Project 1 - Predicting California House Values

Jack Schwanewede, David Yoder
CSC 374 Deep Learning
Dr. Michelle Kuchera
Davidson College

Abstract

In this project, we took housing data from the 1990 California Census and created a model to predict the median house value of houses in a block group based on several variables. We built two models using Linear Regression, one using tools from the Machine Learning library scikit-learn, and the other through manually computing Gradient Descent. This was done using our AutoGrad class created in a previous assignment. Our results, while imperfect, take a step towards predicting house values with an R^2 coefficient score of around 0.64 in the scikit-learn model, and a similar score of 0.64 in our Self-Written Model.

Introduction

Housing markets are an integral part of economic stability and growth, having a great influence on everything from individual investment decisions to urban development progress. Accurate prediction of housing prices informs financial planning and policy-making, and aids in addressing issues like housing affordability and the spreading of urban development. In this paper, we train two Linear Regression models to predict the median house values given a set of features. The dataset used comes from the 1990 California Census Report, as also used in Kelly and Barry's *Sparse spatial autoregressions* (1997) [0]. We describe how we prepared and approached the data, as well as the methodology of how we built and analyzed our models. We conclude with ethical considerations about our project.

Housing Data

The housing data we used comes from the 1990 California Census Report. In the report, information and demographics are collected in block groups, a common geographical unit used by the Census Bureau. There are a total of $20,640$ block groups recorded in the dataset.

The data we used contained the following information for each block group:

- Longitude
- Latitude
- Housing Median Age
- Total Rooms
- Total Bedrooms
- Population
- Households
- Median Income
- Median House Value
- Ocean Proximity

In this project, we are concerned with how the other variables impact the target variable, **Median House Value**, which is the median value of all the households in the block group.

Features

In this section, we will give an overview of the features used to predict Median House Value. We will split these features into three main factors for this explanation: Rooms, Neighborhood, and Location.
Rooms: There are two features related to the room information of the block group. First, **Total Rooms** is the aggregate sum of all rooms in the houses in each block group. Second, **Total Bedrooms** is the total sum of just the bedrooms in the block group.
Neighborhood: There are several features related to information about the neighborhood and block group of each entry. First, **Population** is the number of people living in the block group, and **Households** is the total number of households. **Housing Median Age** is the median age of the households in the block group. **Median Income** is also measured as the median salary, measured in tens of thousands.
Location: Finally, there are three variables related to location. **Latitude** and **Longitude** refer to the geographical location of the block group. **Ocean Proximity** refers to the distance of the block group to the ocean or bay. This is a categorical variable with the following attributes:

- Near Bay
- Near Ocean
- <1H Ocean
- Inland
- Island

Pre-processing

Before we began cleaning data or building models, we created several graphs with the assistance of ChatGPT to help us visualize the data and discover patterns to help us understand the nature of the dataset.

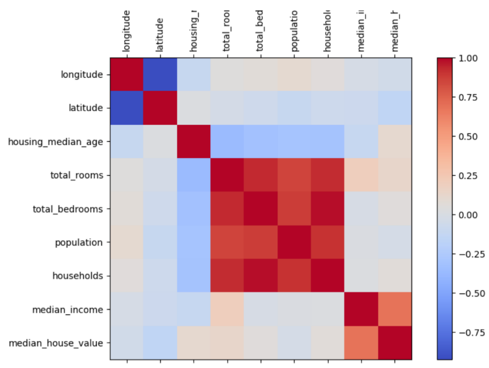

Figure 1: A heatmap of variable correlation, where red indicates a positive variable correlation, and blue indicates a negative correlation.

Figure 1 depicts a heatmap of the variable correlations, where a stronger relationship is indicated by a stronger color of red (positive) or blue (negative). In this initial view, we can see how the Total Rooms and Total Bedrooms features have a high positive correlation. This makes sense, as more bedrooms in a block group would add to the total rooms as well. Median Income and Median House Value had a moderate positive correlation.

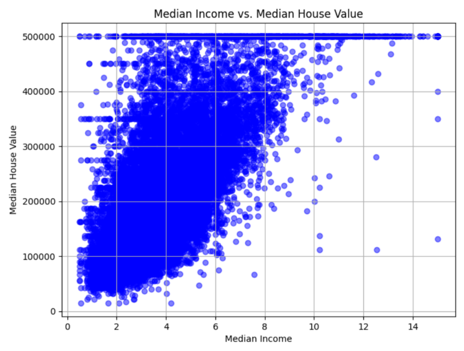

Figure 2: A scatterplot of Median Income and Median House Value.

