California Median House value prediction's report

Group 09

PROFESSORS:

Saeed VARASTEH YAZDI, Antoine SCHERRER, Franck JAOTOMBO

MEMBERS:

Mohamed KANNOU, Qinyi MIAO, Nicholas SEBALD, Ruicong WANG, Fulin ZHANG

Contents

Introduction

This group project is to learn and explore the field of data science step by step with the help of the "California House Prices" dataset. The project was generally divided into three blocks. Firstly, descriptive statistics and data mining of the data were performed to find out the relationship between different variables. Univariate and multivariate analyses were performed sequentially. To explore the deeper connections below the surface of the data; in the second part we performed data cleaning. In this part we practiced the handling of null values and the efficiency of various methods of filling null values by creating nulls and exploring the goodness-of-fit, exploring useless columns and invalid data.

Then we proceeded with numerical treatment of categorical variables and normalization of numerical data. Finally, we fitted multiple models using the prepared data, evaluated the performance of the models using the test set, and compared the performance between the models. The next steps were to configure the models and optimize them with the help of cross-validation and GridSearchCV. We used the tuned hyperparameters to obtain the model with the best fit.

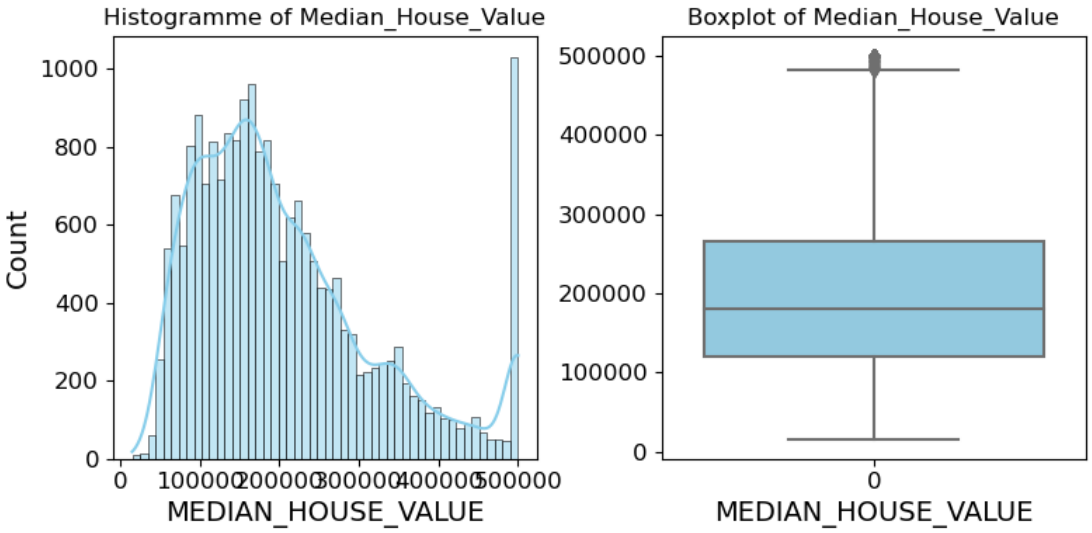
Data exploration

General insight

From the data exploration part, we plotted the histograms and boxplots together for the quantitative variables to see their trends, patterns and whether there are outliers. We will discuss the trends we found in this section.

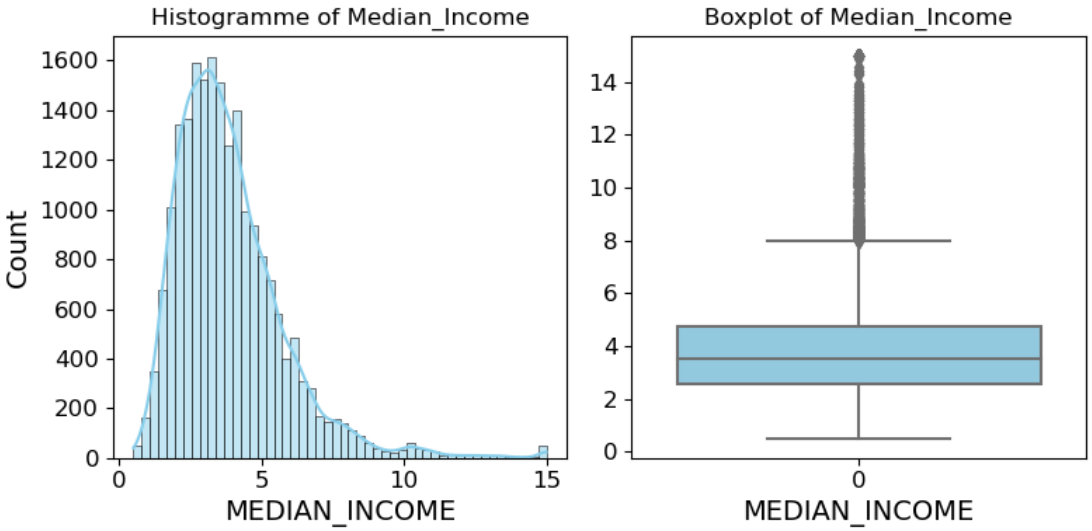
We found that the result of Figure 1.‘MEDIAN_HOUSE_VALUE’ showed a normal distribution, but from the 500,000$ there is a group of outliers, it’s easy to see that the last bar counts more than 1000 units, this outlier will increase the average level. We inferred that when collecting data, the house value worth more than 500,000$ will also be classified to this range, this greatly increases the height of the last bar.

