Springboard - DSC Capstone Project III

California Housing Price Prediction

Final Report

Akbanu Tleubayeva

Data Science Career Track

January, 2022

Table of Contents

1. Introduction

1.1 Objective

Background of Problem Statement :

The US Census Bureau has published California Census Data which has 10 types of metrics such as the population, median income, median housing price, and so on for each block group in California. The dataset also serves as an input for project scoping and tries to specify the functional and nonfunctional requirements for it.

Problem Objective :

The project aims at building a model of housing prices to predict median house values in California using the provided dataset. This model should learn from the data and be able to predict the median housing price in any district, given all the other metrics.

Districts or block groups are the smallest geographical units for which the US Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). There are 20,640 districts in the project dataset.

Domain: Finance and Housing

1.2 Significance

This report is divided into the following sections:

- Section 2: Dataset
- Section 3: Package Introduction
- Section 4: Data Wrangling
- Section 5: Exploratory Data Analysis

• Section 6: Machine Learning
• Section 7: Final Model Selection and Hyperparameters Tuning
• Section 8: Conclusion

Analysis Tasks to be performed:

1. Build a model of housing prices to predict median house values in California using the provided dataset.
2. Train the model to learn from the data to predict the median housing price in any district, given all the other metrics.
3. Predict housing prices based on median_income and plot the regression chart for it.

2. Dataset

2.1 Data Description

The datasets are sourced from the website kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners that allows users to find and publish datasets, explore and build models in a web-based data-science environment.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Content

The data pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data. The columns are as follows, their names are self-explanatory:

Data consists of 20640 rows and 10 features:

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

median_house_value is our target feature, we will use other features to predict it.

The task is to predict how much the houses in particular block cost (the median) based on information of blocks location and basic socio demographic data.

Source

This data was entirely modified and cleaned by me and kagglers. The original data (without the distance features) was initially featured in the following paper:

Pace, R. Kelley, and Ronald Barry. "Sparse spatial autoregressions." Statistics & Probability Letters 33.3 (1997): 291-297.

The original dataset can be found under the following link:

https://www.kaggle.com/fedesoriano/california-housing-prices-data-extra-features

2.2 Dataset Characteristic

Dealing with missing values

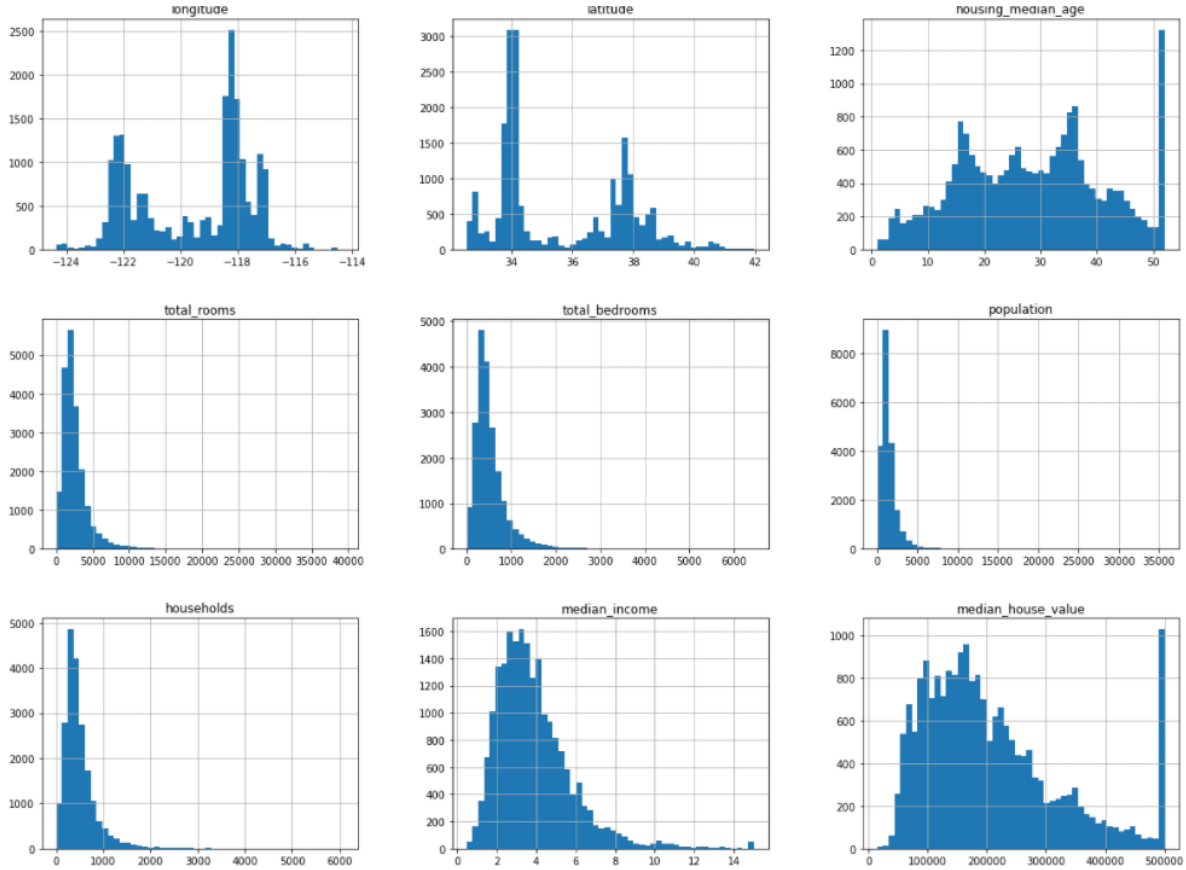
This data contains numerical and categorical columns. i.e in some columns the order of the categories is significant and in others there is no meaning to the order or the quantity of the value.