The scatterplot in Figure 2 looks at how Median Income and Median House Income relate to one another. In Figure 1, the heatmap indicated a moderate positive relationship, and this scatter plot demonstrates that as well. The scaling is slightly off due to how income is measured, but generally an increase in income is correlated with an increase of house value. However, when looking at the graph there is a visible line of points at the $500,000$ y value. This is because the dataset caps house value at $500,000$.

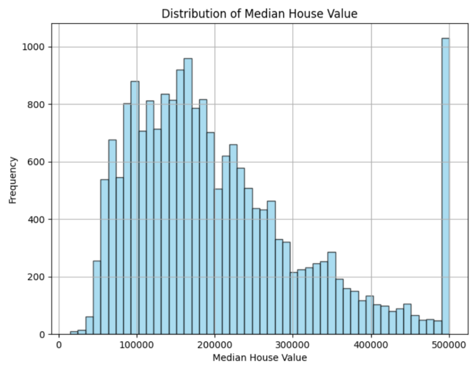

Figure 3: A frequency distribution of the different housing values.

To further exemplify this cap, we constructed a frequency distribution graph (Figure 3) to see what the most frequent house values were. The mode house value in this dataset was at $500,000$, likely due to the hard cap that makes any house worth over $500,000$ to be labeled as $500,000$. Besides this, we can see how the distribution is mostly normal besides this, with other high frequencies between $100,000$-$200,000$.

Methods

For our methods, we created two models to predict median house values using the 1990 California Housing Dataset. Data was cleaned before being trained in the our models. Both models utilized Mutlivariate Linear Regression. The first model was used using scikit-learn, and the second was a Gradient Descent algorithm using our built AutoGrad class. We used GitHub to coordinate our work and files.

Data Cleaning

We started the project by cleaning the data and preparing it for implementation. First, we assessed whether null values were in the dataset, and found a total of 207 null values. All of these values were in the Total Bedrooms feature, and we replaced null values with the median total bedrooms.

Additionally, the Ocean Proximity feature was initially a categorical variable, so we needed to change its encoding. To do this, we used One-Hot Encoding, where each of the five possible attributes of Ocean Proximity were turned into a binary value of 0 or 1. For example, if a block group was located within an hour of the ocean would have values of $[1, 0, 0, 0, 0]$, with the first value representing the <1H Ocean feature, and the zeroes representing the other possible proximity features. Another factor of note is that the Island feature only had five entries.

We ran a program to implement these changes and create a new housing dataset that would be used for our models.

Model 1: scikit-learn

Using scikit-learn, we imported several packages to assist in developing our model. Before that, though, we had to split up the data for training, validation and testing. We split the initial data, setting 20% of the data as testing and leaving the remaining 80% for training and validation. Of this, 80% of the remaining data was used for training while 20% was for testing.

Next, the data was scaled using Standardized Scaling. This is done by taking each value, subtracting the mean of the feature, and dividing by its standard deviation. This process makes all features have a comparable scale, which will improve the performance of our model.

The model we used from scikit-learn utilized Linear Regression. We fit the model on our scaled training features, and the Median House Values. Using the predict function, we used the training data to predict the Median House Values for our training, validation and testing datasets.

Additionally, we created a separate model to compare how outlier removal would impact results. The process was the same, however, outliers were not removed from testing data in order to preserve the integrity and accuracy of results. Outliers were defined as entries that had features that were more than three standard deviations away from the mean. A total of 608 outliers were removed from the training data.

For both models, we calculated R^2 scores for training, validation and testing to see how well the predicted housing values matched the actual housing values. We also found the loss of the function, using the Mean Squared Error (MSE). Finally, we plotted scatter plot graphs using matplotlib of the actual housing values versus the predicted housing values.

Model 2: Self-Written Model

Our methodology for the Self-Written Modelient Descent model (Self-Written Model) is similar to the scikit-learn model but implemented from scratch using our custom AutoGrad class. We began by loading the cleaned dataset and splitting it into training, validation, and testing sets, using the same 80:20 split for training/validation and testing, and then an 80:20 split of the initial 80% for the training and validation sets. The data was also shuffled randomly before splitting. Standardized Scaling was performed manually, applying the same formula as used in the scikit-learn model.

This model relies on the AutoGrad class, which enables automatic differentiation. We built this class in a previous activity. To begin building the model, we initialized the weights and bias parameters randomly using in an initial parameter function. Our predict function uses these parameters, along with the AutoGrad class, to predict the house value for each block group in the rows of the dataset.