Figure 1. Plotted Median House Value



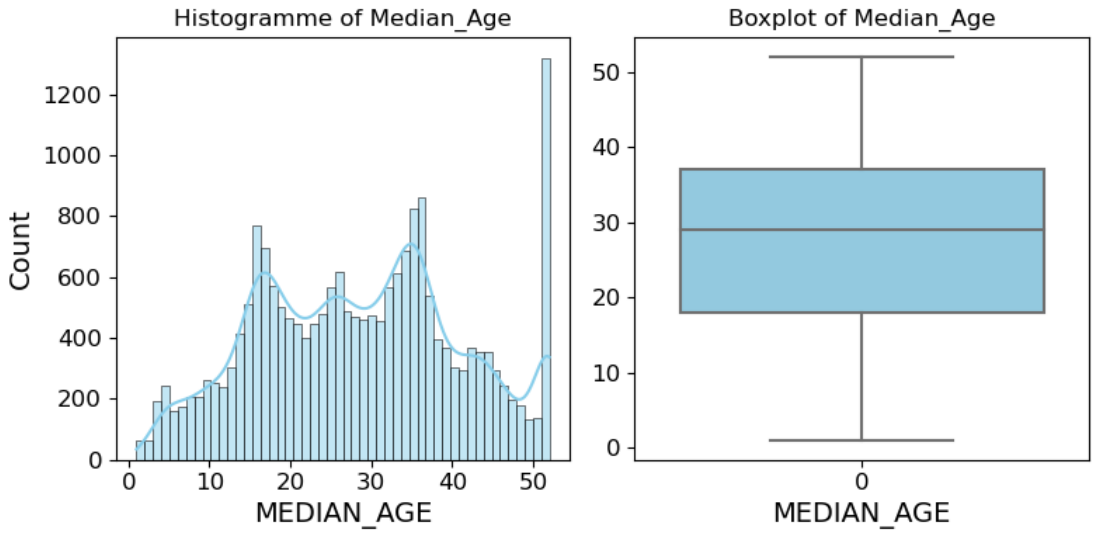
From the Figure 2.‘MEDIAN_INCOME’ variable, the median value is 3.5348. It means that half of the data is lower than this value, but if we observe the boxplot, the upper fence is 8, and there are many outliers higher than upper fence (even superior than 14). According to the statistical data from the previous step, we found that the median is inferior to the mean, so the distribution is skewed to the right.

Figure 2. Plotted Median Income



Then we take a look at Figure 3.'MEDIAN_AGE'. The histogram shows a 'multi-peaked' distribution. This type of distribution indicates that the data may have more than one cluster, most of them focused around 15 years, 25 years and 35 years. We see a similar trend to the 'MEDIAN_HOUSE_VALUE' variable where the last bar is extremely high. The reason may be the same as we mentioned in the 'MEDIAN_HOUSE_VALUE'. There is no outlier in the box plot.

Figure 3. Plotted Median Age



Next, we move into Figure 4.'TOT_ROOMS', 'TOT_BEDROOMS', 'POPULATION' and 'HOUSEHOLDS' these 4 variables, from the results of histograms, they have nearly the same shape and all right-skewed, it seems perfectly logical that it exists high correlation between 'TOT_ROOMS' and 'TOT_BEDROOMS', 'POPULATION' and 'HOUSEHOLDS', it can be confirmed in the heatmap on the following page. Please see the area circled in red.

