Final Report for the DS 675 - 005 Project

TITLE: PREDICTING THE PRICE OF HOUSE IN THE USA.

Keywords: Exploratory Data Analysis, Decision Tree, Random Forest, Gradient boosting, Ridge CV, Elasticnet CV, KNN.

Abstract: This machine learning project focuses on predicting house prices in the U.S. real estate market by employing a diverse set of regression models, including Decision Tree, Random Forest, Gradient Boosting, Ridge CV, and ElasticNet CV. Utilizing a dataset featuring crucial variables such as the number of bedrooms, bathrooms, house size, acreage, state, and city, our study employs advanced algorithms to extract intricate patterns within the data. The research involves rigorous model evaluation and comparison to determine the most effective approach for accurate price predictions. Insights gained from this comparative analysis not only contribute to the field of real estate prediction but also offer valuable guidance for selecting optimal models in similar applications.

Dataset: https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset/data
Code: https://www.kaggle.com/code/nikhilbudarayavalasa/nikhil

1. INTRODUCTION AND PROJECT STRUCTURE

We divided the tasks for this project into the following parts. This allowed us to tackle tasks in an efficient way and excel at each part individually.
I. Exploratory Data Analysis: Before we started applying any algorithms, we went over the data in depth. This step was particularly important because the chosen dataset is extremely imbalanced. We illustrated all the correlations between all the variables and conducted a deep research on which variables to use in our models.
II. "Simple" Classifiers + Performance Assessment: With the information found in the EDA, we chose to apply the Random Forest, Decision tree, gradient boosting, ridge CV and elasticnet CV, dropping some features along the process and used one-hot encoder for the categorical variables.
III. KNN, Random Forest and RidgeCV + Performance Assessment: We trained these three models and compared the performance with previous results.
IV. Combination of multiple classifiers + Performance Assessment: We trained an ensemble model to combine the models we created in the previous two steps to improve the performance further.

Step 1: Task And Scope Definition • *1.1: Define Project Objectives* • *1.2: Scope the Project* **Step 2: Literature Review** • *2.1: Define Research Questions* • *2.2: Search for Relevant Literature* • *2.3: Summarize Key Findings* • *2.4: Identify Research Gaps* **Step 3: Select Baselines** • *3.1: Select Baseline Models* • *3.2: Train Baseline Models*	**Step 4: Evaluation** • *4.1: Define Evaluation Metrics* • *4.2: Split Data for Evaluation* • *4.3: Evaluate Baseline Models* **Step 5: Result Analysis** • *5.1: Analyze Model Performance* • *5.2: identify additional models* • *5.3 train and evaluate additional models* **Step 6: Conclusion** • *6.1: Summarize Findings* • *6.2: Discuss Implications* • *6.3: Write Project Report*

2. RELATED WORKS AND REFINEMENT

Before starting our project, we went through previous studies performed by other people. There were around 33 notebooks related to this dataset.We found around 33 notebooks related to this dataset. We picked the 10 most rated and relevant notebooks to work on in our project. All of them had a similar structure: started with some brief data analysis and then built models that predicted house prices.

In Milestone 2, we particularly focused on the analysis conducted in the notebook Predicting Housing Prices EDA + ML | Kaggle. The author performed basic exploratory data analysis, focusing mainly on handling missing values, dropping Null and NA values and removing outliers. He then built five models, decision tree, random forests, Gradient boosting, ridge CV, elasticnet CV and compared their accuracy scores. The Random Forest algorithm was found to be the most accurate using MSE as evaluation metric.

