Project 8 - Decision trees and neural networks

Nate Taylor & Clark Farnsworth

Slides | Repository | Dataset

Introduction

Housing prices are on the rise, especially in certain parts of the country, and many first-time buyers and sellers alike are uncertain as to what to expect when approaching a sale. Through decision tree and neural network predictive models, we hoped to answer questions Utah/Colorado buyers and sellers might have, such as: What kind of house can I afford with the money I can realistically put towards a house? What are similar houses going for right now? If I want a house with certain characteristics, what can I expect to pay? Using a dataset of houses on the market in mid-2022, we used each house bedroom, bathroom, size, acreage, and location data to answer these questions and discover trends.

Dataset

The dataset we used was a USA Real Estate Dataset from Kaggle, which contained data for over 2 million houses on the market in mid-2022 across all US states. Data it provided included broker, status (recently sold or still on the market), price, numbers of bedrooms and bathrooms, acreage, location, size, and last sell date. The numerical data was especially useful in predicting housing prices, so our models used bedrooms, bathrooms, acreage, size, and state. We narrowed our data down to houses in Utah and Colorado only (about 20,000 of the samples), as our stakeholders reside or wish to reside there. In order to increase predictive power, we separated houses into price ranges based on price quantiles.

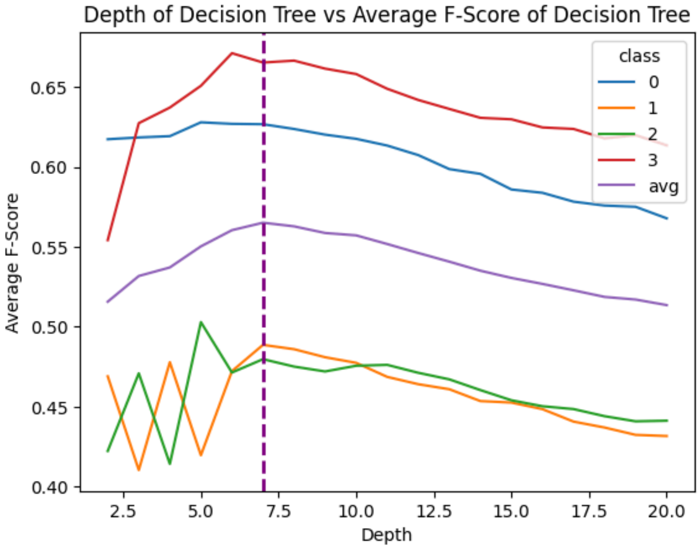
Analysis technique

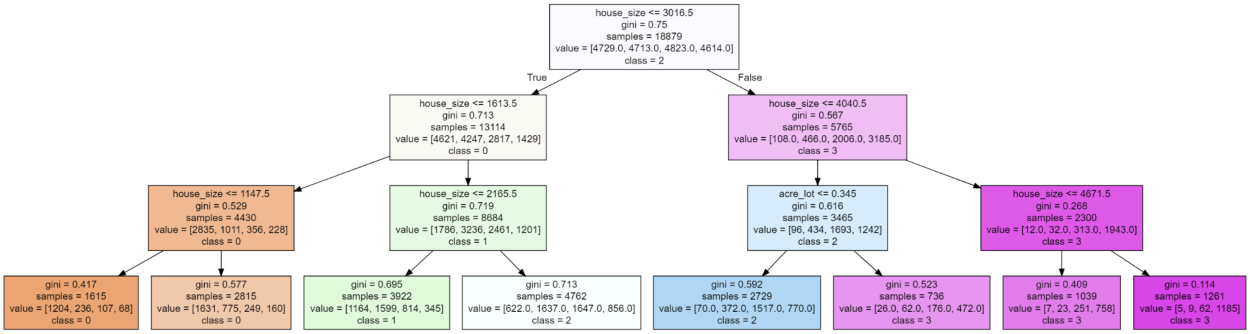
We wanted to easily visualize which house features contributed most to predicting house price, so we implemented decision tree models to predict ranges of prices. This allowed us to see basic decision paths (which can be thought of as house feature combinations) that led to certain price ranges. We ran 30 iterations of Monte-Carlo cross-validation with 75%/25% training/testing data splits to ensure that our models didn’t overfit the data we were using. We also utilized neural networks to further isolate important house features and to see if we could get more powerful predictions.