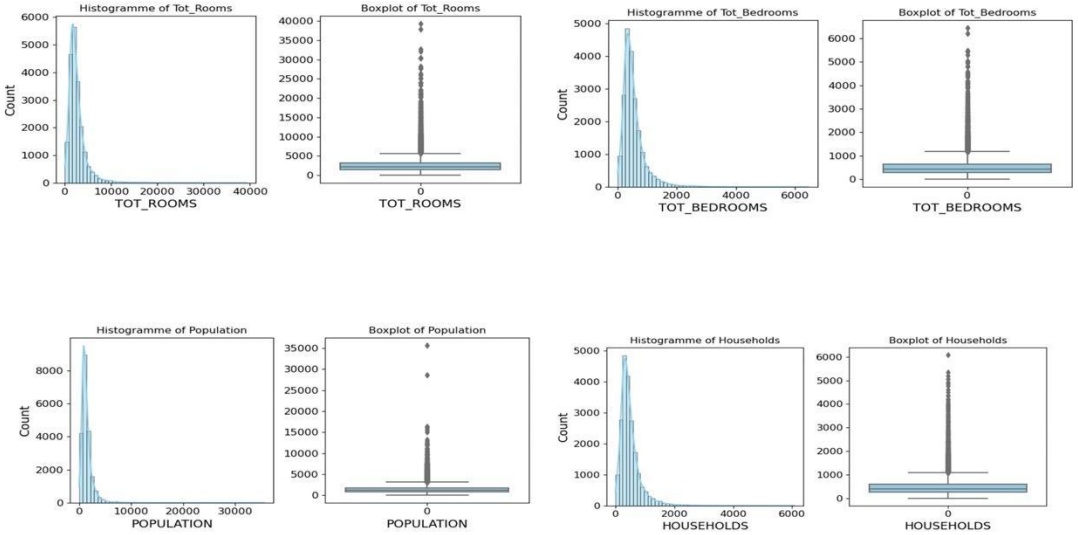
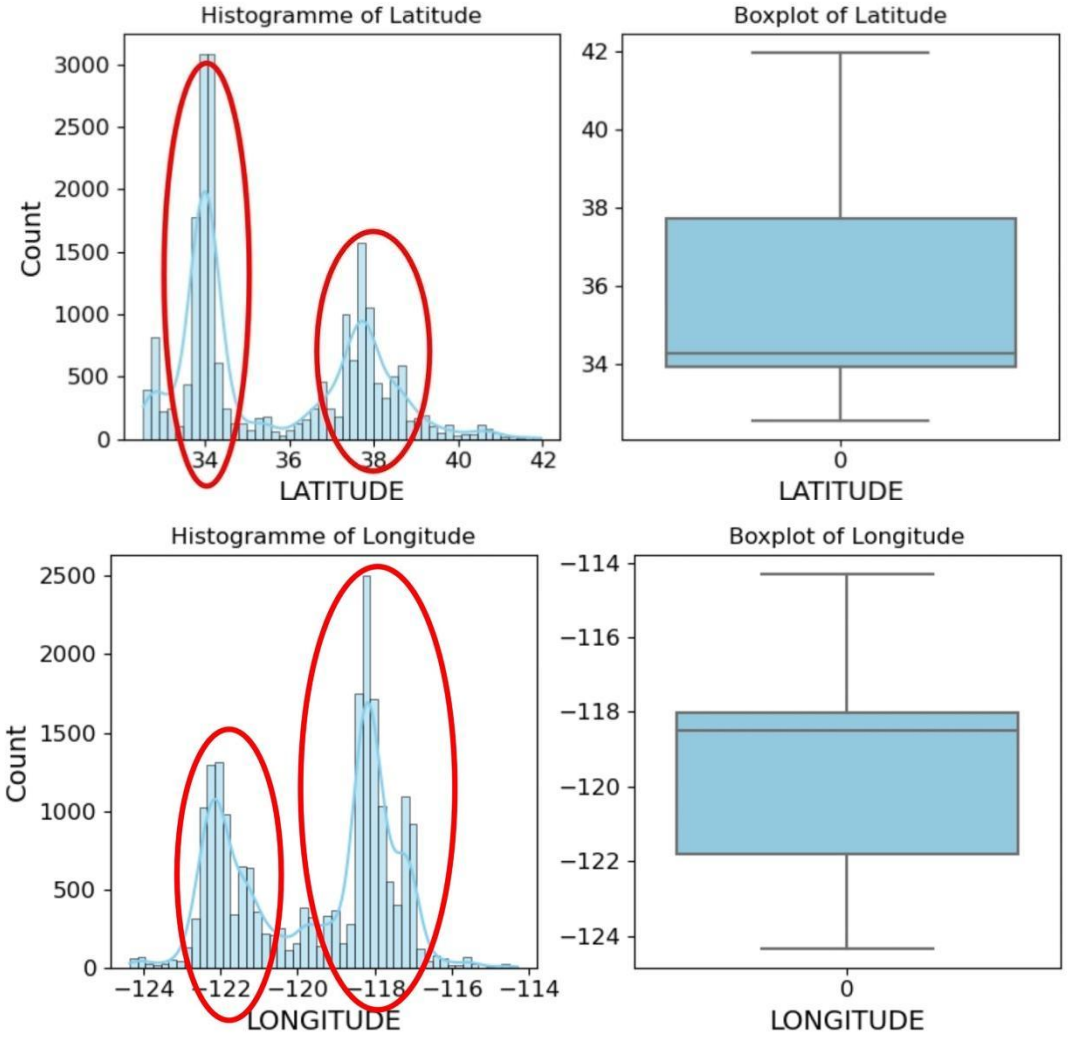
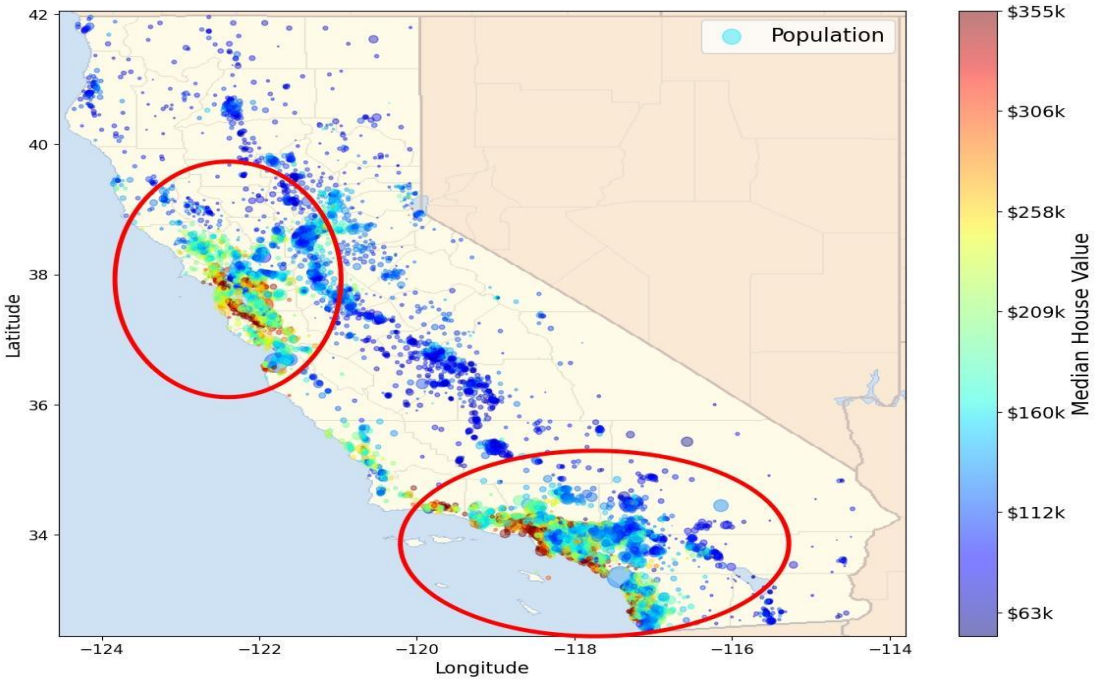