The numerical columns in our data are : housing_median_age, total_rooms, total_bedrooms, population, households, median_income, median_house_value.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Column `total_bedrooms` seem to have about 200 missing values; `ocean_proximity` is not numerical data.

Distribution of Data



- According to the pictures, these attributes have very different scales.
- The housing_median_age and the median_house_value were capped. The median_house_value may be a serious problem since it is the label to predict. The Machine Learning algorithms may learn that prices never go beyond that limit. We need to check to see if this is a problem or not. If precise predictions even beyond 500,000 is needed, then we have two options:
    - Option 1: Collect proper labels for the districts whose labels were capped.
    - Option 2: Remove those districts from the dataset.
- Many attributes are right skewed. This may make it a bit harder for some Machine Learning algorithms to detect patterns. We will try transforming these attributes to have more bell-shaped distributions.

3. Package Introduction

In this study we used JupyterLab (2.1.5) to run all the code. Numpy (1.18.5), Pandas (1.0.5), Matplotlib (3.2.2), Seaborn (0.11.0) were installed as the basic package. Scikit-learn (0.23.1) was installed as the machine learning library. Imblearn (0.7.0) was installed for data oversampling. Scikit Plot (0.3.7) was installed for plotting Cumulative Lift.

4. Data Wrangling

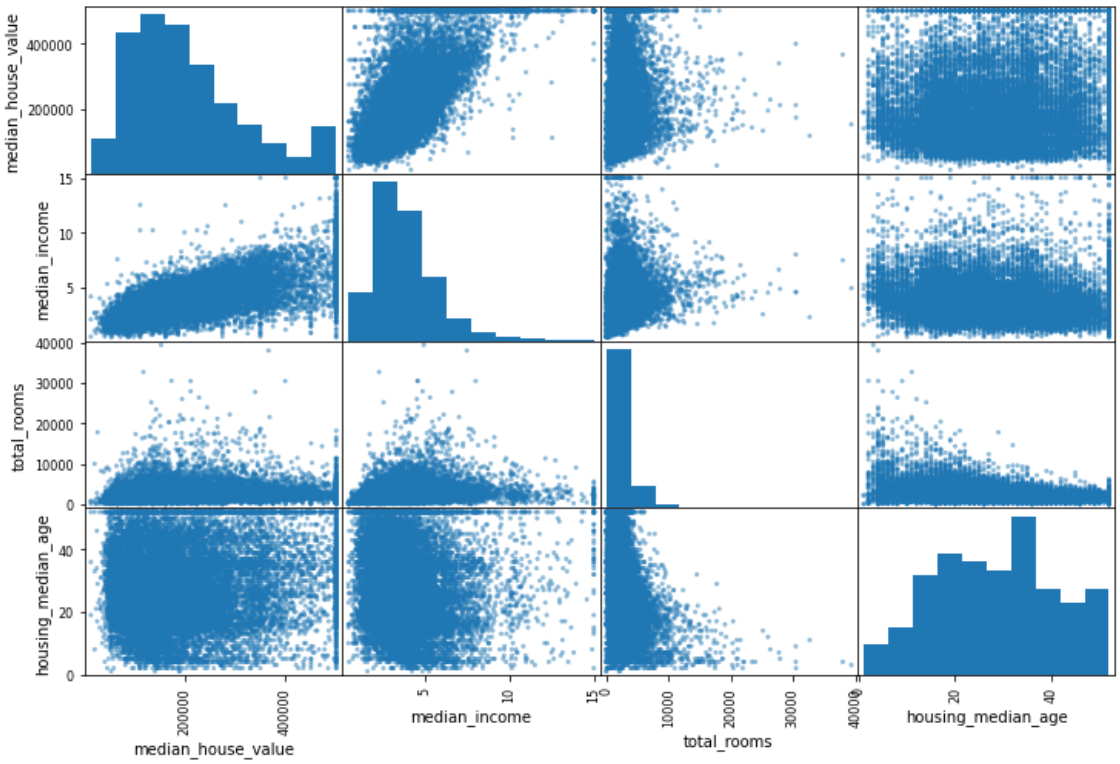
4.1 Dataset Information

First, I visualized the relationships between each variable to the other variables in the data. Then I used methods like head(), describe(), info() ect. to display the first five rows of each data set, display basic statistical details and general information about the data.

