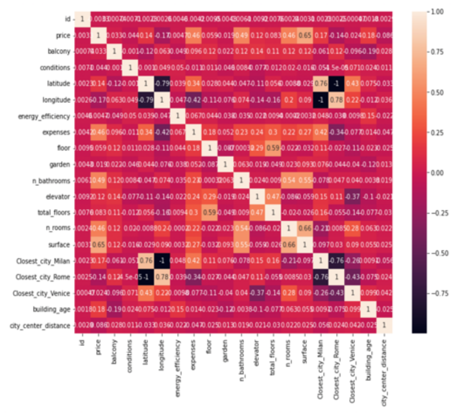
Eren Karsavuranoğlu
Student ID: 3164647
Kaggle Username: erenkarsavuranolu

MACHINE LEARNING CHALLENGE REPORT

INTRODUCTION

The purpose of this project is to use a dataset to estimate the prices of houses in Italy. The dataset contains information on various factors such as location, surface, and number of rooms, which are formed as columns in the dataset. The objective of this project was achieved by a regression model. The model was trained to predict the house prices based on the available features; however, the dataset was cleaned and manipulated as a first step. Then, the categorical variables were transformed into numeric variables to utilize the regression model. All the significant variables with missing values were imputed by training the data. Lastly, the regression model was applied.

PREPROCESSING: DATA CLEANING, MANIPULATION, AND FEATURE ENGINEERING

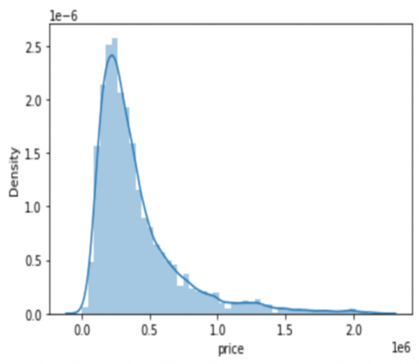


Firstly, the columns of the dataset were investigated. The correlation matrix and the heatmap showed the overall relationship among the variables. Each feature was analyzed to comprehend their significance and relevance for the target variable 'price'. As a result of the heatmap, the 'plot.scatter', 'subplots', and 'groupby' methods were utilized to make a deeper analysis which can be found within the codes in the zip folder. As a result of the first analysis and intuition, surface, number of rooms, number of bathrooms, and location seemed as the most strongly correlated features towards the target variable.

The missing values were identified, and their corresponding percentages were calculated to assess the relative impact of each feature with missing data. In order to apply linear regression to the data, it is imperative to handle missing values and encode categorical variables into numerical ones. Therefore, it is necessary to perform data preprocessing steps to address these issues prior to training the model.

The missing values were filled with 0 for variables 'garden' and 'balcony', because there were only 'True' entries, and the rest was missing. For the variable 'elevator', if the entry has total floors less than 5, the missing values were filled with 0; otherwise with 1. For the feature 'n_bathrooms', the median value, 1, was used to fill these missing values. Also, it was assumed that each house had at least one bathroom. Lastly, all the variables were converted into numerical values with 'pd.to_numeric' method. Then, the categorical variable 'conditions' was converted to a numerical variable using a mapping function. The house with the best condition got a value of 4, and the worst condition was mapped with 1. Others were attained respectively.

In order to improve the accuracy of the predictions, the distance of each house in the dataset to the three major cities in Italy: Milan, Venice, and Rome were calculated. The reason for choosing these three cities was because the coordinates of the houses were observed to be located there. The coordinates of each city were defined using tuples, and a function was created to calculate the distance between them by utilizing the Euclidean distance formula. Then, new columns for the distances to each city were created in the dataset. To determine which of the three cities was closest to each house, the ‘idxmin’ method was used along the ‘axis=1’ to find the column index with the smallest distance value for each row. Then, the closest city's column index was mapped to its corresponding name using a dictionary, which was defined as ‘city_names’. Finally, the unnecessary columns were dropped for distance in Venice, Rome, and Milan, and converted the categorical variable ‘Closest_city’ to a numeric one using the ‘get_dummies’ method. After this step, it was observed that there were still 13 missing values in the longitude, latitude, and proximity to center features. It was hypothesized that these missing values were the same for all observations. These observations were dropped with the subset argument set to ‘proximity_to_center’ feature. By doing these data preprocessing steps, the quality of the data indicating coordinated was improved and made ready for the machine learning algorithm.