Figure 4. Plotted Tot Rooms, Tot Bedrooms, Population, Households

Next, we observe Figure 5.'LATITUDE' and 'LONGITUDE' columns, which are geographical data, and it's very useful when we plot the color scatter plot on the California map in the following step. This data can facilitate the visualization of the median house value on the map. The distribution of latitude and longitude are bimodal distributions; there are two distinct peaks shown in histogram. When we put them together with the California scatter map, obviously, the high median housing values are concentrated in these two peaks, which coincide with California's more metropolitan cities.

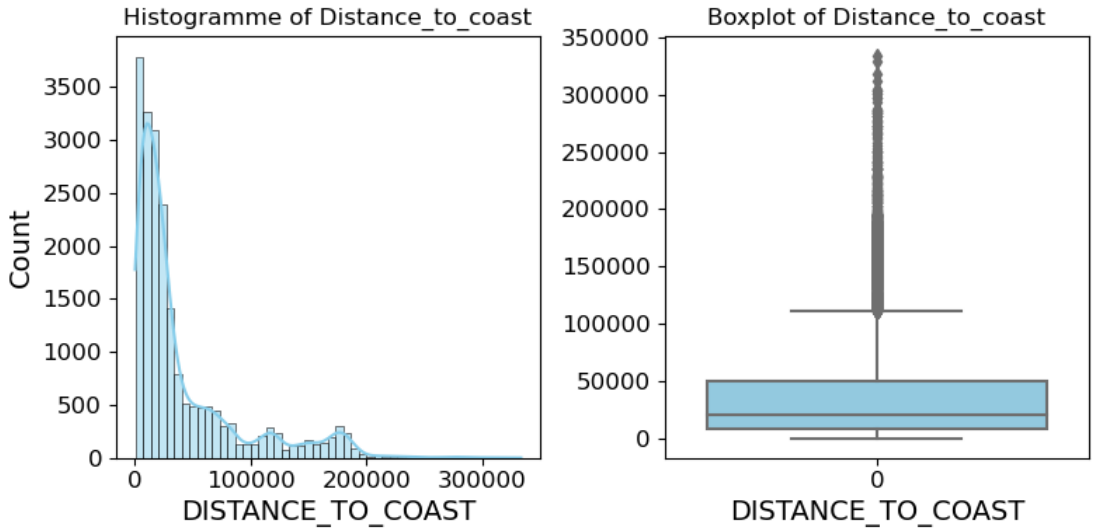
Figure 5. Plotted Latitude, Longitude





Lastly, Figure 6. 'DISTANCE_TO_COAST' variable shows that most of the data represents districts near the coast. The nearest one is only 120 units and we can also find many outliers from 100,000 to 333,804. You can see this distribution in the California mapped scatter plot.