Afterwards I visualized the missing data in order to decide which columns to drop and how to deal with the missing values.

4.2 Data Processing

Let's have a closer look at dependencies between some numerical features.



We can see that on any local territory (you can play with local_coord and euc_dist_th) the linear dependencies between variables became stronger, especially median_income_log / median_house_value_log. So the coordinates are a very important factor for our task.

5. Exploratory Data Analysis

5.1 Summary Statistic

The statistics including mean, standard deviation, minimum values, maximum values and percentile for each variable were summarized.

Split the data

Scikit-Learn provides a few functions to split datasets into multiple subsets in various ways. The simplest function is train_test_split(), which provides a couple of additional features.

First, there is a random_state parameter that allows us to set the random generator seed. Second, we can pass it multiple datasets with an identical number of rows, and it will split them on the same indices (this is very useful, for example, if we have a separate DataFrame for labels.

```
test_set.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	359.0	1.6812	47700.0	INLAND
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	584.0	2.5313	45800.0	INLAND
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	963.0	3.4801	500001.0	NEAR BAY
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	495.0	5.7376	218600.0	<1H OCEAN
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	428.0	3.7250	278000.0	NEAR OCEAN

```
train_set.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	623.0	3.2596	103000.0	NEAR OCEAN
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	756.0	3.8125	382100.0	NEAR OCEAN
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	336.0	4.1563	172600.0	NEAR OCEAN
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	355.0	1.9425	93400.0	NEAR OCEAN
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	380.0	3.5542	96500.0	INLAND

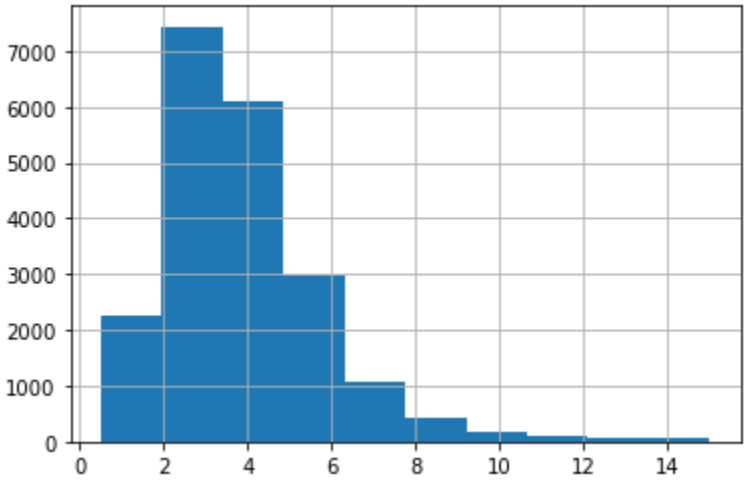
5.2 Overall Distribution

The overall distribution of numerical variables was visualized.

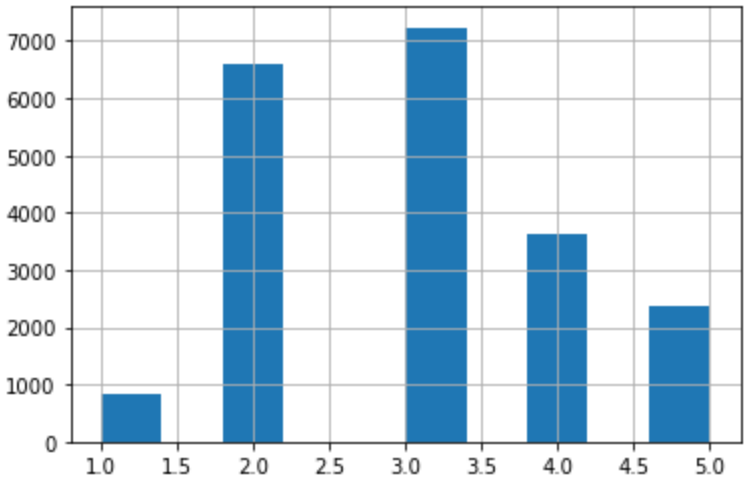
So far we have considered purely random sampling methods. This is generally fine if the dataset is large enough (especially relative to the number of attributes), but if it is not, will face the risk of introducing a significant sampling bias.

Suppose median_income is a very important attribute to predict median housing prices. We want to ensure that the test set is representative of the various categories of incomes in the whole dataset.

Since the median_income is a continuous numerical attribute, we first need to create an income category attribute.