The prices of the houses had an asymmetric distribution with positive skew and kurtosis. The 'price' feature of the training dataset was standardized by scaling it to have a mean of 0 and a standard deviation of 1 using the ‘StandardScaler()’ function. It then selected the 20 lowest and 20 highest values from the scaled prices and printed them as 'low_range' and 'high_range', respectively. after the standardized ‘price’ feature was printed, it was observed that the highest prices were so far away from the standardized mean ‘0’. These observations were interpreted as extraordinary conditions which seemed like outliers. When more restrictive boundaries were applied, such as setting them within the outlier range, my method became more accurate. However, this approach caused overfitting. Even though the squared error value decreased significantly, my actual predictions became worse. Therefore, I set wide boundaries which eliminated only extreme outliers.

The outlier detection was made for all variables by using the Interquartile Range technique and plotting each feature with respect to the target variable. Initially, the first and third quartiles of the columns were defined using the ‘quantile()’ method with a parameter of 0.25 and 0.75. The IQR was calculated by finding the difference between the third and first quartiles and used to identify outliers by setting lower and upper boundaries at 1.5 times the IQR from the quartiles. Rows with values outside of these boundaries were filtered out using a Boolean expression. The resulting outliers were printed, but, removing them would result in losing a significant amount of data. To get a better understanding of how many observations lie outside the outlier boundaries, the ‘sum()’ method was used. After analyzing the data, the boundaries were expanded to remove only the extreme outliers, allowing to retain a substantial portion of the data. Likewise, I decided to exclude houses with total number of floors and floor number that were outside the reasonable range of 0 to 30. This decision was based on previous research in the field which has shown that such values are unlikely and may be errors.

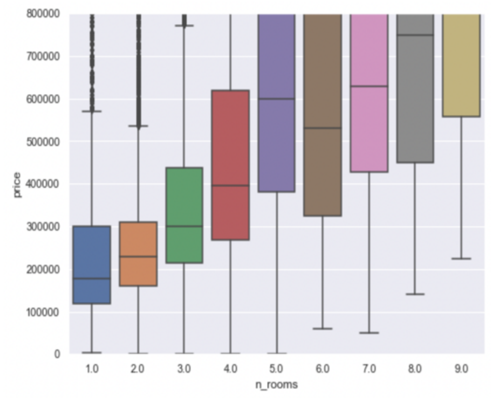
A new feature called 'city_center_distance' was formed, representing the distance of each house from the city center. The feature is calculated based on the 'proximity_to_center' feature.

Lambda function was used to map the 'proximity_to_center' to 'city_center_distance' feature. If proximity is 1, distance is set to 0, otherwise, distance is set to 1/proximity squared. 'proximity_to_center' feature was then dropped from the dataset.

The building age was calculated by subtracting the construction year from the current year (2023). Then, the rows with building age less than zero were dropped. Finally, the 'construction_year' column was dropped.

The ‘IterativeImputer’ function from scikit-learn library was used to impute the remaining missing values. The ‘random_state’ parameter was set to 42 for reproducibility. The ‘min_value’ and ‘max_value’ parameters were set to ensure that the imputed values fall within a certain range. Then, the dataset was copied to avoid modifying the original data. The ‘cols_to_imp’ variable was created to specify the columns that need to be imputed. These columns were ‘energy_efficiency’, ‘expenses’, ‘building_age’, ‘floor’, and ‘total_floors’.

The imputation was performed using the ‘fit_transform()’ method. The imputed values for the specified columns were then assigned back to the original dataset. Overall, this part of the code was performing iterative imputation to fill in missing values in specific columns of the dataset, allowing for more complete data to be used in subsequent analyses.