Figure 6. Plotted Distance To Coast

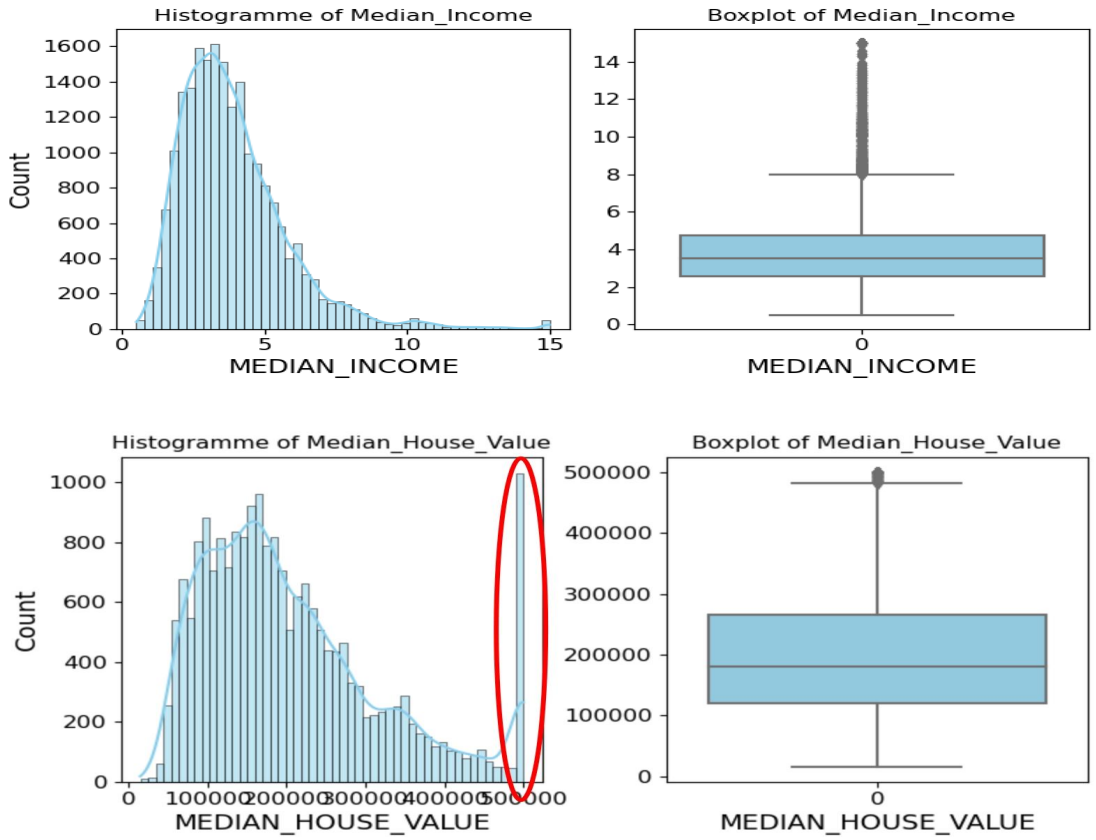


Exploring 'Median_Income'

We then focus more on Figure 2. 'Median income'. We chose a higher bin count to explore details of 'Median income'. The higher bin count gives a more accurate representation of the data. We decided to see more details when 'MEDIAN_INCOME' is more than 14 and

'MEDIAN_HOUSE_VALUE' superior is more than 499,999, because in the previous step we observed the last bar is extremely high. We found that there are 992 units superior to these values, which represents nearly 5% of the total data.

Figure 7. Median Income & Median House Value



Data Cleaning

Handle missing values

To handle missing values, we create a dataset where there are 10% of missing values in one variable. When we have missing values, there are two main possibilities: we simply drop the rows associated with the missing value or we estimate the missing values through an imputation method - the simplest and safest is to use the median. So we creates a new DataFrame called “houses_drop” by dropping rows from the “houses_miss” DataFrame where the “Tot_Bedrooms” column has missing values; and then calculate the median value of the “Tot_Bedrooms” column from the original “houses_miss” DataFrame and assigns it to the variable “Bed_med”, and then imputes the missing values in the “Tot_Bedrooms” column of the “houses_miss” DataFrame with the value stored in the “Bed_med” variable which is the median value calculated in the previous step.

We looked at two options to handle missing values in our data set. First we used SimpleImputer to compute the mean for each column, transform the DataFrame by replacing missing values with the mean, the result of the transformation is a NumPy array, so convert it back to a DataFrame. Now, housing_imputed_df contains the imputed data with missing values replaced by the mean. Second, we use KNN Imputer, impute missing data in each variable. Compare these two Imputes, we find KNN is better. We separate the DataFrame houses_df into two DataFrames, “houses_df_cat” containing non-numeric columns and “houses_df_num”containing numeric columns. Then we calculate the mean squared error for the numeric columns between the original data “houses_df_num” and the “housing_miss_simple” DataFrame using the ‘compare_df’ function. Then, we calculate the mean of these errors using NumPy's ‘np.mean’ function and assign it to the “mse_mean_simple”variable. Here is the same for the KNN.