5.3 Distribution Median Income Category.



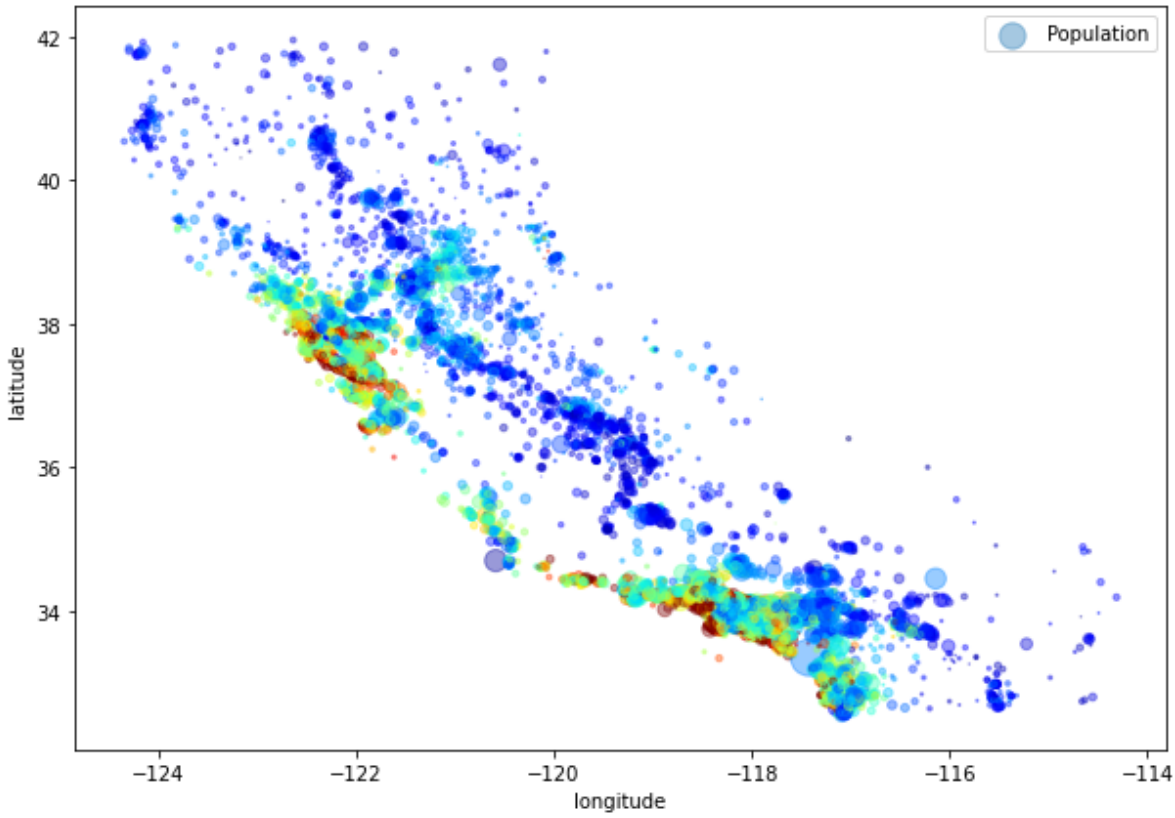
According to the analysis,

5.4 Split the Data

We can take a look at the comparison of stratified sampling and random sampling.

	Overall	Stratified	Stratified1	Random
1	0.039826	0.039729	0.039729	0.040213
2	0.318847	0.318798	0.318798	0.324370
3	0.350581	0.350533	0.350533	0.358527
4	0.176308	0.176357	0.176357	0.167393
5	0.114438	0.114583	0.114583	0.109496