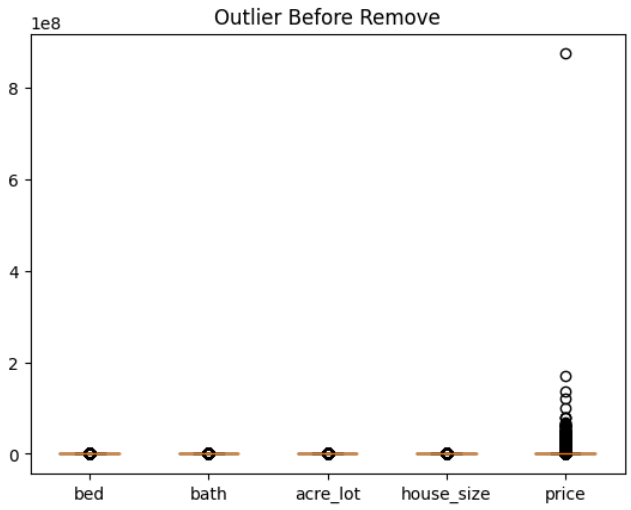
Our work is different from previous studies because of the following reasons:
First, we conducted **in depth exploratory data analysis**, looking into every single variable instead of only some of them. To be specific, within the possible values of each categorical variable, we analyzed the distribution of house price with almost all the variables. This allowed us to determine which variables would be more relevant to include in our models instead of automatically using all of them. We have done dimensionality reduction using PCA. Modified the feature selection as well. Second, all the preceding work was based only on a single approach. We combined the three best performing models into a single model. It is true that we took inspiration from previous implementations, but we're the only ones who came up with a combination. We worked on defining our own Ensemble, based on the three best models we obtained during parts II and III of this study.

2. DATA PREPROCESSING

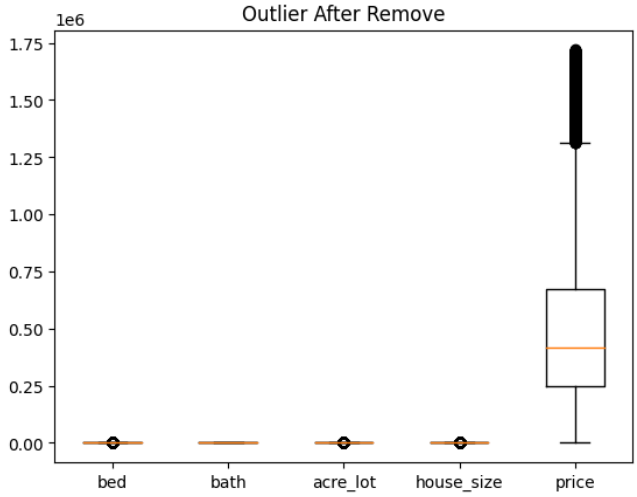
After a thorough examination, we discovered duplicate entries in the dataset. Upon further investigation into the duplicates, we identified a total of 979,394 instances. Consequently, we have removed these duplicates from the dataset. We have dropped them to ensure data quality and consistency. If duplicates are not removed, the model might assign more importance to the repeated values, leading to biased predictions. By eliminating duplicates, we are reducing the risk of introducing ***bias*** into our model.

Missing values are inevitable in any dataset, In our dataset, we have found 14.7% of NA values in "bed" , 13.3% of NA values in "bath", 28.3% of NA values in "acre_lot", 33.2% of NA values in "house size" and 46.6% of NA values in "prev_sold_state". We have dropped the "prev_sold_state" column as it doesn't affect the price of the house, so it's redundant in training the model. For the rest of the attributes with the missing values, we replaced them using "mode" (as per our problem statement, mode is the best way to fill the NA's of the number of beds and baths).

After the above mentioned process, we verified that there are zero null values in any of the columns, then we proceeded to remove the outliers,
We have plotted the outliers in the bed, bath, acre_lot, house_size, price.