Results

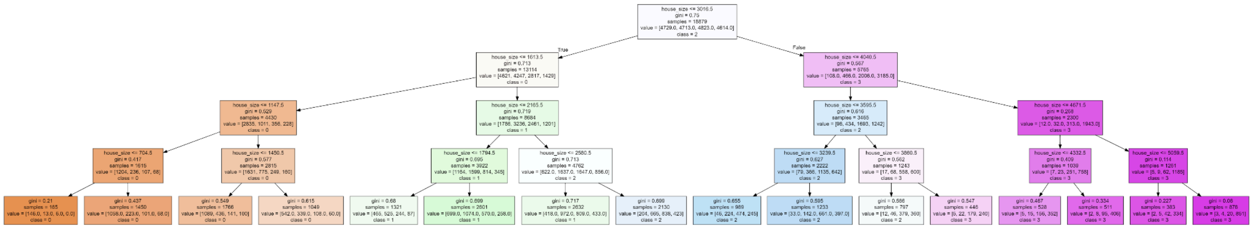
We found that our predictive models weren’t as powerful as we hoped they would be at predicting price ranges of houses. On average, our best predictions could predict houses in the price range of $12,000 to $430,000 with an f-score of about 62.7% and houses in the price range of $775,000 to $3,190,000 with an f-score of 66.5%.

Predictions of houses in the middle two price ranges ($430,000 to $555,000 and $555,000 to $775,000) averaged f-scores of 48.8% and 48.0% respectively at best. We used f-score as an evaluation of our predictions because between precision and recall, neither was more crucial than the other. We concluded that the sheer variety of house feature combinations contributed to this unpredictable quality of our data. Our models overall were better at predicting houses in the lowest and highest price range, but surprisingly, predictive power did not vary significantly between decision tree models and neural networks, and even different tree depths or network layer counts did little to improve it.

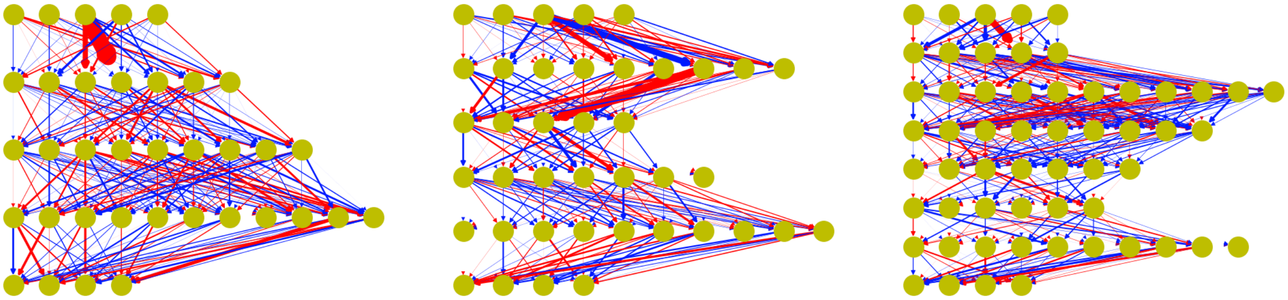




Our decision trees seemed to indicate that a house's price was almost entirely accounted for by the house's size. Tree visualizations revealed that relatively clear size boundaries could be drawn between different price ranges. $12,000 to $430,000 homes had square footages up to around 1,600 square feet, while $430,000 to $555,000 homes ranged up to 2,580 square feet and $555,000 to $775,000 homes ranged up to 3,680 square feet. Acreage played an extremely minor role in determining price, and bathrooms an even smaller role. This indicates to a seller that house size is a much more important factor in price-setting than number of bedrooms or other features.