5.5 Gain Insight

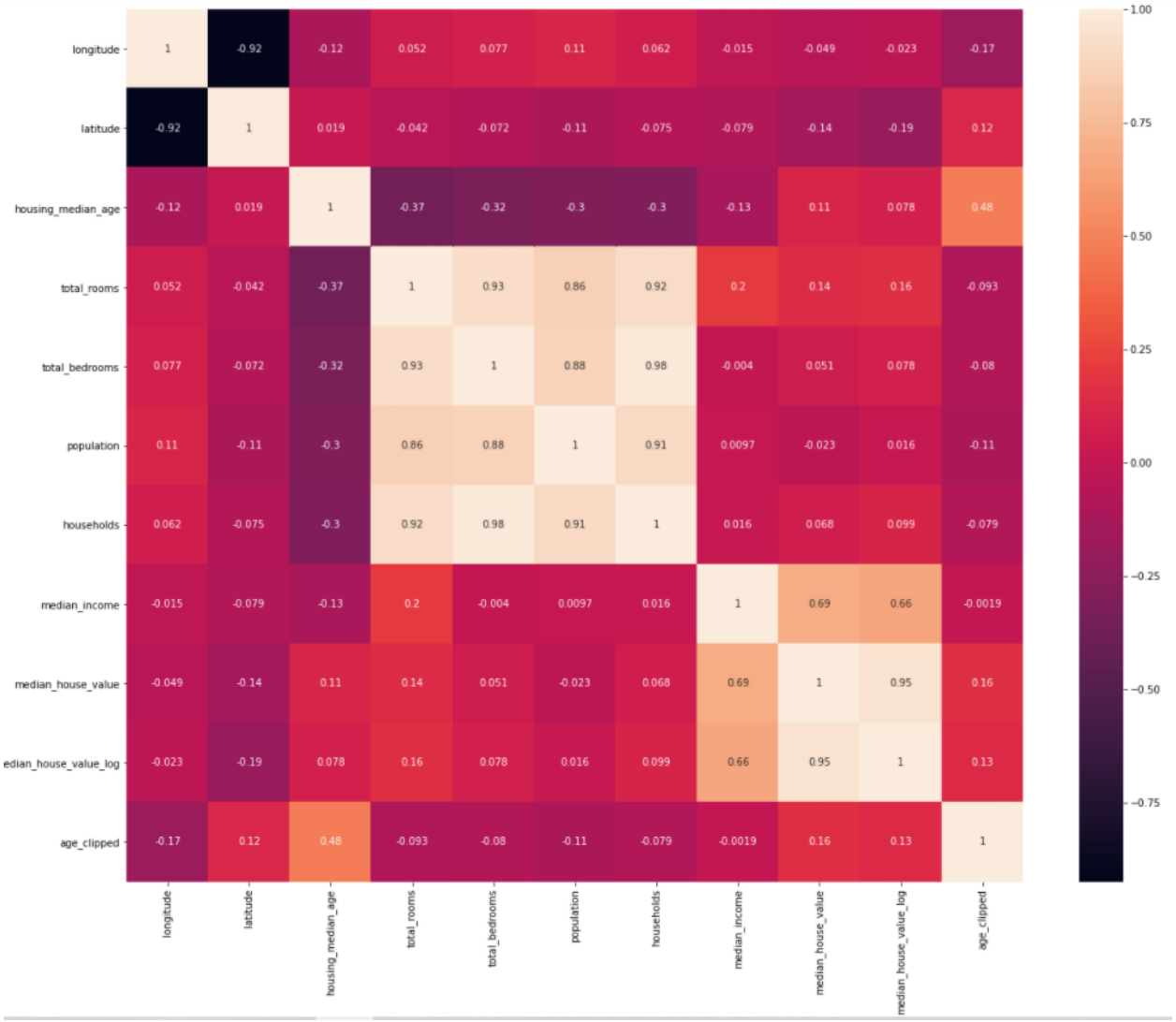


This image tells that the housing price is very much related to the location and to the population density.

From the above values we can see that the median_income has the highest linear correlation with the media_house_value. The correlation close to zero means that there's no linear correlation with that attribute.

5.6 Variable Correlation Coefficient

We will analyze correlation between features and target variable



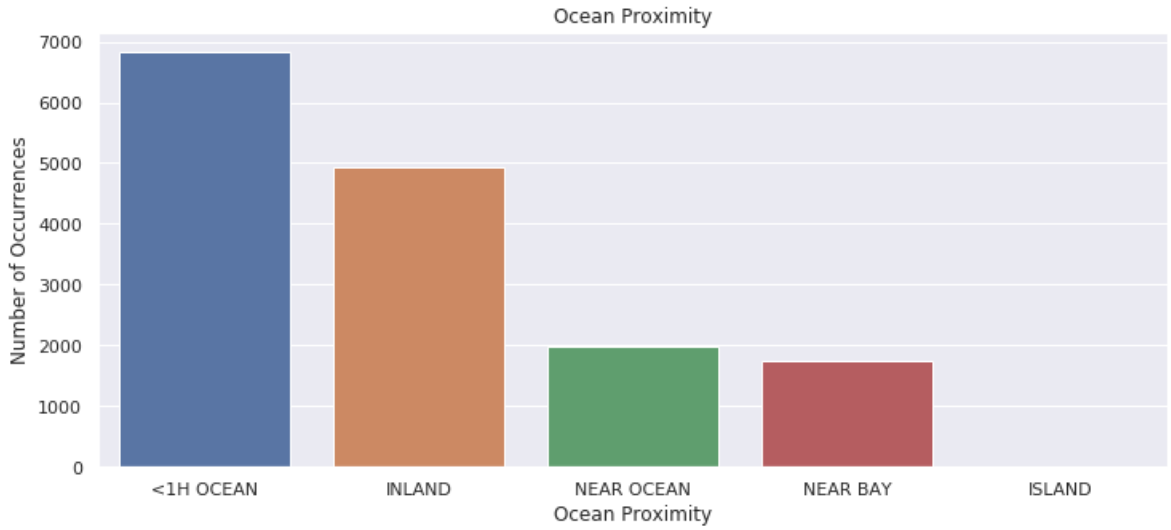
We can see some (maybe obvious) patterns here:

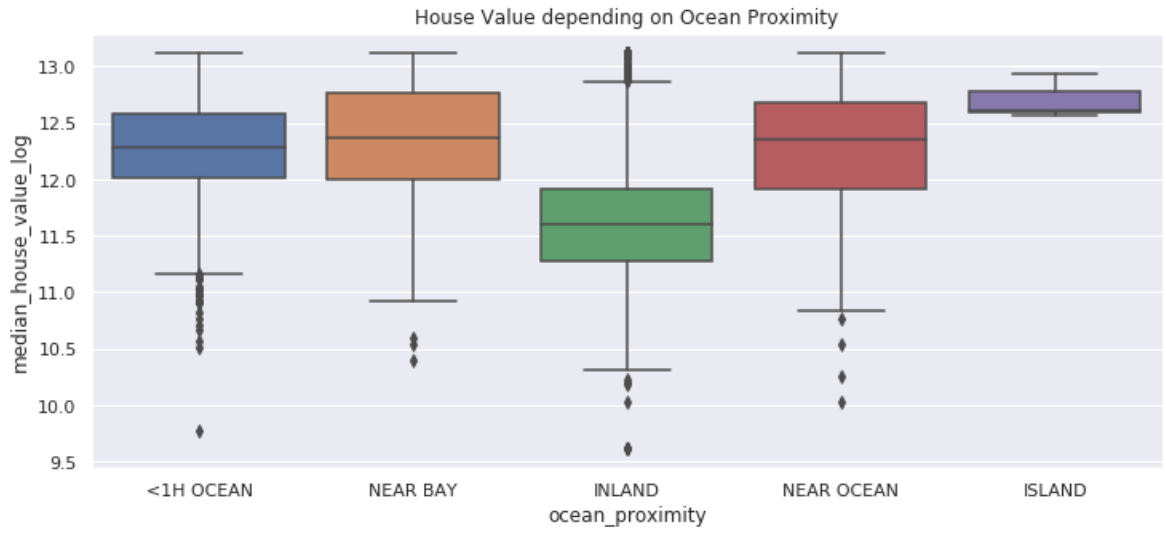
- House values are significantly correlated with median income
- Number of households is not 100% correlated with population, we can try to add average_size_of_household as a feature
- Longitude and Latitude should be analyzed separately (just a correlation with target variable is not very useful)
- There is a set of highly correlated features: number of rooms, bedrooms, population and households. It can be useful to reduce dimensionality of this subset, especially if we use linear models

- total_bedrooms is one of these highly correlated features, it means we can fill NaN values with high precision using simplest linear regression

6. Machine Learning

6.1 Data Preprocessing and Feature Selection





We can see that INLAND houses have significantly lower prices. Distribution in others differs but not so much. There is no clear trend in house price / proximity, so we will not try to invent a complex encoding approach. Let's just do OHE for this feature.

To Avoid From Dummy Variable Trap we will use pd.get_dummies

What is a Dummy Variable Trap?

The Dummy variable trap is a scenario where there are attributes which are highly correlated (Multicollinear) and one variable predicts the value of others. When we use one hot encoding for handling the categorical data, then one dummy variable (attribute) can be predicted with the help of other dummy variables. Hence, one dummy variable is highly correlated with other dummy variables. Using all dummy variables in models leads to a dummy variable trap. So, the models should be designed excluding one dummy variable.

For Example, Let’s consider the case of gender having two values male (0 or 1) and female (1 or 0). Including both the dummy variables can cause redundancy because if a person is not male in such a case that person is a female, hence, we don’t need to use both the variables in models. This will protect us from dummy variable traps.

6.2 Model Selection

In this section we studied the performance of 3 classification models:

1.Linear Regression

2.Decision Tree Regression

3.Random Forest Regression (Ensemble Learning).

The pros and cons of each model are summarized.

Perform Linear Regression :

1.Perform Linear Regression on training data.

2.Predict output for test dataset using the fitted model.

3.Print root mean squared error (RMSE) from Linear Regression.

Data preparation for Machine Learning algorithms

It's time to prepare the data for Machine Learning algorithms. Instead of doing this manually, we should write functions for this purpose, for several good reasons:

- This will allow reproduction of these transformations easily on any dataset (e.g., the next time get a fresh dataset).
- We can gradually build a library of transformation functions that you can reuse in future projects.

- We can use these functions in your live system to transform the new data before feeding it to ML algorithms.

6.3 Data Cleaning

Most Machine Learning algorithms cannot work with missing features, so let's create a few functions to take care of them. We saw earlier that the total_bedrooms attribute has some missing values, so let's fix this with three options:

1. Get rid of the corresponding districts.
2. Get rid of the whole attribute.
3. Set the values to some value (zero, the mean, the median, etc.).

We can accomplish these easily using DataFrame's dropna(), drop(), and fillna().

There is no obvious reason for some total_bedrooms to be NaN. The number of NaNs is about 1% of the total dataset. Maybe we could just drop these rows or fill it with mean/median values, but let's wait for a while, and deal with blanks after initial data analysis in a smarter manner.

6.4 Baseline model evaluation

Linear regression is fast, simple and can provide quite a good baseline result for our task. Tree based models can provide better results in case of nonlinear complex dependences of variables and in case of small number of variables, they are also more stable to multicollinearity (and we have highly correlated variables). Moreover in our problem target values are clipped and targets can't be outside the clipping interval, it is good for the tree-based models.

The results of using these models will be compared at the end of the project. Tree-based models are expected to work better in this particular problem, but we will start with a more simple model.

We will start with standard linear regression, go through all of the modeling steps, and then do some simplified computations for 2 other models (without in-depth explanation of every step).

The final model selection will be done based on the results

Linear Regression

Since most median housing values range between 120k and 265k, a typical prediction of 68,628 USD is not very satisfying. So, we can conclude that our model is underfitting the data.