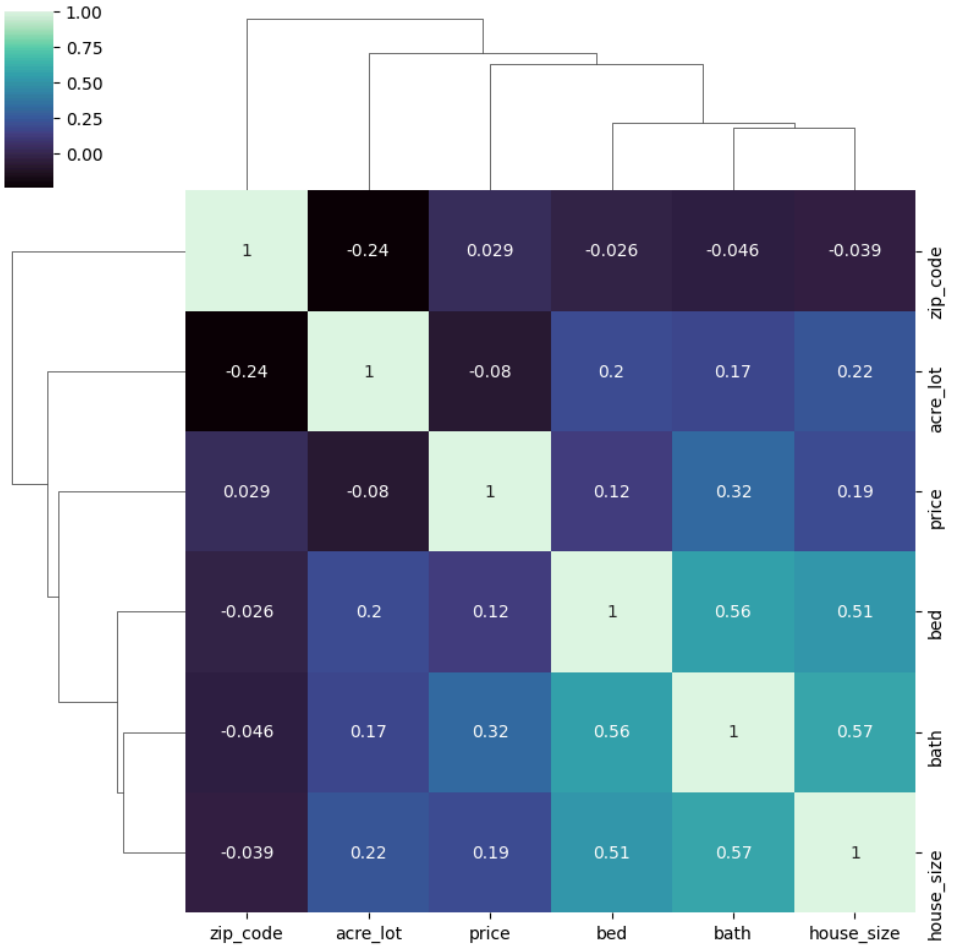
And the total number of rows with outliers is 125212.
We then proceeded to remove the outliers as it can affect the model performance, which can result in a model that is overly sensitive to extreme values and performs poorly on new, unseen data. Removing outliers will help in enhancing model robustness.



Total number of rows without outliers is 88016
For the categorical data, we have used one-hot encoding to convert categorical variables into a format that can be used by models. We also used scaling , Standard Scaling for 'house_size' and 'price', Standardization involves subtracting the mean and dividing by the standard deviation for each value. This results in the transformed values having a mean of 0 and a standard deviation of 1. Used MinMax Scaling for 'bed', 'bath', and 'acre_lot'. MinMax scaling transforms the values to a specified range (usually between 0 and 1). It's useful for algorithms that are sensitive to the scale of input features.

3. EXPLORATORY DATA ANALYSIS

First, we found the correlation among all variables and noticed that the strongest correlations are among these pairs in light blue colors: house size and bath of 0.57, bed and bath of 0.56, house size and bed of 0.51, bath and price of 0.32. From the above correlations, we can consider the bed, bath, house size attributes for model building.



