COE 379L - Stubbs
Arsh Dauwa

Project 2 Report

Introduction:

Housing prices are a critical part of real estate markets, impacting buyers, sellers, and policymakers. It would be extremely helpful to know whether a house is overpriced compared to the median for purposes of investment, market trends, and economic planning. This project will develop a classification model to determine if a house in California is overpriced compared to the median using various housing-related features such as median income, age of the house, and average rooms. Utilizing machine learning techniques, we work through the dataset, perform exploratory data analysis (EDA), and execute some different classification techniques to achieve best prediction accuracy. We compare the different models in an attempt to determine the leading method and identify the most significant drivers of house prices.

Comment on Anomalies in Data

The information includes multiple features related to California house prices, including median income, house age, number of average rooms, and population. Upon initial review of the information, a couple of anomalies were found. MedInc mode reported two single values, indicating bimodality or outliers. The histogram for MedInc would also need to be verified for skewing, and there might be multicollinearity among variables like AveRooms and AveBedrms, which would have an effect on model efficiency.

Comment on Univariate Analysis

Univariate analysis was conducted to compute the mean, median, and mode of all variables to comprehend their central tendencies and distribution in general. Additionally, graphical representations such as histograms and boxplots were employed to continue evaluating distributions, identify extreme values, and recognize possible imbalances in the dataset. The *HouseAge* boxplot will be particularly useful to identify outliers, indicating potentially much older or newer homes in some neighborhoods. Similarly, the *MedInc* histogram would allow one to see whether income distribution is skewed, normally distributed, or clustered within specific ranges. These analyses help in understanding the characteristics of the dataset and, therefore, guide the preprocessing steps, which may include handling outliers or applying transformations to improve model performance.

Which Techniques Did You Use to Train the Models? (1 point)

Several machine learning techniques were applied to train models to predict whether a house price was above the median. The K-Nearest Neighbors (KNN) classifier was implemented with *n_neighbors=3*, while a Decision Tree classifier was trained without pruning, leading to signs of overfitting. A Random Forest classifier was used with hyperparameter tuning via *GridSearchCV* to optimize parameters such as *n_estimators*, *max_depth*, and *min_samples_leaf*. Additionally, an AdaBoost classifier was employed using a decision stump as the base learner to enhance predictive performance.

Several machine learning algorithms were applied to train models to forecast whether a house price was greater than the median. The K-Nearest Neighbors (KNN) classifier with *n_neighbors=3* was applied,

which considers the three closest data points when making a prediction. The Decision Tree classifier was trained without interruption, allowing it to reach its full depth, and therefore producing overfitting indicators since it learned the training set patterns by heart rather than generalizing well. To improve performance, a Random Forest classifier was used with hyperparameter tuning using *GridSearchCV*, adjusting key parameters such as *n_estimators*, *max_depth*, and *min_samples_leaf* to get a balance between bias and variance. In addition, an AdaBoost classifier was applied that used a decision stump (a one-split tree) as the base learner, iteratively adjusting weights to focus on the misclassified instances and enhance predictive accuracy. Each model brought unique strengths, with AdaBoost ultimately achieving the perfect balance between accuracy and generalization.

Explain Any Techniques Used to Optimize Model Performance?

To achieve maximum model performance, several techniques were employed for maximizing generalization as well as accuracy. Hyperparameter tuning through *GridSearchCV* was pivotal in determining optimal values for the Random Forest as well as AdaBoost classifiers to ensure that the models were neither too complex nor too simple. Balancing classes was also done in the Random Forest model by adjusting the *class_weight* parameter, which also prevented class imbalances and avoided the model overrepresenting the majority class. Additionally, cross-validation was used to test model performance on different subsets of data, improving reliability and preventing overfitting. These methods together improved the models, with better predictive performance and more accurate results.

Compare the Performance of All Models to Predict the Dependent Variable? (1 point)

The performance of all models was evaluated based on accuracy, precision, recall, and F1-score. The KNN model performed the best, achieving an accuracy of approximately 88%. The Decision Tree classifier showed strong performance on the test set, with an accuracy of 83%, but displayed signs of overfitting, achieving 100% accuracy on the training data. The Random Forest classifier, after hyperparameter tuning, attained an accuracy of 72%, with a recall of 96% for the positive class, making it suitable for identifying houses above the median price. However, the AdaBoost classifier demonstrated the best overall performance, achieving 88% accuracy on both the test and training sets, with balanced precision, recall, and F1-scores.

Model	Precision (Test)	Recall (Test)	F1- Score (Test)	Accuracy (Test)	Precision (Train)	Recall (Train)	F1- Score (Train)	Accuracy (Train)
K- Nearest Neighbors	0.88	0.88	0.88	0.88	-	-	-	-
Decision Tree	0.83	0.83	0.83	0.83	1.00	1.00	1.00	1.00
Random Forest	0.78	0.73	0.71	0.72	0.79	0.72	0.70	0.72
AdaBoost	0.88	0.88	0.88	0.88	0.89	0.89	0.89	0.89

COE 379L - Stubbs
Arsh Dauwa

ChatGPT Usage in the Project

During training the Random Forest Classifier, I initially used a more complex hyperparameter grid that consumed a lot more computation time when running GridSearchCV. The initial parameter configuration, as shown in the part of the code that is commented out, used a wide range of values for n_estimators, max_depth, and min_samples_leaf. In particular, n_estimators ranged from 10 to 100 by steps of 2, so it had 45 different values. In the same manner, max_depth contained 8 different values while min_samples_leaf contained 5, leading to an excessively high number of hyperparameter combinations. The model thus took an inordinate amount of time to run, making it difficult to compare and assess models efficiently.

To determine this issue, I used ChatGPT, which successfully identified the issue immediately. The too-large hyperparameter grid was causing an exponential increase in computation time. ChatGPT suggested reducing the parameter grid by lowering the number of values but maintaining a huge range for hyperparameter tuning. Following this advice, I also reduced the grid by choosing three values for *n_estimators* (10, 50, 100), cutting down *max_depth* to three values (3, 5, 10), and reducing *min_samples_leaf* to three values (1, 5, 10). I also cut down class_weight options to two, allowing for a more efficient but not less effective search.

Which Model Would You Recommend to Be Used for This Dataset? (1 point)

Based on the results, AdaBoost classifier is the best algorithm for this data set due to its good generalization and balanced performance across all measures. It performed very well (88%) on the test set without a corresponding loss on the training set (89%), which indicates that it learns patterns effectively without overfitting. However, the Decision Tree model, while very accurate (85%) on the test set, exhibited ideal scores on the training set (100%), suggesting overfitting and possible weakness when applied to unseen data. Random Forest model, as a good all-around performer, while, however, exhibited lower test accuracy (72%) and poorer recall compared to AdaBoost, which suggests it would likely perform lower at properly classifying positive cases. K-Nearest Neighbors (KNN) was worst overall at only 61% accurate, and it is not possible that it is not well-suited for this data. In contrast, the highest precision, recall, and accuracy were given by AdaBoost and hence is the most stable model for this classification task.

For This Dataset, Which Metric Is More Important, and Why? (1 point)

As the issue here is, by definition, recall is a very important step because correctly capturing houses that are higher than the median price is more vital for potential buyers, investors, and policymakers who rely on accurate market data. High recall will ensure that the majority of the expensive houses are captured correctly, and there are fewer possibilities of missing out on important properties of potential interest. For investors, it means they can make more informed decisions about high-value property, and policymakers can better analyze housing trends and affordability. Additionally, for real estate agents, a model with high recall minimizes the danger of underpricing property, leading to more precise market analysis and strategic planning.