latent independent variables for the remaining features. The linear regression results for the three feature extractions we experimented with are compared with Table 6 to test the hypothesis.

4.4.1. Alternative Hypothesis for PCA Feature Extraction

OLS Regression Results			
Dep. Variable	y	**R-squared**	0.26
Model	OLS	**Adj. R-squared**	0.259
Method	Least Squares	**F-statistic**	302.3
Date	Sun, 27 Jun 2021	**Prob (F-statistic)**	0
Time	10:43:23	**Log-Likelihood**	-4.00E+05
No. Observations	24081	**AIC**	8.00E+05
Df Residuals	24052	**BIC**	8.00E+05
Df Model	28	**Durbin-Watson**	1.988
Covariance Type	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.93E+06	4.20E+05	4.59	0	1.10E+06	2.75E+06
x1	3.32E+04	627.506	52.979	0	3.20E+04	3.45E+04
x2	4037.2744	3413.125	1.183	0.237	-2652.664	1.07E+04
x3	4.47E+04	6611.448	6.766	0	3.18E+04	5.77E+04
x4	-6.27E+04	1.46E+04	-4.309	0	-9.12E+04	-3.42E+04
x5	1.33E+04	5205.912	2.545	0.011	3046.525	2.35E+04
x6	-2.32E+06	1.23E+05	-18.8	0	-2.56E+06	-2.08E+06

x7	1.10E+06	1.48E+05	7.454	0	8.13E+05	1.39E+06
x8	1.25E+06	1.36E+05	9.183	0	9.81E+05	1.51E+06
x9	3.12E+04	1.24E+04	2.513	0.012	6871.113	5.56E+04
x10	-4.32E+04	5.96E+04	-0.724	0.469	-1.60E+05	7.37E+04
x11	0.4784	0.053	9.062	0	0.375	0.582
x12	-0.5374	0.061	-8.743	0	-0.658	-0.417
x13	-1.29E+05	1.02E+04	-12.624	0	-1.49E+05	-1.09E+05
x14	-6822.3444	1.36E+04	-0.501	0.616	-3.35E+04	1.99E+04
x15	4.50E+04	9147.362	4.917	0	2.70E+04	6.29E+04
x16	8.56E+04	1.19E+04	7.214	0	6.23E+04	1.09E+05
x17	2.31E+04	7890.631	2.93	0.003	7656.742	3.86E+04
x18	-6.33E+04	2.13E+04	-2.979	0.003	-1.05E+05	-2.17E+04
x19	-6.55E+04	8176.239	-8.011	0	-8.15E+04	-4.95E+04
x20	-0.3185	0.1	-3.189	0.001	-0.514	-0.123
x21	-8.60E+05	5.79E+04	-14.847	0	-9.74E+05	-7.47E+05
x22	-7.03E+05	1.20E+05	-5.864	0	-9.38E+05	-4.68E+05
x23	1.03E+06	5.60E+04	18.348	0	9.18E+05	1.14E+06
x24	-2.47E+05	5.47E+04	-4.522	0	-3.54E+05	-1.40E+05
x25	-5.88E+05	4.33E+04	-13.562	0	-6.72E+05	-5.03E+05
x26	-9.59E+05	5.53E+04	-17.329	0	-1.07E+06	-8.50E+05
x27	-9.87E+05	7.00E+04	-14.104	0	-1.12E+06	-8.50E+05
x28	-6.04E+05	5.44E+04	-11.118	0	-7.11E+05	-4.98E+05

Table 7: OLS Regression Results – PCA Feature Extraction

Multivariate regression analysis was performed to analyze the effect of 8 variables added after PCA feature extraction on the real estate transaction price. Before analyzing the results, Durbin Watson was 1.988, close to 2, which can be regarded as a suitable multivariate regression model. From Table 7, every added variable (from x21 to x28) have a significant influence on the target variable ($P < 0.05$). The R-squared value was found to be 0.26, which was higher than the R-squared value in the model in which feature extraction was not performed (0.229). From the above statistics, the results reject the null hypothesis and accept the alternative hypothesis in PCA feature extraction.