Next, we trained the model using the train function, which updates the weights and bias using Gradient Descent. Gradient Descent is a common algorithm used in optimization methods, and aims to minimize a function by following the direction of the slope at a point until a minimum is reached (Ruder, 2016) [0].

We also calculated loss with the Mean Squared Error function each step of the way, and used the Gradient Descent function to update the weights and biases. This process continues until the maximum iterations are reached, and we experimented with using different max iterations such as $1,000$ or $5,000$. The training loop would also end if the loss converges, which means the change in loss would be under a given tolerance value. Besides max iterations, another hyperparameter we tuned was the learning rate, which changes the extent to which the gradient descent algorithm updates.

Like our scikit-learn model, we measured the R^2 score on the training, validation, and testing sets, which is a measure of how well the model fits the data. We also plotted a loss curve throughout the iterations of training, and finally we created scatter plots of the actual housing values compared to the predicted housing values.

Claude AI assisted in formatting and organizing code relating to evaluation and visualization, after we verified our code was functional and working.

Results

Model 1: scikit-learn

In our regular model, we found a R^2 score of 0.638 with a loss of 4.71 billion. Our no outlier model had a R^2 score of 0.624 and a loss of 4.9 billion, which suggests our methods of removing outliers was ineffective and actually hurt the model. For this reason, we did not remove outliers in our Self-Written Model.

Figure 4, below, displays the plot of the scikit-learn predicted testing values versus the actual values, and has an R^2 score of 0.6381.

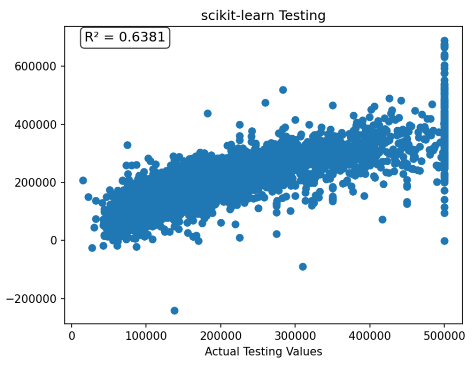

Figure 4: Predicted vs Actual Values for scikit-learn test data

Model 2: Self-Written Model

For the Self-Written Model, we recorded the predicted versus actual testing data for different amounts of maximum iterations. At $1,000$ iterations, our calculated R^2 score is 0.5529, as seen in Figure 5. Figure 6 depicts our scatterplot at 5000 iterations, which results in a R^2 score of 0.636. Next, Figure 7 depicts the predicted house values and the actual house values at $10,000$ iterations, which continues to improve the R^2 score to 0.64.

We also measured feature importance at each iteration. There was some variation of feature importance for the difference in 1000, 5000, and 10000 iterations.

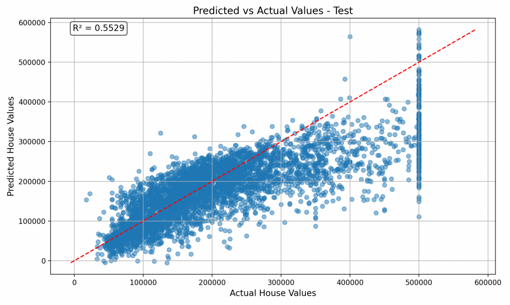

Figure 5: Predicted vs Actual values for test data at 1,000 iterations

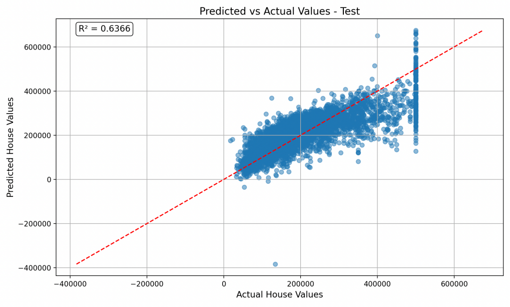

Figure 6: Predicted vs Actual values for test data at 5,000 iterations

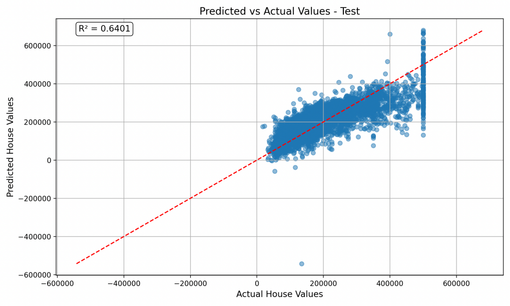

Figure 7: Predicted vs Actual values for test data at 10,000 iterations

Analysis

One method of re-analyzing our model was through the changing of the scaling. On a second attempt, we used Robust Scaling, which aims to reduce the impact of outliers by using the median and inter-quartile range (IQR) of the dataset to scale values. We compared Standardized Scaling to Robust Scaling, yet found there to be no difference in the results.

