Rhea Samuel
COE 379L
rss3488

Project 02: California Housing Classifier

Data Preparation

In order to prepare this dataset, I first started by examining the data types of each variable within the dataset. By getting familiar with the variables, I identified that none of the variables, except for the **price_above_median** variable, required data type conversion. This variable included just two unique values, 0 and 1, showing that it was categorical data. Therefore, I was able to proceed with one-hot encoding. However, before doing this, I first checked if there were any missing or duplicate values in the dataset. As there was not, I was able to proceed with converting this variable. Finally, I was able to prepare the model for evaluation by splitting the dataset into a 70/30 training and testing set using the *train_test_split* function.

Model Training Techniques and Optimization

Within my project, I used three different classification machine learning models:

1. K-Nearest Neighbors
2. Decision Tree Classifier
3. Random Forest Classifier
4. AdaBoost Classifier

For my four models, I used the *sklearn* library to train and applied different techniques to improve their performance. The K-Nearest Neighbors algorithm classifies data points based on the labels of their closest neighbors. For this model, I standardized the data using *StandardScaler* so that each feature contributed. I also used *GridSearchCV* to find the optimal number of neighbors for better accuracy. The Decision Tree Classifier creates predictions by dividing the data into branches based on decision rules. I decided on using constraints, which are similar to those used in the Random Forrest Classifier, to prevent overfitting. The Random Forest Classifier builds multiple decision trees and combines their predictions. Finally, the AdaBoost Classifier trains multiple simple decision trees one after the other, reviewing the past tree in order to correct any mistakes. Again for this method, I standardized the data using *StandardScaler*. I also optimized the learning rate and the number of estimators using *GridSearchCV* to improve performance. As a result of all of these techniques, my goal is to optimize performance of the model and reducing overfitting.

Model Performance

In regards to the model performance, we can see the following results:

Testing Data:

	Accuracy	Recall	Precision	F1
KNN	84%	84%	83%	84%
Decision Tree	78%	74%	88%	77%
Random Forrest	88%	90%	87%	88%
ADABoost	88%	88%	87%	88%

Training Data:

	Accuracy	Recall	Precision	F1
KNN	86%	86%	86%	86%
Decision Tree	80%	75%	83%	78%
Random Forrest	97%	100%	95%	97%
ADABoost	89%	89%	89%	89%

Confusion matrices:

K-Nearest Neighbors	Decision Tree
[[2575 520] [496 2600]]	[[2555 540] [489 2607]]
Random Forrest	ADABoost
[[2659 436] [301 2795]]	[[2692 403] [361 2735]]

Looking at each model, we can see many differences in each of their results based on the metrics. The K-Nearest Neighbors model had good balance, with 84% accuracy on the test set

and similar scores for recall and precision. The Decision Tree had a test accuracy of 78% and a recall of 74%. However, it had a high precision of 88%, meaning that its predictions when a house was above the median price was usually correct. Random Forest performed well on the test set, achieving 88% accuracy with a recall of 90% and precision of 87%. However, we can see that on the training set, there could be a sign of overfitting as there are high values for both accuracy and recall. AdaBoost performed the best, with 88% accuracy and 87% precision score, showcasing its balance.

We can also compare the model performances further through the confusion matrices. The KNN and Decision Tree models both had similar values for false positives and false negatives. Random Forest had the fewest false negatives, meaning it was strong at detecting positive cases. Finally, we can see that ADABoost showed the best overall balance, with 2692 true negatives and 2735 true positives, which is consistent from the insights we stated above.

Conclusion

After testing each model, I would choose the AdaBoost Classifier for this dataset. It performed well across all metrics compared to the other models. Furthermore, I think accuracy would be the most important metric for this particular dataset. The metric precision tells us how many of the models predicted positive cases are correct. In this case, we want to confidently predict houses that are truly above the median price. False positives could lead to costly mistakes in the market, so it's important to ensure that the predicted high-value homes are actually priced above the median.