4.4.2. Alternative Hypothesis for K-Means Feature Extraction

OLS Regression Results			
Dep. Variable	y	**R-squared**	0.232

Model	OLS	**Adj. R-squared**	0.231
Method	Least Squares	**F-statistic**	269.1
Date	Sun, 27 Jun 2021	**Prob (F-statistic)**	0
Time	10:46:32	**Log-Likelihood**	-4.00E+05
No. Observations	24081	**AIC**	8.01E+05
Df Residuals	24053	**BIC**	8.01E+05
Df Model	27	**Durbin-Watson**	1.965
Covariance Type	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	2.24E+06	3.69E+05	6.066	0	1.52E+06	2.96E+06
x1	3.48E+04	636.414	54.613	0	3.35E+04	3.60E+04
x2	-8782.5034	3350.187	-2.621	0.009	-1.53E+04	-2215.927
x3	4.41E+04	6761.614	6.526	0	3.09E+04	5.74E+04
x4	7.62E+04	1.72E+04	4.429	0	4.25E+04	1.10E+05
x5	-1.17E+04	5161.792	-2.268	0.023	-2.18E+04	-1590.889
x6	-2.49E+06	1.16E+05	-21.485	0	-2.71E+06	-2.26E+06
x7	1.52E+06	1.05E+05	14.486	0	1.31E+06	1.72E+06
x8	5.54E+05	1.22E+05	4.543	0	3.15E+05	7.92E+05
x9	3.58E+04	1.26E+04	2.83	0.005	1.10E+04	6.06E+04
x10	3.71E+05	5.48E+04	6.775	0	2.64E+05	4.78E+05
x11	0.6334	0.053	11.98	0	0.53	0.737
x12	-0.5342	0.063	-8.472	0	-0.658	-0.411
x13	-1.31E+05	1.01E+04	-13.053	0	-1.51E+05	-1.12E+05
x14	-1.83E+04	1.40E+04	-1.309	0.191	-4.56E+04	9089.574
x15	6.20E+04	7949.105	7.797	0	4.64E+04	7.76E+04
x16	-2.53E+04	1.21E+04	-2.095	0.036	-4.89E+04	-1623.958
x17	1.81E+04	7623.903	2.372	0.018	3143.862	3.30E+04
x18	2.54E+04	2.34E+04	1.085	0.278	-2.05E+04	7.12E+04
x19	-2.40E+04	8522.685	-2.81	0.005	-4.07E+04	-7243.908
x20	-0.5764	0.099	-5.804	0	-0.771	-0.382
x21	4.28E+05	1.08E+05	3.965	0	2.16E+05	6.39E+05
x22	7.60E+04	1.25E+05	0.608	0.543	-1.69E+05	3.21E+05
x23	6.38E+05	1.13E+05	5.639	0	4.16E+05	8.60E+05
x24	-1.16E+05	4.08E+05	-0.285	0.775	-9.16E+05	6.83E+05
x25	-8.55E+05	1.56E+05	-5.489	0	-1.16E+06	-5.50E+05

x26	5.92E+05	1.12E+05	5.265	0	3.72E+05	8.13E+05
x27	1.01E+06	1.99E+05	5.069	0	6.18E+05	1.40E+06
x28	4.71E+05	1.12E+05	4.204	0	2.51E+05	6.90E+05

Table 8: OLS Regression Results – K-Means Feature Extraction

Table 8 shows the result of multivariate regression analysis on the effect of 8 variables added after the K-Means feature. Added variables (from x21 to x28) are the one-hot representation of the assigned cluster. Before analyzing the results, Durbin Watson was 1.965, close to 2, therefore our model can be regarded as a suitable multivariate regression model. From Table 8, six variables (x21, x23, x25, x26, x27, and x28) have a significant influence on the target variable ($P < 0.05$). On the other hand, p-values of x22 and x24 are larger than 0.05, therefore those two variables have no significant effect on the target variable. The R-squared value was found to be 0.232, which was higher than the R-squared value in the model in which feature extraction was not performed (0.229). From the above statistics, the results reject the null hypothesis and accept the alternative hypothesis in PCA feature extraction.