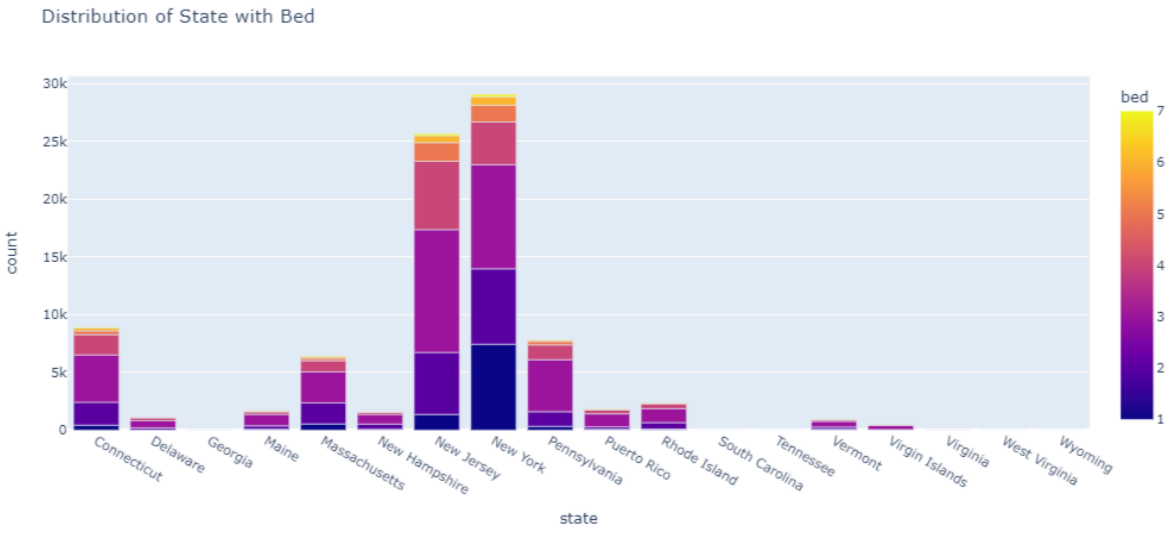
The descending order of distribution by bed is 36628 for 3 beds, 18928 for 2 beds and 15141 for 4 beds. Distribution by bath is 38760 for 2 baths, 25702 for 1 bath 18950 for 3 baths. The distribution of beds depending on bath shows that the most common pair is 3 beds and 2 baths, which counted for 19974 in the heatmap used.

From the distribution of acre lots, we found that while most US lots fall within the compact range of 0.04 to 0.11 acres, larger ones (>0.6 acres) exist. Remarkably, 31,700 lots have a singular size of 0.06 acres, highlighting a potential standardization trend.

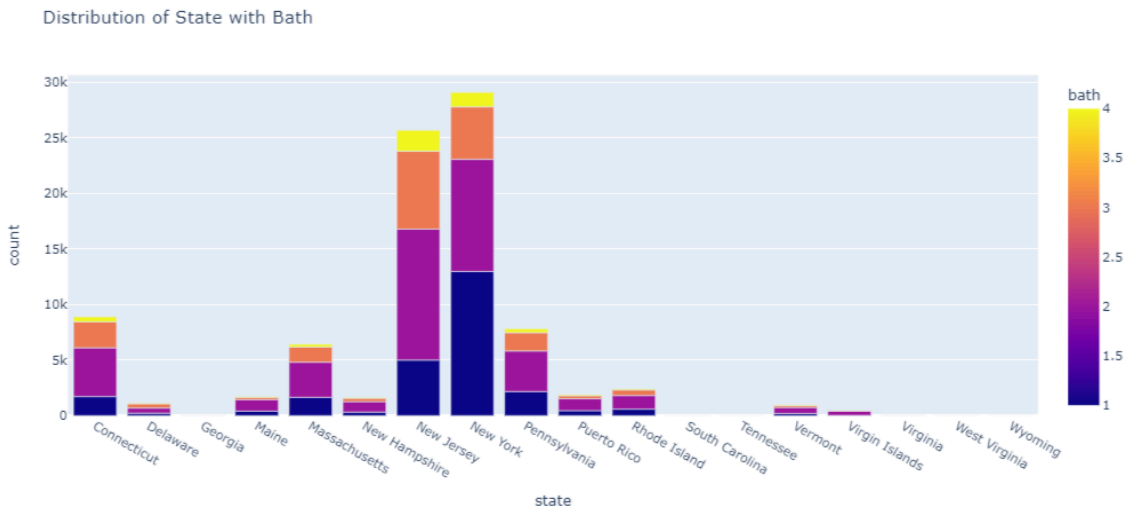
From the distribution of cities, we found that New York city, Philadelphia, Brooklyn, New York, and the Bronx have the highest number of houses.
We can see there is a flaw in data curation, as we can see New York mentioned here in the cities attribute which makes the data not accurate.