This can mean that the features do not provide enough info or the model is not powerful enough. So, we have to select a more powerful model, to feed the algo with better features, or to reduce the constraints on the model. And also, this model is not regularized.

Decision Tree

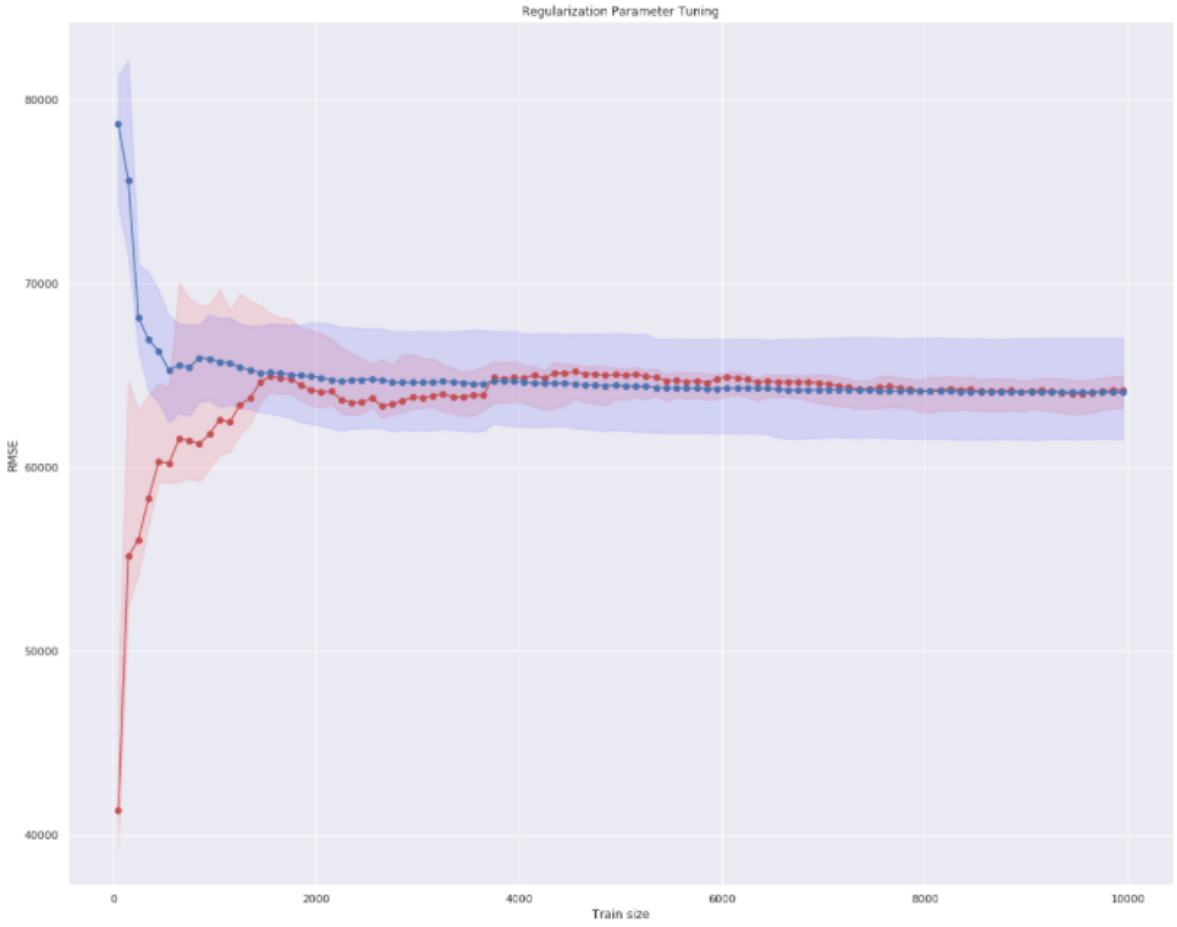
We can conclude that the decision tree is highly overfitting the data to the extent that it has performed worse in comparison with the linear regression model used earlier. So, let's now give another try with the random forest model.

Random Forest

Step1: Firstly, we trained all the models with the baseline implementation, meaning all the hyperparameters of the models were left as the default value in the scikit-learn APIs.