Stratifying data

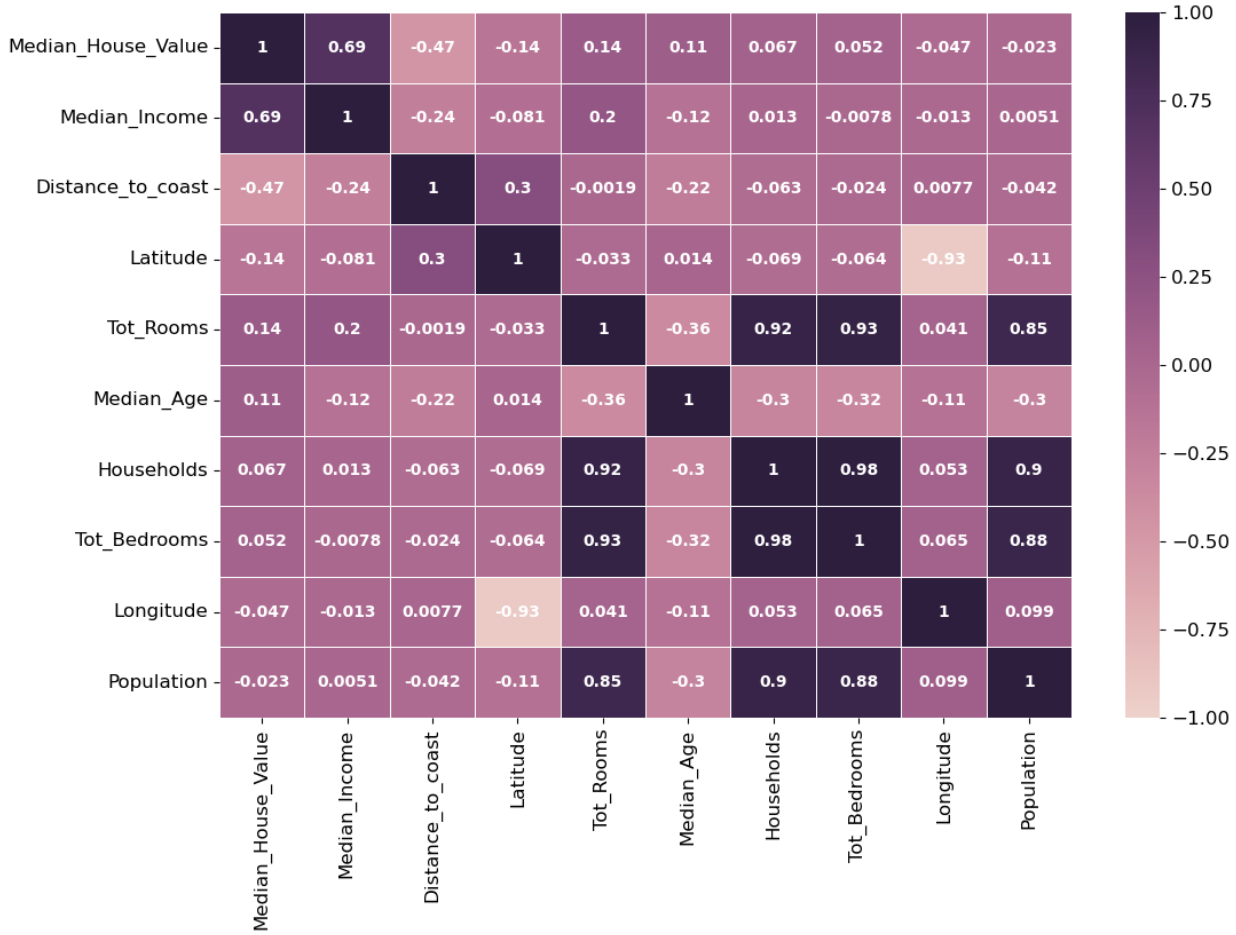
We create a list “cat” that finds the quantile values [0.20, 0.40, 0.60, 0.80] of “Median_Income”. After that we create a new categorical feature called (income_cat) based on the “Median_Income” column in the DataFrame housing. We use “cut ()” function to categorizes the values in the "Median_Income" column into “housing”; in this case “cat” is a list that contains the bin edges, which are the minimum value, four quantiles (20th, 40th, 60th, and 80th percentiles), and the maximum value of the "Median_Income" column. The “income_cat” column is then used to stratify the data to make sure that we have an accurate representation in the training and test sets.

Bivariate Analysis

Our goal of the analysis of the dataset “Housing Prices California” is to to get an understanding of the factors that affect houses’ values throughout the state of California.

