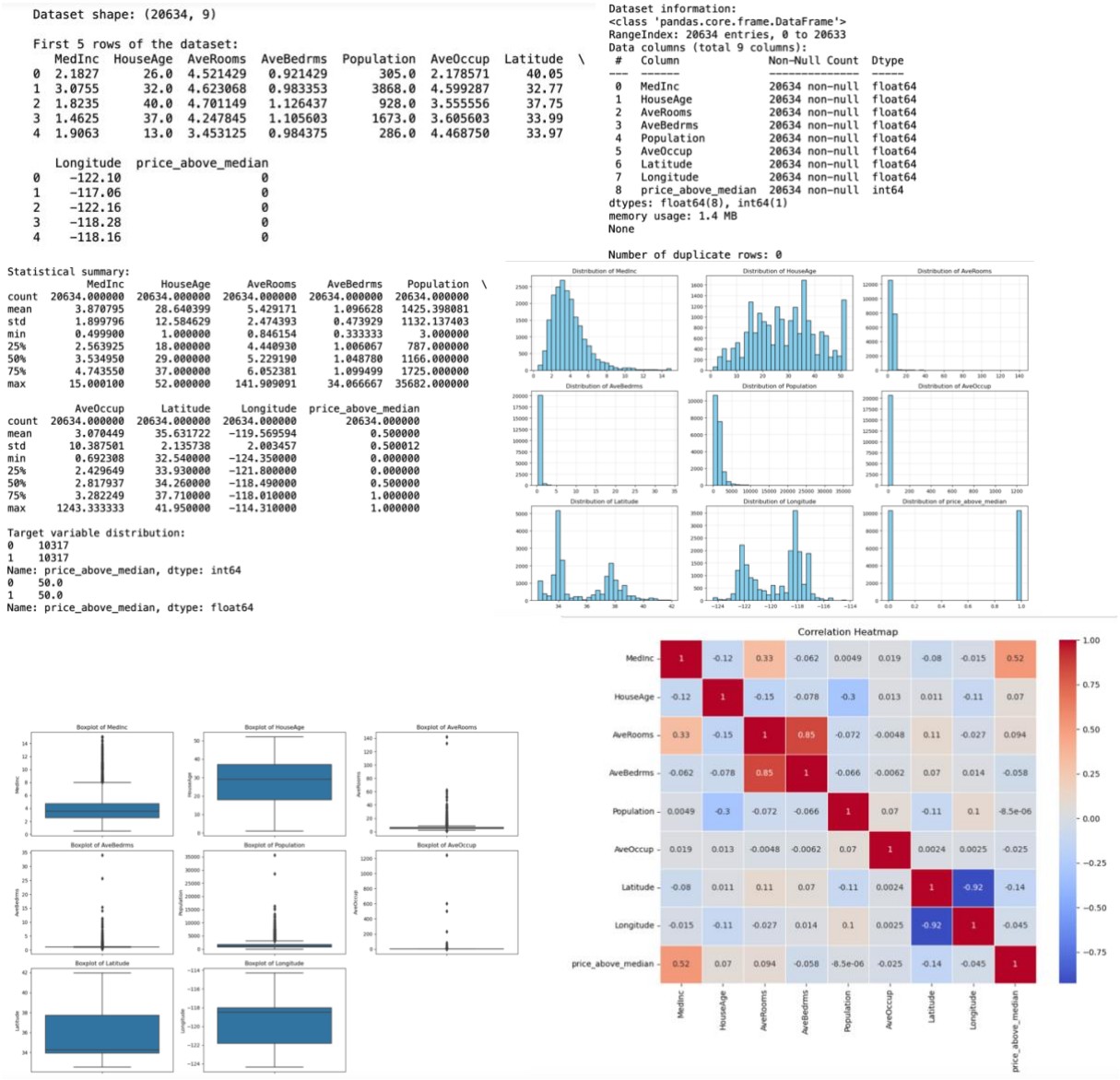
Part 1

The dataset contains 20,634 rows and 9 columns, including 8 features and 1 target variable. The features represent different characteristics of housing blocks in California, and the target variable indicates whether the house price is above the median.

All features in the dataset are of floating-point type, but the target variable is an integer type (0 or 1). The analysis shows no duplicate rows in the dataset, so no data cleaning was required.



```
Dataset shape: (20634, 9)

First 5 rows of the dataset:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  2.1827      26.0  4.521429   0.921429       305.0  2.178571     40.05
1  3.0755      32.0  4.623068   0.983353      3868.0  4.599287     32.77
2  1.8235      40.0  4.701149   1.126437       928.0  3.555556     37.75
3  1.4625      37.0  4.247845   1.105603      1673.0  3.605603     33.99
4  1.9063      13.0  3.453125   0.984375       286.0  4.468750     33.97

   Longitude  price_above_median
0    -122.10                   0
1    -117.06                   0
2    -122.16                   0
3    -118.28                   0
4    -118.16                   0
```

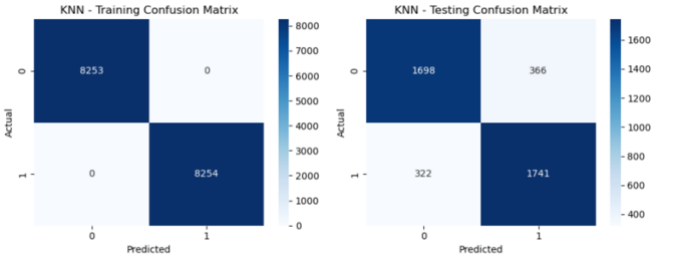
```
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20634 entries, 0 to 20633
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   MedInc              20634 non-null  float64
 1   HouseAge            20634 non-null  float64
 2   AveRooms            20634 non-null  float64
 3   AveBedrms           20634 non-null  float64
 4   Population          20634 non-null  float64
 5   AveOccup            20634 non-null  float64
 6   Latitude            20634 non-null  float64
 7   Longitude           20634 non-null  float64
 8   price_above_median  20634 non-null  int64
dtypes: float64(8), int64(1)
memory usage: 1.4 MB
None

Number of duplicate rows: 0
```