6.4 Model Comparison

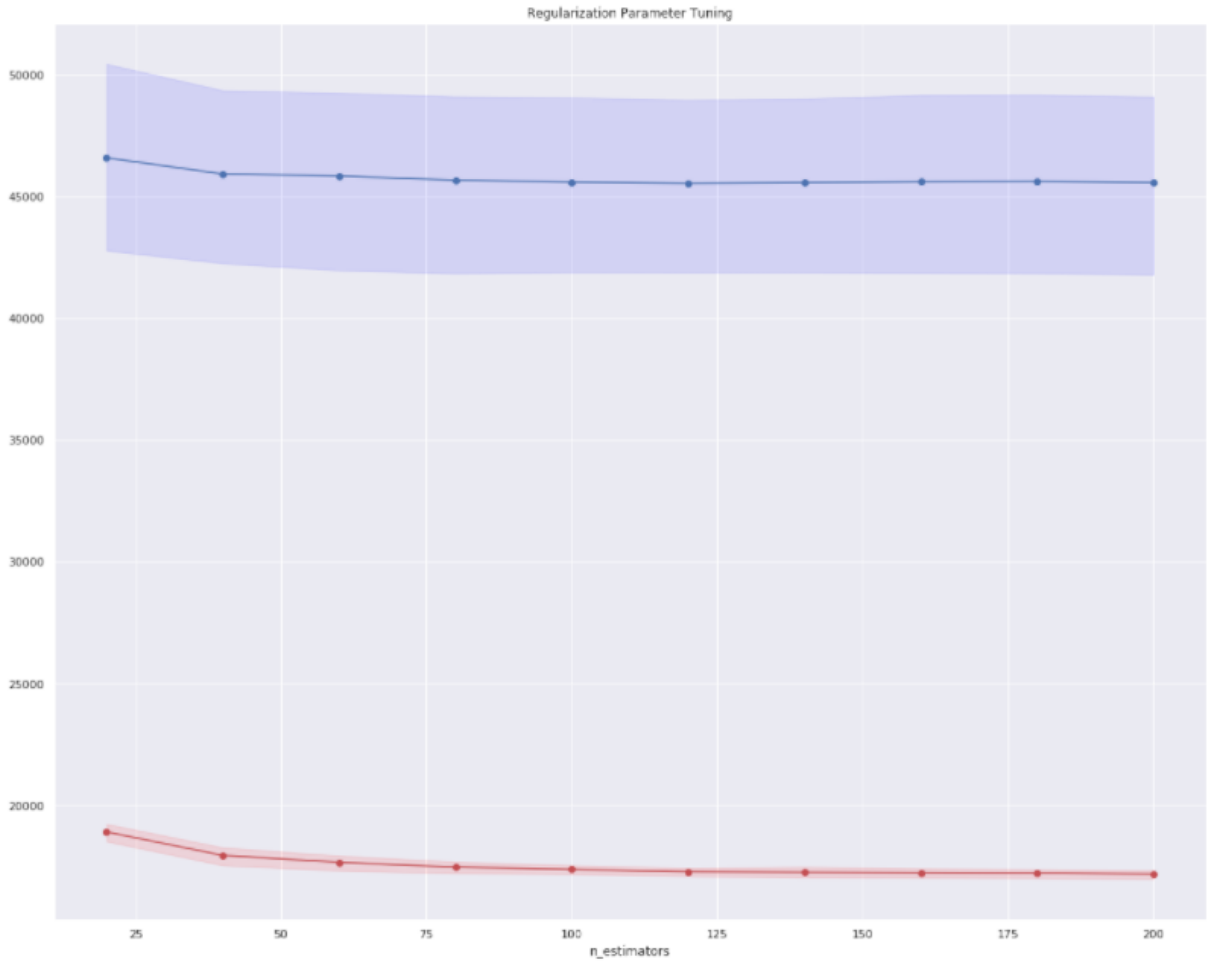
1、Linear Regression



Learning curves indicate high bias of the model - this means we will not improve our model by adding more data, but we can try to use more complex models or add more features to improve the results.

This result is inline with the validation curve results. So let's move on to the more complex models.

2.RandomForestClassifier



This time we can see that the results of the train are much better than LR, but it is totally ok for the Random Forest.

Higher value of n_estimators (>100) does not help much. Let's stick to the n_estimators=200 - it is high enough but not very computationally intensive.

3.Decision Tree Regression

7. Final Model Selection and Hyperparameters Tuning

	Model	CV_results	Holdout_results
1	Linear Regression	0.605	0.627
2	Decision Tree Regression	0.593	0.642
3	Random Forest	0.426	0.815

	Models	**Accurasy score**
1	Linear Regression	0.605 - 0.627
2	Desision Tree Regression	0.593 - 0.642
3	Random Forest Reg	0.426 - 0.815

Cross Validation results are inline with holdout ones. Our best CV model - Random Forest turned out to be the best on hold-out dataset as well.

12. Conclusion

To sum up, we have got the solution that can predict the mean house value in the block with RMSE $46k using our best model - Random Forest Regression. It is not an extremely precise prediction: $46k is about 20% of the average mean house price, but it seems that it is near the possible solution

for these classes of model based on this data (it is a popular dataset but I have not found any solution with significantly better results).

We have used old Californian data from 1990 so it is not useful right now. But the same approach can be used to predict modern house prices (if applied to the recent market data).

We have done a lot but the results surely can be improved, at least one could try:

- feature engineering: polynomial features, better distances to cities (not Euclidean ones, ellipse representation of cities), average values of target for the geographically closest neighbors (requires custom estimator function for correct cross validation)
- PCA for dimensionality reduction (I have mentioned it but didn't used)
- other models (at least KNN and SVM can be tried based on data)
- more time and effort can be spent on RF and LGB parameters tuning