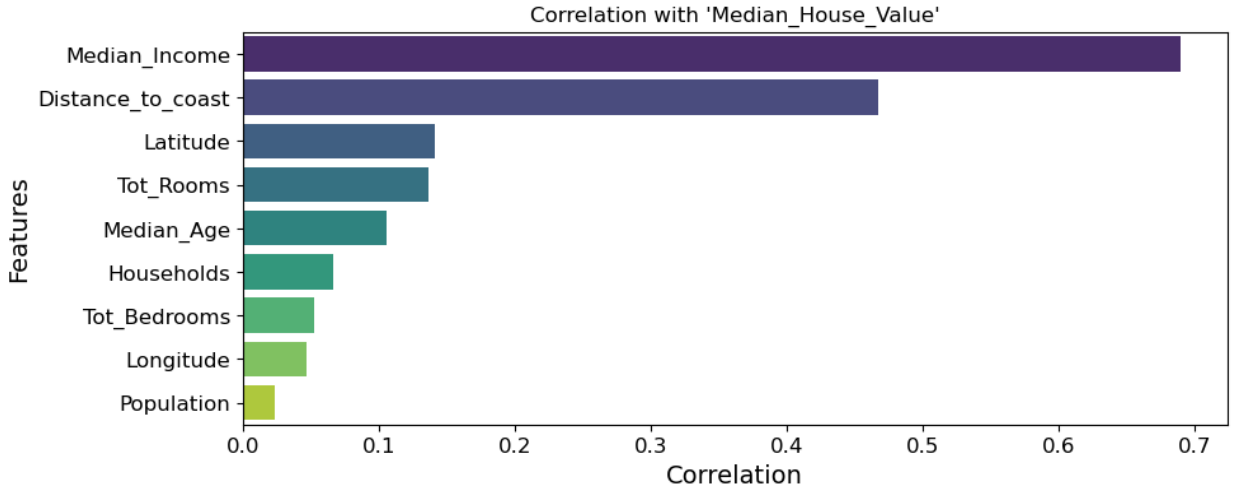
Each row of the dataset represents information on attributes (variables) of a specific district in California.

Heatmap figure displaying the correlation matrix between all variables in dataset:



⇨ The variable Median_House_Value describes the median house price in a district, thus making it the ideal target variable for our analysis.

Bar chart representing correlation between Median_house_value and other variables:

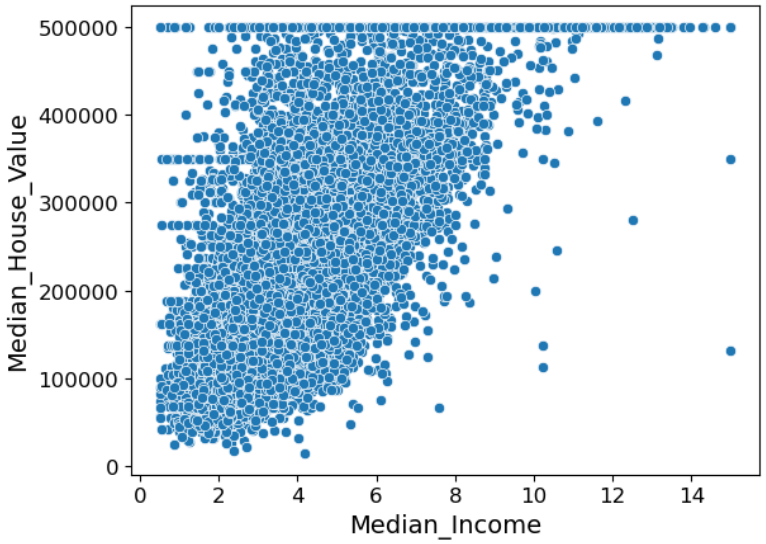


⇨ Median_Income correlation coefficient with Median_House_Value is equal to 0.69, and Distance_To_coast correlation coefficient is 0.47, almost 0.5. Since the latter two values are between 0.5 and 0.7, we can deduct that these variables are moderately correlated with our target variable Median_House_Value.

⇨ All other variables have a correlation coefficient lower than 0.3, therefore they have little if any correlation with Median_House_Value.

1. scatter plots depicting the relationship between median_income and Median_house_value, and distance_to_coast with Median_house_value:

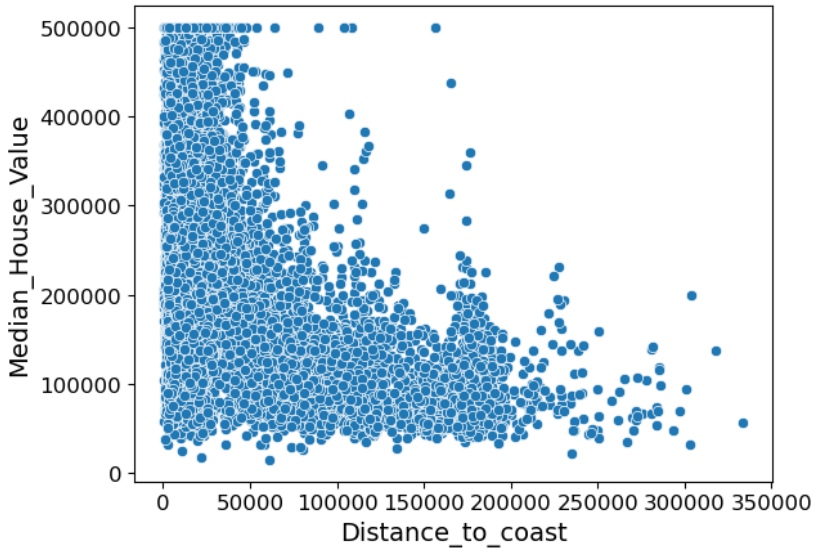
a. Scatter plot 1: Median_income vs Median_house_value



⇨ We see a positive relationship as Median_House_Values increases as Median_Income increases.

⇨The majority of observation are in the range between 0 and 10 in the Distance_to_coast axis

b. Scatter plot 2: Distance_to_coast vs Median_house_value



⇨ We see a negative relationship as Median_House_Value decreases as Distance_to_coast increases.

