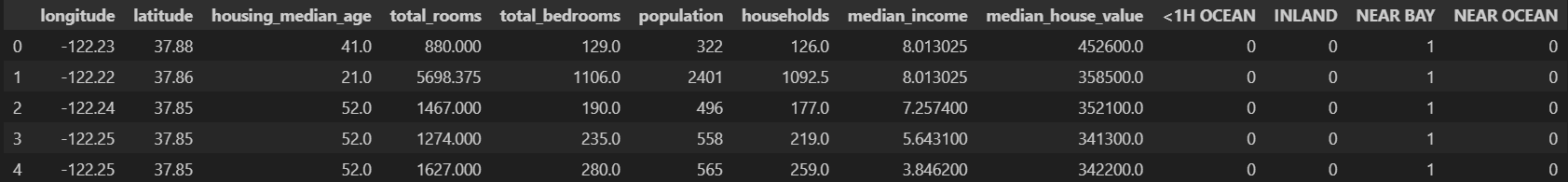
**Housing California Price**

Link in Kaggle:

[California Housing Prices (kaggle.com)](https://www.kaggle.com/datasets/camnugent/california-housing-prices)

1 Data Summary

Here we first summarize the California Housing dataset using visualization and some basic statistics. As showed in Figure 1California Housing dataset contains 20640 rows and each one of them stores information about a specific block. It contains 13 columns with 12 features and one target variable-median house value.



**About Dataset:**

1.**longitude**: A measure of how far west a house is; a higher value is farther west  
  
 2. **latitude**: A measure of how far north a house is; a higher value is farther north  
  
 3. **housingMedianAge**: Median age of a house within a block; a lower number is a newer building  
  
 4. **totalRooms**: Total number of rooms within a block  
  
 5. **totalBedrooms**: Total number of bedrooms within a block  
  
 6. **population**: Total number of people residing within a block  
  
 7. **households**: Total number of households, a group of people residing within a home unit, for a block  
  
 8. **medianIncome**: Median income for households within a block of houses (measured in tens of thousands of US Dollars)  
  
 9. **medianHouseValue**: Median house value for households within a block (measured in US Dollars)  
  
 10. **oceanProximity**: Location of the house w.r.t ocean/sea Acknowledgements

Inspiration

See my kernel on machine learning basics in R using this dataset, or venture over to the following link for a python based introductory tutorial: <https://github.com/ageron/handson-ml/tree/master/datasets/housing>

**Preprocessing steps:**

1. **A list of values in a table

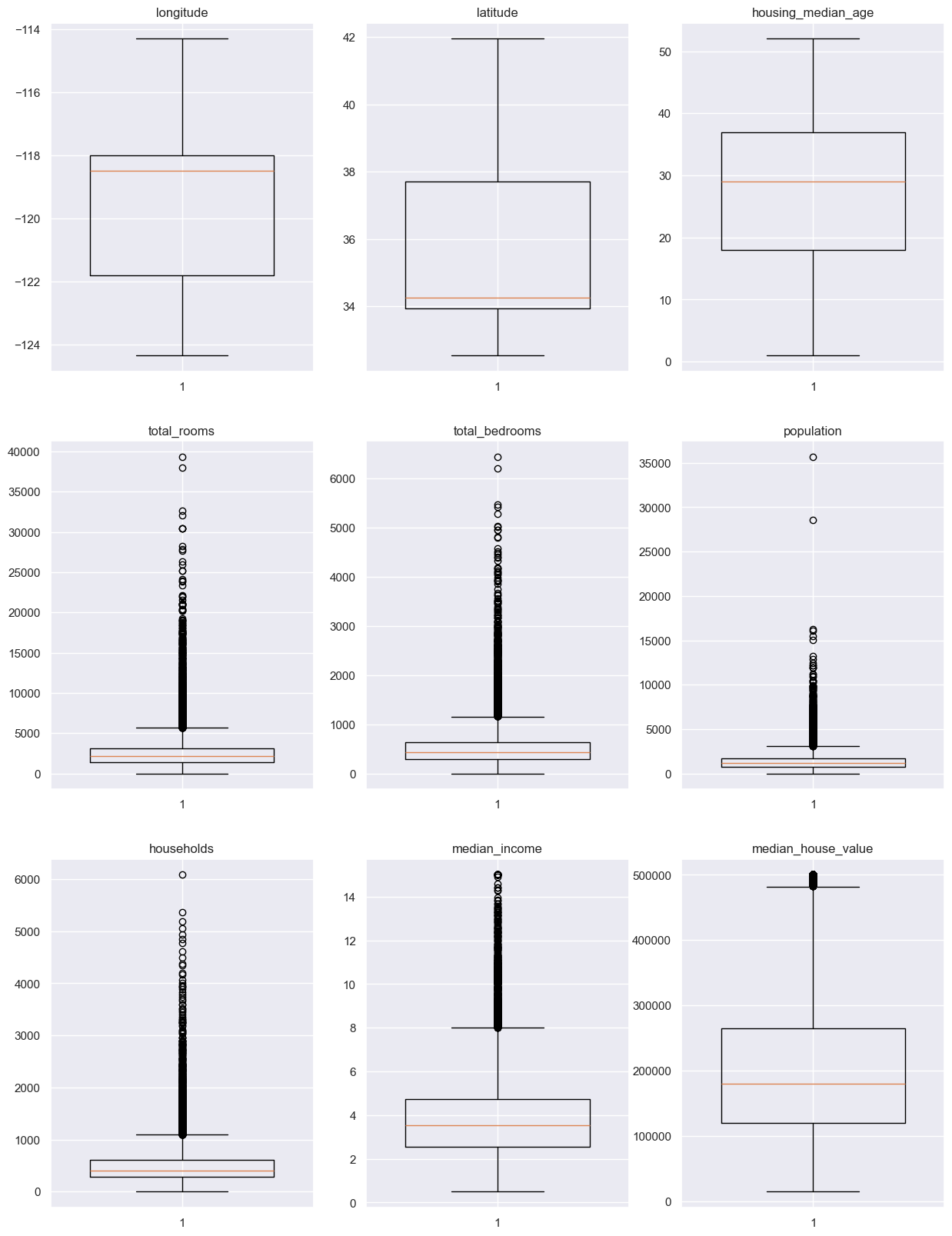
   Description automatically generated**Handling missing values:

* In feature total\_bedrooms there are missing values

A screen shot of a computer

Description automatically generatedTo handle this missing values we fill this missing values with mean for feature total\_bedrooms the code below explain how we do this:

1. handling Outliers with IQR method

* detected if their outliers using boxenplots so there the shape of some of columns before deleting outliers:
* And this is the shape after deleting.

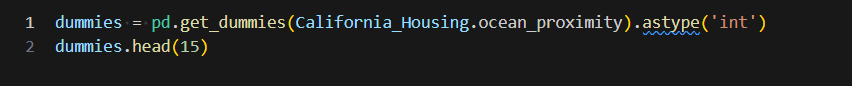
A screenshot of a graph

Description automatically generated

* Quantile based flooring and capping: In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value.
* A screen shot of a computer program

  Description automatically generatedWe use the Quantile based flooring and capping to handle the outliers the code below explain how we do that:

1. Handling Categoriacal values (using encoding)

* In machine learning, categorical values refer to variables that can take on a limited, and usually fixed, number of possible values. These values typically represent categories or labels, such as "red," "blue," "green" for a color variable. Handling categorical values in machine learning involves techniques such as one-hot encoding, label encoding, or using techniques like decision trees that can naturally handle categorical data
* The code below explain how we handling categorical values

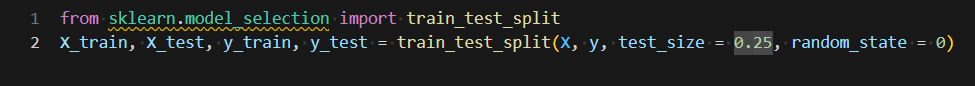
1. Feature Scalling

* In machine learning, feature scaling is the process of normalizing or standardizing the range of independent variables or features of the data. It is important because many machine learning algorithms perform better when the input numerical variables are scaled to a standard range. Common techniques for feature scaling include min-max scaling, standardization (Z-score normalization), and robust scaling. These techniques help to ensure that all features have a similar scale and do not disproportionately influence the learning algorithm.
* A black background with colorful text

  Description automatically generatedThe code below explain how to make feature scalling

1. Spliting Data to Train,Test

* Splitting data into training and testing sets is a fundamental practice in machine learning. This involves dividing the available dataset into two separate sets: the training set, which is used to train the model, and the testing set, which is used to evaluate the model's performance. Typically, a larger portion of the data is allocated to the training set (e.g., 75%), while the remaining portion is used for testing. This process allows for assessing how well the trained model generalizes to new, unseen data.
* The code below explain how we make splitting data to train and test data.



The data is ready to be in the models.

**a)** Linear Regression

**b)** Goal : **median\_house\_value**

**c)** Code : **used linear regression from sklearn.lineae\_model**

**d)** Result

A screen shot of a computer program

Description automatically generated

And this the plotting of the model:A diagram of a scatter plot

Description automatically generated

**a)**k Nearest Neighbors (KNN)

**b)** Goal : **median\_house\_value**

**c)** Code : **used** KNeighborsRegressor from sklearn. neighbors

A computer screen shot of a program

Description automatically generated**d)** Result

And this the plotting of the model:

A diagram of a scatter plot

Description automatically generated

According as using Metrics, score, plotting we get that using KNN is better than Linear Regression here.