4.4.3. Alternative Hypothesis for Autoencoder Feature Extraction

OLS Regression Results			
Dep. Variable	y	**R-squared**	0.234
Model	OLS	**Adj. R-squared**	0.233
Method	Least Squares	**F-statistic**	261.8
Date	Sun, 27 Jun 2021	**Prob (F-statistic)**	0
Time	10:49:56	**Log-Likelihood**	-4.00E+05
No. Observations	24081	**AIC**	8.01E+05
Df Residuals	24052	**BIC**	8.01E+05
Df Model	28	**Durbin-Watson**	1.967
Covariance Type	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.60E+06	6.30E+05	2.542	0.011	3.66E+05	2.84E+06
x1	3.45E+04	636.992	54.15	0	3.32E+04	3.57E+04
x2	-8107.1878	3391.024	-2.391	0.017	-1.48E+04	-1460.568
x3	5.41E+04	6724.474	8.038	0	4.09E+04	6.72E+04
x4	1.01E+04	1.48E+04	0.681	0.496	-1.89E+04	3.91E+04
x5	-5556.0882	5178.793	-1.073	0.283	-1.57E+04	4594.671
x6	-2.39E+06	1.16E+05	-20.532	0	-2.62E+06	-2.16E+06

x7	1.47E+06	1.00E+05	14.626	0	1.27E+06	1.66E+06
x8	5.98E+05	1.18E+05	5.045	0	3.65E+05	8.30E+05
x9	3.11E+04	1.26E+04	2.457	0.014	6286.327	5.58E+04
x10	3.26E+05	5.39E+04	6.048	0	2.20E+05	4.31E+05
x11	0.6112	0.053	11.544	0	0.507	0.715
x12	-0.5582	0.063	-8.829	0	-0.682	-0.434
x13	-1.41E+05	1.03E+04	-13.652	0	-1.61E+05	-1.20E+05
x14	-2.21E+04	1.39E+04	-1.584	0.113	-4.93E+04	5240.376
x15	4.72E+04	8208.107	5.751	0	3.11E+04	6.33E+04
x16	9106.9091	1.14E+04	0.798	0.425	-1.33E+04	3.15E+04
x17	6186.3705	7717.9	0.802	0.423	-8941.197	2.13E+04
x18	3.08E+04	2.14E+04	1.441	0.149	-1.11E+04	7.26E+04
x19	-2.61E+04	8286.758	-3.147	0.002	-4.23E+04	-9837.367
x20	-0.5852	0.1	-5.823	0	-0.782	-0.388
x21	-2.94E+06	2.57E+06	-1.146	0.252	-7.98E+06	2.09E+06
x22	8.43E+06	3.13E+06	2.689	0.007	2.29E+06	1.46E+07
x23	-9.32E+06	3.17E+06	-2.938	0.003	-1.55E+07	-3.10E+06
x24	7.46E+06	3.39E+06	2.198	0.028	8.07E+05	1.41E+07
x25	1.52E+06	4.07E+06	0.375	0.708	-6.45E+06	9.49E+06
x26	-3.03E+06	3.19E+06	-0.951	0.341	-9.27E+06	3.21E+06
x27	-2.47E+07	2.99E+06	-8.246	0	-3.05E+07	-1.88E+07
x28	5.62E+05	3.20E+06	0.176	0.861	-5.71E+06	6.83E+06

Table 9: OLS Regression Results – Autoencoder Feature Extraction

Multivariate regression analysis was performed to analyze the effect of 8 variables added after Autoencoder feature extraction on the real estate transaction price. Before analyzing the results, Durbin Watson was 1.967, close to 2, therefore it is regarded as a suitable multivariate regression model. From Table 9, only half of the added variables (x22, x23, x24, and x27) were found to be significant ($P < 0.05$). According to Table 9, the R-squared value was 0.234, which is seen as a model with better explanatory power than the null hypothesis (0.229), but since half of the added variables are not significant, the research do not reject the null hypothesis and do not support the alternative hypothesis.

5. Discussion

This study presented a comparative analysis of applying different feature extraction methods on the real estate valuation model. From the models presented in this study, about 300 features from original data were selected with an automated algorithm. Also, the performance change was observed

as more information was provided to the model by extracting the features excluded in the feature selection step.

Through experiments, this study confirmed that the nonlinear modeling method is superior to the linear regression model in predictive power, which is in line with the findings of Limsombunchai (2004). In the case of the linear regression-based model, the number of features was greatly affected. Therefore, feature selection played a significant role in linear regression. However, in the case of Random Forest or LightGBM-based models, although the performance increased through feature selection, it provided good performance even when all 300 features were used. This implies that tree-based models can perform modeling with high predictive power by considering more information.

Our experiment is meaningful in selecting useful features and extracting meaningful information without any human intervention in data processing phase. Since linearity between variables must be considered in linear regression, the best effect was obtained when the top 20 were selected in the order of correlation coefficient. To test the hypothesis, this research investigated the effect of 8-dimensional features added by applying feature extraction methods to the linear regression model. When our proposed methods were applied, the explanatory power of the model was improved. This result is in line with the previous findings of Lee and Landgrebe (1993).