house_size <= 3016.5 gini = 0.75 samples = 18879 value = [4729.0, 4713.0, 4823.0, 4614.0] class = 2
True
False
house_size <= 1613.5 gini = 0.713 samples = 13114 value = [4621, 4247, 2817, 1429] class = 0
house_size <= 4040.5 gini = 0.567 samples = 5765 value = [108.0, 466.0, 2006.0, 3185.0] class = 3
house_size <= 1147.5 gini = 0.529 samples = 4430 value = [2835, 1011, 356, 228] class = 0
house_size <= 2165.5 gini = 0.719 samples = 8684 value = [1786, 3236, 2461, 1201] class = 1
house_size <= 3595.5 gini = 0.616 samples = 3465 value = [96, 434, 1693, 1242] class = 2
house_size <= 4671.5 gini = 0.268 samples = 2300 value = [12.0, 32.0, 313.0, 1943.0] class = 3
house_size <= 704.5 gini = 0.417 samples = 1615 value = [1204, 236, 107, 68] class = 0
house_size <= 1450.5 gini = 0.577 samples = 2815 value = [1631, 775, 249, 160] class = 0
house_size <= 1794.5 gini = 0.695 samples = 3922 value = [1164, 1599, 814, 345] class = 1
house_size <= 2580.5 gini = 0.713 samples = 4762 value = [622.0, 1637.0, 1647.0, 856.0] class = 2
house_size <= 3239.5 gini = 0.627 samples = 2222 value = [79, 366, 1135, 642] class = 2
house_size <= 3860.5 gini = 0.562 samples = 1243 value = [17, 68, 558, 600] class = 3
house_size <= 4332.5 gini = 0.409 samples = 1039 value = [7, 23, 251, 758] class = 3
house_size <= 5059.5 gini = 0.114 samples = 1261 value = [5, 9, 62, 1185] class = 3
gini = 0.21 samples = 165 value = [146.0, 13.0, 6.0, 0.0] class = 0
gini = 0.437 samples = 1450 value = [1058.0, 223.0, 101.0, 68.0] class = 0
gini = 0.549 samples = 1766 value = [1089, 436, 141, 100] class = 0
gini = 0.615 samples = 1049 value = [542.0, 339.0, 108.0, 60.0] class = 0
gini = 0.68 samples = 1321 value = [465, 525, 244, 87] class = 1
gini = 0.699 samples = 2601 value = [699.0, 1074.0, 570.0, 258.0] class = 1
gini = 0.717 samples = 2632 value = [418.0, 972.0, 809.0, 433.0] class = 1
gini = 0.699 samples = 2130 value = [204, 665, 838, 423] class = 2
gini = 0.655 samples = 989 value = [46, 224, 474, 245] class = 2
gini = 0.595 samples = 1233 value = [33.0, 142.0, 661.0, 397.0] class = 2
gini = 0.586 samples = 797 value = [12, 46, 379, 360] class = 2
gini = 0.547 samples = 446 value = [5, 22, 179, 240] class = 3
gini = 0.467 samples = 528 value = [5, 15, 156, 352] class = 3
gini = 0.334 samples = 511 value = [2, 8, 95, 406] class = 3
gini = 0.227 samples = 383 value = [2, 5, 42, 334] class = 3
gini = 0.08 samples = 878 value = [3, 4, 20, 851] class = 3

Conversely, neural networks relied heavily on acreage in predictive models, with house size and number of bathrooms playing a limited role in strengthening predictions. Summarily, between the two model types, it unsurprisingly seems like acreage and house size consistently predicted the price of houses the best. This means that buyers more concerned about other house features such as number of bedrooms or bathrooms will have an easier time finding affordable houses by searching for these features amongst houses of smaller size and acreage.



Technical

We wanted to focus our analyses on houses in or near Utah, so we first narrowed our data down to several states in the general area. Distribution comparisons revealed that Colorado's housing data most closely matched Utah's while other states varied, so we further limited the data down to Utah and Colorado only. We dropped any houses that were missing important feature values (the feature combinations varied so widely that we felt like trying to estimate values would lead to problems), and we also removed price outliers (more than 2 standard deviations from the mean price), as we wanted to focus more on houses affordable to the average American than the grossly expensive houses that were severely right-skewing the data. Lastly, we separated houses into price ranges based on price quartiles to limit the number of classes to predict. Despite the difference in ranges of the quartiles, we felt like quartile ranges were a suitable choice because it provided even classes that did not need to be weighted in predictive models to account for class imbalance. As we should have predicted, our choice of price ranges led to the middle price ranges with narrower ranges being harder to predict, so an alternative approach we might've tried would be to separate the data into uniform price ranges and either random sample each price range to produce a balanced number for each class or weight classes in predictive models.

We felt this dataset would do well with decision trees because we thought house feature combinations would be easy to visualize along tree paths. Trees did indeed show some interesting trends and provided some easily-visualizable separations of the data. One such observable pattern was the effect of tree depth. Our tree of depth 3 relied almost exclusively on house size (seemingly high bias), but the tree of depth 5 only relied on two additional features to only a very minor extent. Trees using only housing price performed about the same as trees with additional features, showing that the simpler trees were not overly simple or biased. Testing depths ranging from 2 to 20 showed that a depth of only 7 maximized predictive power, while depths greater than 7 showed the effects of high variance and overfitting with decreasing f-scores. Deeper trees showed splits based on house size at the highest levels, followed by acreage and then number of bathrooms. Interestingly, simpler trees showed that acreage was key in distinguishing between the highest two price ranges, but not as key for the other price ranges.

We hoped neural networks would perform better than our trees, but they performed about the same. We did not maximize the usefulness of neural networks because we had few features to feed it as inputs, so finding a housing dataset with an increased variety of features would’ve been more effective. Interestingly, neural networks relied much heavier on house acreage than other features. We were expecting to see a reliance on house size like the decision tree models, but varying hidden layer sizes and counts showed a consistent pattern of preference for acreage in the models. Between the two types of models, it makes sense that house size and acreage are the two biggest factors in house price, so we might assume that our models were functioning correctly.