From the distribution of top cities with most beds, we can see that Philadelphia has the highest total bed count with “15206” followed by New York City with “9503” and Brooklyn with “9468”
From the distribution of top cities with most baths, we can see that Philadelphia has the highest total bath count with “9949” followed by New York City with “7053” and Brooklyn with “6695”

From the distribution of top cities with most acre lot, we can see that New York Cityhas the highest acre lot size with “44648” followed by Brooklyn with “21499” and Philadelphia with “20494”\
We can conclude that even though Philadelphia has less acre lot size when compared to NYC and Brooklyn, it has more bed and bath count.
We have plotted another visualization to look at the mean acre lot to see the top 10 cities, where we found that Sorrento has the highest mean acre lot of 1.16.
From the distribution of states, It is evident that New York and New Jersey dominated amongst the others with the highest number of houses of 29k and 25k respectively.
We have plotted a stacked bar chart to look at the bed distribution in the states. We have used color saturation to distinguish between the number of beds. The chart is attached below:



The same has been done with bath



We plotted another chart to look at the distribution of house size (in sqft). From the chart, we can see that there are 35k houses with house sizes between 1200-1399 sqft.

From the distribution of house size with bed, It is evident that 3 bed house counts are significantly larger than the remaining houses. In the EDA part of our project, we have plotted similar plots of various attributes with respect to other attributes to learn more about the distribution of the data across the highly correlated columns(attributes)

Models and assessments:

The steps we followed in this part were influenced by our findings in the EDA part and conclusions. In the EDA, we analyzed all the variables from the dataset and showed the correlation between and among these variables and between the price and some of these variables.

First, we included the bed, bath and house size attributes since they had a higher correlation with the price. We have found that the author did not include Zip Code in the feature selection as the correlation is less(0.013) between zip_code and the target variable. But, it is possible that the relationship between "zip_code" and "price" is non-linear or it involves interactions with other features. Non linear models like Gradient boosting , Random forest can capture such patterns. As we are already using them, we proceeded to add Zipcode to see the change in model performance.

We tested the five regression models: decision tree, random forest, gradient boosting, ridge CV and elasticnet CV, including the zip code variable and then re-tested the same models after dropping the variable. We found out that we should keep it in our model because all the models performed better when including it except for the elasticnet CV model. Then we proceeded to add the zip code attribute in the feature selection, x_train. Interestingly the results are improved.

In categorical variables, we added a one-hot encoder for the following variables: zip code, city and state in order to use regression on categorical variables for better outcomes and results.

After finishing with correlations, testing and normalizing the variables, we started by testing our models. Below mentioned is the breakdown of the entire training and testing process.

Results of baseline model:
Using the same feature selection and preprocessing.

Model	MSE	RMSE	MAE	R^2
Decision Tree	0.61	0.78	0.57	0.40
Random Forest	0.37	0.61	0.39	0.64
Gradient Boosting	0.54	0.74	0.54	0.46
Ridge CV	0.41	0.64	0.46	0.60
ElasticNet CV	0.97	0.98	0.76	0.01

We then trained a KNN regressor to check on the localized patterns(Even if the overall correlation is low, there could be localized patterns where specific zip codes have a more pronounced impact on prices.)