From the experimental results of this study, the first limitation lies in that it is difficult to choose the best feature extraction method. When applying feature extraction to a new model that has not been tested above, the research faces difficulty to know the optimal extraction method to use. This problem has disadvantages in terms of time and computational cost because additional experiments must be carried out every time when a new model is developed.

Another limitation is related to the number of dimensions set in the feature extraction process. In this paper, to compare the performance of the feature extraction methods, different conditions were fixed as much as possible. This paper set the latent dimension of the output of feature extraction to 8 because this paper set the number of clusters of K-Means to 8. However, there is a possibility that 8 is not the optimal output dimension for other feature extraction methods. The unification of the number of all latent dimensions in our experiment was for comparative analysis between methodologies, therefore, the results of this research recommend future practitioners find the optimal latent dimension for each through experiments for better performance.

6. Conclusion

In the experiments of this study, different machine learning algorithms were tested with different feature selection and extraction methodologies. As a result, like the results of previous studies, tree-based algorithms such as Random Forest and LightGBM showed better prediction performance than linear regression. Performance change by applying feature extraction differed depending on the feature selection method and the model used. As a result of hypothesis testing through regression

analysis, this research rejected the null hypothesis for PCA and K-Means and did not reject the null hypothesis for Autoencoder.

In future research, the result is anticipated that the predictive performance of the model can be improved by introducing state-of-the-art machine learning algorithms to AVMs. In addition to the automated feature extraction method proposed in our study, various feature engineering using domain knowledge can be applied. Finally, the future research will also be an important follow-up study to find an appropriate number of latent dimensions, which has been pointed out as a limitation in our experiments.

References

Buchanan, B. G. (2005). A (very) brief history of artificial intelligence. *Ai Magazine*, 26(4), 53-53.

Chandrashekar, G., and Sahin, F. (2014). A survey on feature selection methods. *Computers and. Electrical Engineering, 40*(1), 16-28.

Chau, K. W., and Chin, T. L. (2003). A critical review of literature on the hedonic price model. *International Journal for Housing Science and Its Applications*, 27(2), 145-165.

Court, A. T. (1939), Hedonic Price Indexes with Automotive Examples. *The Dynamics of Automobile. Demand*, 98-119.

Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on. multiple classifier systems* (pp. 1-15). Springer, Berlin, Heidelberg.

Elgamal, T., & Hefeeda, M. (2015). Analysis of PCA algorithms in distributed environments. *arXiv. preprint arXiv:1503.05214*.

Gangurde, H. D. (2014). Feature selection using clustering approach for big data. *Int. J. Comput. Appl*, 975, 1-3.

Guyon, I., and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of. machine learning research*, 3(Mar), 1157-1182.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1), 389-422.

Hinton, G. E., and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural. networks. *science*, 313(5786), 504-507.

Huang, Y., Gertler, J., & McAvoy, T. J. (2000). Sensor and actuator fault isolation by structured partial. PCA with nonlinear extensions. *Journal of Process Control, 10*(5), 459-469.

Indyk, P., and Motwani, R. (1998, May). Approximate nearest neighbors: towards removing the curse. of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (pp. 604-613).

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... and Liu, T. Y. (2017). Lightgbm: A. highly efficient gradient boosting decision tree. *Advances in neural information processing systems, 30*, 3146-3154.

Khalid, S., Khalil, T., and Nasreen, S. (2014, August). A survey of feature selection and feature. extraction techniques in machine learning. In *2014 science and information conference* (pp. 372-378). IEEE.

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint. arXiv:1412.6980.*

Kok, N., Koponen, E. L., and Martínez-Barbosa, C. A. (2017). Big data in real estate? From manual. appraisal to automated valuation. *The Journal of Portfolio Management*, 43(6), 202-211.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2), 233-243.

Lal, T. N., Chapelle, O., Weston, J., and Elisseeff, A. (2006). Embedded methods. In *Feature extraction.* (pp. 137-165). Springer, Berlin, Heidelberg.

Lee, C., and Landgrebe, D. A. (1993). Feature extraction based on decision boundaries. *IEEE. Transactions on Pattern Analysis and Machine Intelligence*, 15(4), 388-400.