⇨We also see a substantially higher number of observations in the range between 0 and 150000 on the Distance_to_coast axis.

Machine Learning - Modeling

We will discuss the portion of the project that relates to configuring the machine learning (ML) models. In Part 3 we determined our target outcome and chose the model that we would like to hypertune at a later stage. The data had to go through one last round of preparation to be used in a ML model.

“Median House Value” was chosen as our target outcome. The target was chosen for a couple reasons. Primarily, “Median House Value” is useful information. Predicted median house values could be useful for a future home buyer, for example. Also, the variable analytics (discussed in “Data Exploration”) showed statistical significance with the other variables. Therefore, the selected target seemed suitable to pursue further.

We assessed different ML regression models, to determine which model had the lowest error. We employed root mean squared error (RMSE) as a benchmark between models. The results will be shown later in this section.

We first split the data into a training and a test set. The data was partitioned with 80% allocated to the training set and 20% for the test set. The data was split with stratification in mind to make sure that each set accurately represented the data as a whole. The stratified groups were based on the income category (“income_cat”) which ranged from 1-5. Please reference “Part 1 - Data explorations” for the code.

The next step was dealing with missing quantitative values and numerically representing the categorical columns for closest city (“Closest_city”) and income category (“income_cat”). These steps were necessary to prepare the data for the ML models. Missing values were resolved by an imputer (SimpleImputer), which replaced missing values with the median values of each column. The imputer could be explored further, such as we did in the “Data Cleaning” portion, comparing SimpleImputer and KNNImputer. Categorical values were one hot encoded. We set up a pipeline to streamline the process. Please reference “Part 2.2.4 - Pipeline for the quantitative and categorical variables”.

Once the data was ready, we fit the data using 5 different models: LinearRegression, Penalized Linear Regression (Elasticnet), DecisonTreeRegressor, RandomForestRegressor and Support Vector Regression (SVR). Each model was also checked across a 10-fold cross-validation. The root mean squared error was evaluated for each model to determine which provided the best fit.

The results were as follows:

Table 1. ML Model Error

Model	Error (RMSE)
Linear Regression	72482.8
Elasticnet	79709.0
DecisonTreeRegressor	71064.8
RandomForestRegressor	**54751.1**
SVR	110311.8

The findings reveal that the RandomForestRegressor stands out as the most robust model, yielding the lowest error with an RMSE of 54,751.1. Considering that the mean value of "Median_House_Value" is 206,855, this RMSE represents a significant deviation, approximately 26% of the mean value. Nonetheless, it's worth noting that the error is approximately 23% lower than the next best model, as represented by the DecisionTreeRegressor.

The model parameters were then lightly hypertuned using RandomSearchCV. Two parameters were checked with each model. TRandomForestRegressor came out on top with a MSE of 2977289233.01. The ideal parameters were 15 max features and 100 n-estimators. The results were manually converted to RMSE by taking the square root and displayed in the table below. The table shows the comparison of pre and post hypertuning and the best parameters.

The results of model scores after hyperparameter tuning are as follows:

Table 2. ML Model Score

Model	GridSearch CVError (RMSE)	Original Score	Error Reduction	Best Parameters
Elasticnet	72565.3	79709.0	9.0%	{'alpha': 46.41588833612782, 'l1_ratio': 1.0}
DecisonTreeRegressor	60975.6	71064.8	14.2%	{'min_samples_split': 9, 'min_samples_leaf': 20}
RandomForestRegressor	**54564.5**	**54751.1**	**0.3%**	**{'max_features': 15, 'n_estimators': 100}**

We tested the models on the test data and received the following errors:

Table 3. Errors from Test Data

Model	Error (RMSE)
Elasticnet	69036.82
DecisonTreeRegressor	69129.27
RandomForestRegressor	**66893.57**

Conclusion

The error is significant to the point where hyperparameter tuning will not resolve the issue. We could keep trying new models to see if we can find a better option. We can also take another look at the data. We did not adjust for the capped data that can be seen in the “Median_House_Value” histogram (Figure 1 - Data Exploration). The house values around 500,000 have an abnormally large quantity that could throw off the model. We could potentially treat this using some form of imputation that would project a more realistic house value given the other variables. We also could have used a better imputation method for the missing values. We ended up using SimpleImputer where a KNNImputer could have worked better. There is also the possibility that the data was too complex and it was difficult for a model to find a proper fit with the data.

In general, we learned how to take a data set and analyze the data for correlation between variables (quantitative and categorical). We used the correlations to help us determine a useful target variable. We used various methods to treat the data prior to modeling, such as filling in missing values and one hot encoding categorical variables.

We found that setting up a pipeline and using tools, such as GridSearchCV, are much more efficient methods in treating the data and checking for best model parameters. We learned that certain models will work better than others and hypertuning can vary in the effect it will have on improving the model’s performance. In the end, the error doesn’t come down to just using the right model and finding the best parameters. It is important to analyze the data properly and prepare it well for the model. It is an iterative process to determine how to reduce model errors. But most importantly, you must try to have fun while doing so.