A similar procedure was followed with the ‘df_test’ dataset. It was made sure that there were no missing values in the observations. However, the missing values in each significant variables were imputed instead of dropping observations with missing values. This allowed to retain all the features in the dataset. Leaving the same columns and using the same technique ensured consistency in the data processing steps between the two data frames: ‘df_train’ and ‘df_test’. The ‘IterativeImputer’ was again used as an imputation method that works by estimating missing feature values using regression. This technique is beneficial for datasets with non-random missing data that depends on the values of other features. The code imputes several essential predictors of the target variable, including 'energy_efficiency,' 'expenses,' 'conditions,' 'surface,' 'n_rooms,' 'city_center_distance,' 'building_age,' 'floor,' and 'total_floors.'

REGRESSION MODEL, CONCLUSON, AND EVALUATION

The final part of the machine learning project is the regression model that was used for training and predicting house prices. Two different models were used and compared. The first one was ‘XGBRegressor’ and the other was ‘RandomForestRegressor’.

Firstly, the ‘XGBRegressor’ model was used for training and predicting house prices. The first step was to impute the missing values in the dataset. ‘KNNImputer’ was used to fill in the missing values in the train and test datasets. After imputing the missing values, the 'price' column was dropped from the training dataset and saved as 'X' while the 'price' column was saved as 'y'. The 'y' variable was then log transformed to make the data more normally distributed. Then, ‘StandardScaler’ was used to scale the features in the training dataset. The training dataset was then split into a training set and a validation set using the ‘train_test_split’ function. The validation set will be used to evaluate the performance of the model during training. The ‘XGBRegressor’ model was then defined with hyperparameters such as 'tree_method', 'objective', 'random_state', and 'n_jobs'. The 'tree_method' parameter specified the algorithm to use for constructing the trees in the model. The 'objective' parameter was set to 'reg:squarederror' to specify that the model will be used for regression. The 'random_state' parameter ensured that the model produced the same results in each run. The ‘XGBRegressor’ model was then trained on the training set using the fit method. The trained model was used to predict the house prices on the validation set using the prediction method. The model's performance was assessed by computing the root mean squared error (RMSE) between the predicted and actual prices in the validation set. Once evaluated, the model was utilized to predict house prices in the test set. The features in the test set were scaled using the ‘StandardScaler’ that was previously fitted on the training set. The predicted prices were then transformed back to the original scale by taking the exponential of the log-transformed predicted prices. Finally, the predictions were saved to a CSV file with the 'id' column from the test dataset and the predicted house prices.

Lastly, the Random Forest Regression was used to predict house prices. The process started by imputing missing values in the dataset using the ‘KNNImputer’. Then, the features were scaled using the ‘StandardScaler’. The dataset was then split into a training set and a validation set. The ‘RandomForestRegressor’ model was trained using the training set. The model was evaluated using the validation set, and the root mean squared error (RMSE) was calculated. The overall intuition behind this code was to create a model that can predict house prices based on the available features. By using Random Forest Regression, an ensemble of decision trees was generated, enabling the model to capture intricate associations between the target variable and the features. Additionally, the application of imputation and scaling techniques enhanced the data quality, leading to a better-performing model.

Overall, the results of the ‘XGBRegressor’ model were more accurate. I utilized this model in my final code, which gave the best predictions. However, the Random Forest Regression model can be seen in my codes too. It was shown that data preprocessing, feature scaling, and model training are essential steps in machine learning. The ‘XGBRegressor’ model was able to predict the prices of houses with good accuracy, as indicated by the low RMSE value. This model can be used in the real estate industry to predict the prices of houses and help people in making informed decisions.

One potential improvement to this model could involve manipulating the coordinates and proximity to center variables. For example, we could select expensive streets or well-known landmarks as coordinates and expect the price of the house to increase as it gets closer to these areas. However, since there are multiple expensive areas throughout the city, we would need to create a more complex distance formula that incorporates a multiplier. With help of that we could better capture importance of each neighborhood and how they may impact the overall price of a house. For instance, some neighborhoods may be known for their proximity to well-known streets or significant places, while others may be located in rural places. By assigning different multipliers to each neighborhood, we can better account for these factors and provide a more accurate prediction of the houses’ value. Overall, incorporating a more complex distance formula that considers neighborhood multipliers could greatly improve the accuracy of this model and make it more functional.