Lever, J., Krzywinski, M., and Altman, N. (2017). Points of significance: Principal component. analysis.

Limsombunchai, V. (2004). House price prediction: hedonic price model vs. artificial neural network. In *New Zealand agricultural and resource economics society conference* (pp. 25-26).

Manganelli, B. (2015). *Real estate investing*. Springer.

Oshiro, T. M., Perez, P. S., and Baranauskas, J. A. (2012, July). How many trees in a random forest?. In *International workshop on machine learning and data mining in pattern recognition* (pp. 154-168). Springer, Berlin, Heidelberg.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703.*

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.

Poteraş, C. M., Mihăescu, M. C., and Mocanu, M. (2014, September). An optimized version of the K-Means clustering algorithm. In *2014 Federated Conference on Computer Science and Information Systems (pp. 695-699)*. IEEE.

Qi, Y. (2012). Random forest for bioinformatics. In *Ensemble machine learning* (pp. 307-323). Springer, Boston, MA.

Quinlan, J. R. (1996, August). Bagging, boosting, and C4. 5. In *Aaai/iaai, Vol. 1* (pp. 725-730).

Reilly, F. and K. Brown, *Investment Analysis and Portfolio Management,* Sixth edition, Dryden Press, 2000.

Rokach, L., and Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery. handbook* (pp. 321-352). Springer, Boston, MA.

Sagi, O., and Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data. Mining and Knowledge Discovery*, 8(4), e1249.

Sánchez-Marono, N., Alonso-Betanzos, A., and Tombilla-Sanromán, M. (2007, December). Filter. methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 178-187). Springer, Berlin, Heidelberg.

Sberbank. (2017, April). Sberbank Russian Housing Market, Retrieved May 3rd, 2021 from. https://www.kaggle.com/c/sberbank-russian-housing-market/data.

Selim, H. (2009). Determinants of house prices in Turkey: Hedonic regression versus artificial neural. network. *Expert systems with Applications*, 36(2), 2843-2852.

Song, Y. Y., and Ying, L. U. (2015). Decision tree methods: applications for classification and. prediction. *Shanghai archives of psychiatry, 27*(2), 130.

Valier, A. (2020). Who performs better? AVMs vs hedonic models. *Journal of Property Investment. and Finance.*

Van Craenendonck, T., and Blockeel, H. (2015). Using internal validity measures to compare clustering. algorithms. *Benelearn 2015 Poster presentations (online)*, 1-8.

Van Doorn, J., Verhoef, P. C., and Bijmolt, T. H. (2007). The importance of non-linear relationships. between attitude and behaviour in policy research. *Journal of consumer policy*, 30(2), 75-90.

Wang, J. (2010). Consistent selection of the number of clusters via crossvalidation. *Biometrika*, 97(4), 893-904.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and. intelligent laboratory systems*, 2(1-3), 37-52.

Xu, R., and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural. networks*, 16(3), 645-678.

Zhou, Z. H. (2009). Ensemble learning. Encyclopedia of biometrics, 1, 270-273.