Model	MSE	RMSE	MAE	R^2
KNN	0.31	0.56	0.37	0.68

Shows that there are some localized patterns, and is the better performing model so far across all the metrics with low MSE and high R^2

Then added Zipcode in the feature selection, without one hot encoding the categorical variable

Results With Zip_code (without one hot encoding)

Model	MSE	RMSE	MAE	R^2
Decision Tree	0.54	0.74	0.53	0.46
Random Forest	0.26	0.51	0.32	0.75
Gradient Boosting	0.46	0.68	0.49	0.55
Ridge CV	0.41	0.64	0.46	0.60
ElasticNet CV	1.00	1.00	0.77	0.01
KNN	0.28	0.53	0.35	0.72

We can see that adding zipcode is a better choice as the performance of every model has improved except the linear models. Reason might be that we didn't one hot encode the categorical variable, that's why the linear models are taking a hit. For linear models, like RidgeCV, one-hot encoding is often recommended for categorical variables to ensure that the model treats each category independently. Decision trees, including those in ensemble models like Random Forest or Gradient Boosting, can handle categorical variables directly without one-hot encoding. Trees naturally split on categorical features.

Results With Zip_code (with one hot encoding)

Model	MSE	RMSE	MAE	R^2
Decision Tree	0.61	0.78	0.57	0.40
Random Forest	0.31	0.56	0.36	0.69
Gradient Boosting	0.54	0.73	0.54	0.47
Ridge CV	0.33	0.58	0.41	0.67
ElasticNet CV	0.61	0.78	0.57	0.40
KNN	0.28	0.53	0.35	0.72

The Linear models are greatly improved than that of the baseline models and the previous models. We can see a decrease in the MSE values from 1.00 to 0.61, same in the other metrics as well.
Now in order to further improve the performance of the models, we proceeded to work on an ensemble model: Voting regressor[Hard Voting]

First we went with 4 models, evaluation metrics are as follows:

Ensemble [RF,KNN,RCV, GB]	0.27	0.52	0.36	0.73

Then proceeded with 3 models, evaluation metrics are as follows:

Ensemble [RF,KNN,RCV]	0.25	0.50	0.33	0.75

Ensemble model with the models Random Forest, KNN and Ridge CV is giving us the best performance across all the metrics, with the lowest MSE rate of 0.25

Conclusion:

During this project, we faced some limitations with the dataset since it had a lot of missing values and duplicates. It had 979394 duplicates that were dropped and the missing values were replaced by calculating the mode of the respective attributes. We tested all the models and tested their performance by using MSE, RMSE, MAE and R squared, with and without the "zip code" variable.
We then added the KNN algorithm which is ideal for house price prediction regression analysis. We found that this model performed the best among all the others with the lowest error rate of 0.28 using MSE. Then we added the Ensemble model to our arsenal, which turned out to be the best performing model beating all the individual model metrics.
We concluded that, for future projects, we can work more on the dataset by merging the cities with keywords "NY" and "NYC" into one variable, and also by using an ensemble model technique in order to get more accurate results, joining RidgeCV, random forest and KNN all together in one simple ensemble model.

References:

Notebooks links:
https://www.kaggle.com/code/masghiff/predicting-housing-prices-eda-ml/comments
https://www.kaggle.com/code/kylwood/house-prediction/comments
https://www.kaggle.com/code/kaanxtr/real-estate-eda/comments
https://www.kaggle.com/code/dharmik34/eda-ml/comments
https://www.kaggle.com/code/izkoli/house-price-predictions/comments
Dataset link:
https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset
Models:
https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
https://scikit-learn.org/stable/modules/ensemble.html
https://machinelearningmastery.com/elastic-net-regression-in-python/
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html