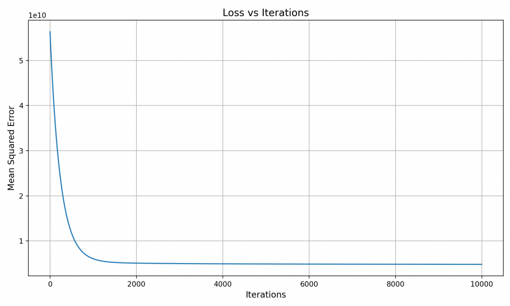

Figure 8: The Loss Curve over 10,000 iterations

Effect of Learning Rate and Iterations

We examined our self-written gradient descent model with varying hyperparameter configurations. Our findings revealed that as iterations increased, the R^2 score improved significantly, with the most dramatic improvements seen when adjusting the learning rate.

Learning Rate	**Iterations**	**Testing R^2 Score**
0.0001	1000	-1.9476
0.0001	5000	0.1207
0.001	1000	0.5529
0.001	5000	0.6366
0.001	10000	0.6401

Table 1: Effect of learning rate and iterations on model performance

The results demonstrate that both parameters significantly impact performance. At a low learning rate of 0.0001 with 1000 iterations, the model performed worse than a simple mean predictor (negative R^2). However, increasing either the learning rate or the number of iterations led to substantial improvements, with the optimal configuration achieving a test R^2 score of 0.6401 at a learning rate of 0.001 and 10000 iterations.

Compare the effect of iterations on feature performance In all three iteration levels, Median Income was the most important feature. Population was top three in all iteration levels as well, being second in both 5000 and 10000.

One inconsistency we found was in the importance of Total Rooms and Total Bedrooms. In the 10000 iterations Total Rooms was last by a significant amount, yet in 1000 iterations it was higher than Total Bedrooms. We would also expect these to have similar importance levels since Total Rooms in part relies on Total Bedrooms, yet this was not the case in any iterations.

In the 10000 iterations, the Ocean Proximity features were in the bottom half of the feature importance,

Ethics

When using data to create models, it is important to be aware of the potential impacts and sensitivity of the information. The housing data we used for this project was collected in

1990 through the Census Bureau and has been public information for quite some time.

Still, the data we used includes information such as income and latitude and longitude, which can be used to assess socio-economic status and location. There is potential for misuse of a model like this for gentrification, where lower income areas may be targeted due to other favorable features that would be predictive of higher income.

However, since the data looks at block groups and not individual houses, we believe that there is less reason to be concerned about the privacy of individual homeowners and their data. This, alongside the age of the data, mitigates negative impacts from the model. This data has also been used in existing papers, such as Pace '97. [0] Overall, we conclude that potential ethical considerations from this project are minor, and our private application of predicting house values does no harm.

Limitations

There are a few limitations our model did not address. One limitation has to do with the the data itself, as it does not have the full scope of actual house prices. Since the housing values are capped at $500,000$, houses worth multiple millions of dollars would appear identical to a house worth $500,000$.

Outliers may still impact our data - our method of removing them in the scikit-learn model was not effective. With more time or other procedures, outlier removal could potentially help the performance of both of our models.

Hardware - some tests took upwards of an hour to run, and using Juypter Hub did not speed up the process. It was in fact, slower than the MacBook Pro with an M3 Pro chip used to run the model.

Conclusion

Say what we did Compare models Takeaways about predicting house values, how we did, and future improvements?

Acknowledgments

ChatGPT assisted in our initial visualizations during the preprocessing phase. Additionally, once our models were created, Perplexity and Claude assisted in guiding us in the right direction when we were stuck. We used Claude to format our code when deemed necessary for organization, while keeping our original logic the same. Any code that was generated by an LLM was used for plotting, or functions to calculate R^2 score. In addition, if we reached a wall in our code, we would ask for guidance with how to continue. All code generated by any sort of LLM will be stated in the code found in the GitHub repository. ChatGPT assisted in LaTeX formatting for the writing of this paper.

References

R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.