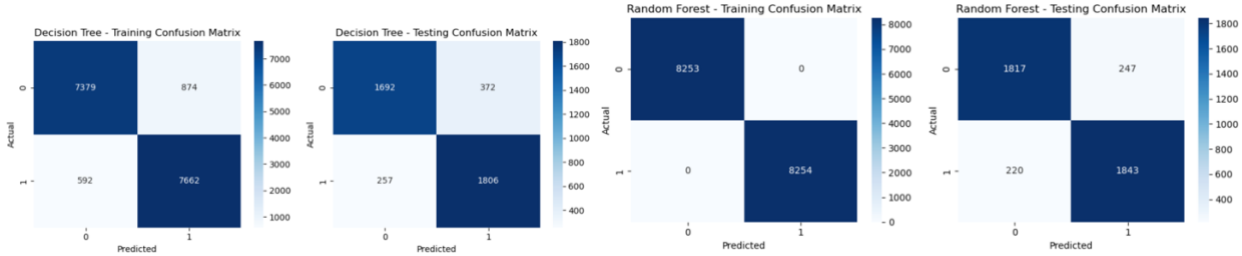
```
Statistical summary:
              MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20634.000000  20634.000000  20634.000000  20634.000000  20634.000000
mean       3.870795     28.640399      5.429171      1.096628   1425.398081
std        1.899796     12.584629      2.474393      0.473929   1132.137403
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563925     18.000000      4.440930      1.006067    787.000000
50%        3.534950     29.000000      5.229190      1.048780   1166.000000
75%        4.743550     37.000000      6.052381      1.099499   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

           AveOccup      Latitude     Longitude  price_above_median
count  20634.000000  20634.000000  20634.000000        20634.000000
mean       3.070449     35.631722   -119.569594            0.500000
std       10.387501      2.135738      2.003457            0.500012
min        0.692308     32.540000   -124.350000            0.000000
25%        2.429649     33.930000   -121.800000            0.000000
50%        2.817937     34.260000   -118.490000            0.500000
75%        3.282249     37.710000   -118.010000            1.000000
max     1243.333333     41.950000   -114.310000            1.000000

Target variable distribution:
0    10317
1    10317
Name: price_above_median, dtype: int64
0    50.0
1    50.0
Name: price_above_median, dtype: float64
```

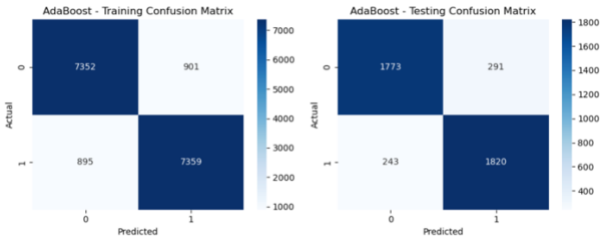
Part 2

The dataset was split into training and test sets. Training set: 16,506 instances (80%) and test set: 4,128 instances (20%). The class distribution was preserved in both sets, with exactly 50% of instances in each class. I used KNN, Decision Tree, Random Forest and AdaBoost classifier.

```
Training set shape: (16507, 8)
Testing set shape: (4127, 8)
Training set class distribution:
1    50.003029
0    49.996971
Name: price_above_median, dtype: float64
Testing set class distribution:
0    50.012115
1    49.987885
Name: price_above_median, dtype: float64
```








```
Model Performance Comparison:

Model Metrics Summary:
           Model  Train Accuracy  Test Accuracy  Precision    Recall  F1 Score
0            KNN        1.000000       0.833293   0.826293  0.843917  0.835012
1  Decision Tree        0.911189       0.847589   0.829201  0.875424  0.851686
2  Random Forest        1.000000       0.886843   0.881818  0.893359  0.887551
3       AdaBoost        0.891198       0.870608   0.862151  0.882210  0.872065

Best performing model based on test accuracy: Random Forest
```

Part 3

1.Which techniques did you use to train the models?

All models were trained using a stratified train-test split way to ensure balanced class representation in both sets.

KNN was used with distance-weighted voting and an optimal k=11 neighbors. Decision Tree classifier was used with parameters including max_depth=10 and min_samples_split=10 to control complexity. Random Forest combined 200 decision trees with a maximum depth of 20 to

improve predictive performance. AdaBoost was implemented with a learning rate of 1.0 and 200 estimators. And I asked chatgpt for helping me find the best ways to train the models.

2.Explain any techniques used to optimize model performance?

For hyperparameter tuning, I used Grid search CV with the help of chatgpt. And I also used Feature Standardization so that all features were standardized.

3.Compare the performance of all models to predict the dependent variable?

Random Forest has results with 88.68% test accuracy, AdaBoost has 87.06% , Decision Tree 84.76%, and KNN 83.33%. Random Forest also has 88.18% precision, 89.34% recall, and 88.76% F1 score.

Both KNN and Random Forest achieved 100% training accuracy, but Random Forest showed better generalization on test data. Decision Tree showed signs of overfitting with a notable gap between training and testing accuracies. AdaBoost has good generalization with minimal difference between training and testing performance. All models maintained balanced performance across both housing price classes.

4.Which model would you recommend to be used for this dataset?

Random Forest is recommended.

5.For this dataset, which metric is more important, why?

I think precision is more important because false classifying could lead to significant financial losses for buyers and investors.