Appendix

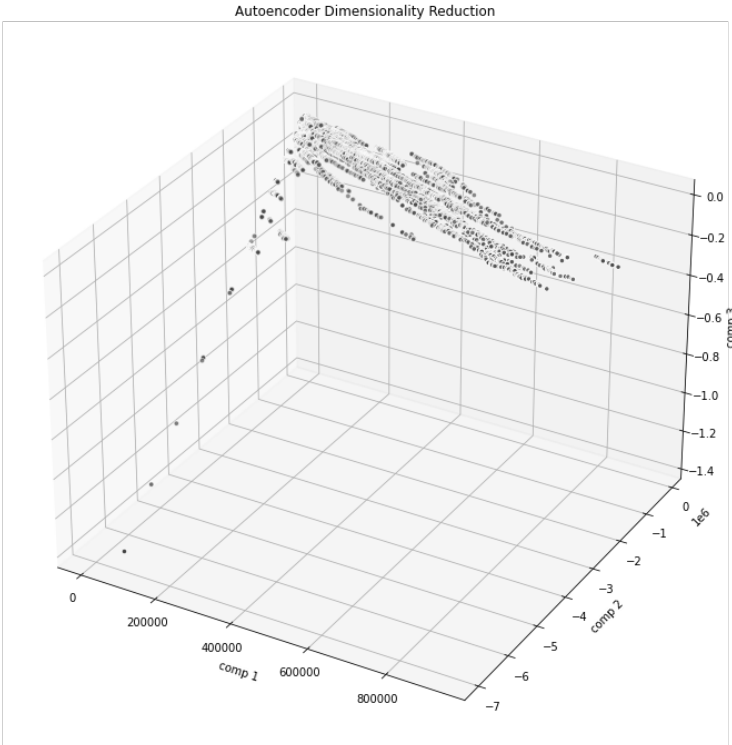


Figure 5: Example of Low-Dimensional Representation - Autoencoder

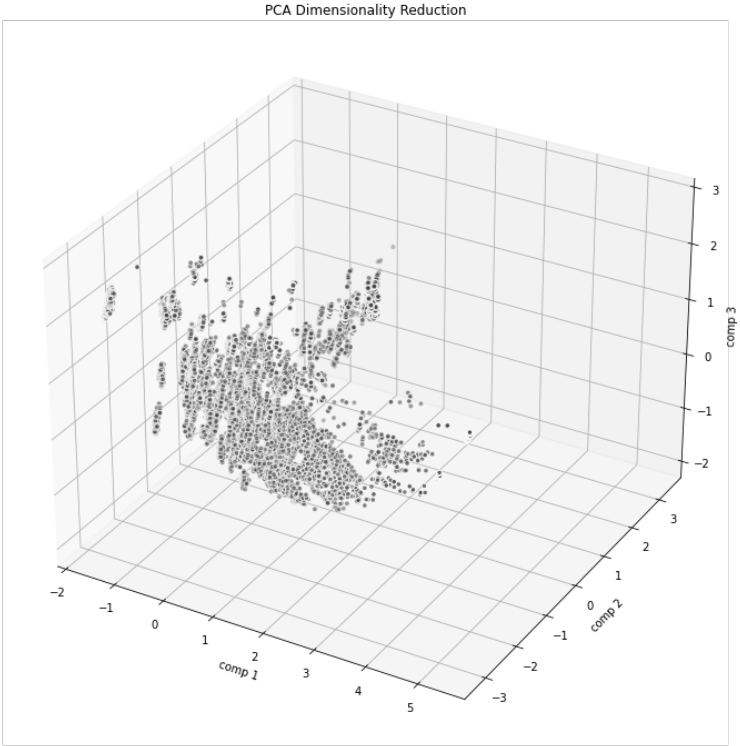


Figure 6: Example of Low-Dimensional Representation - PCA

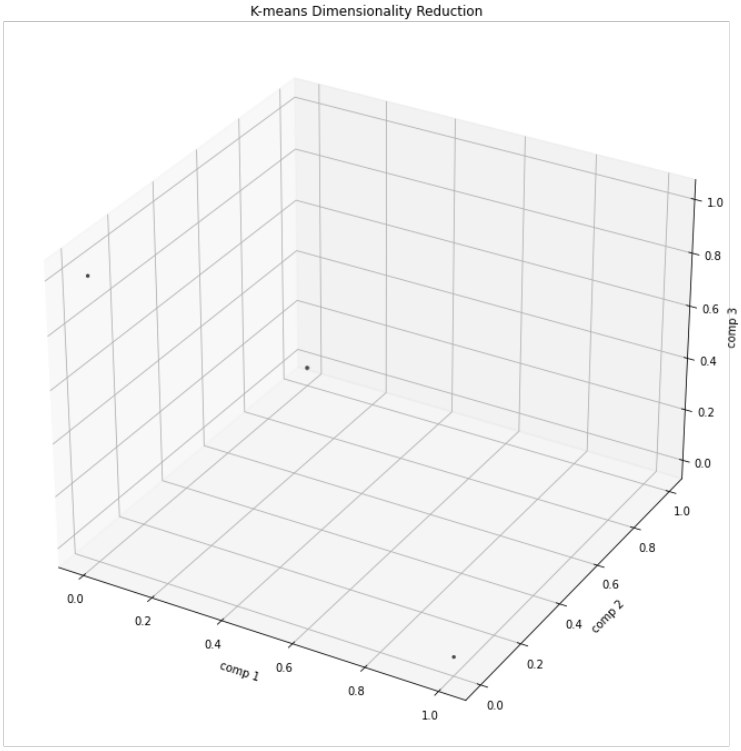


Figure 7: Example of Low-Dimensional Representation - K-Means