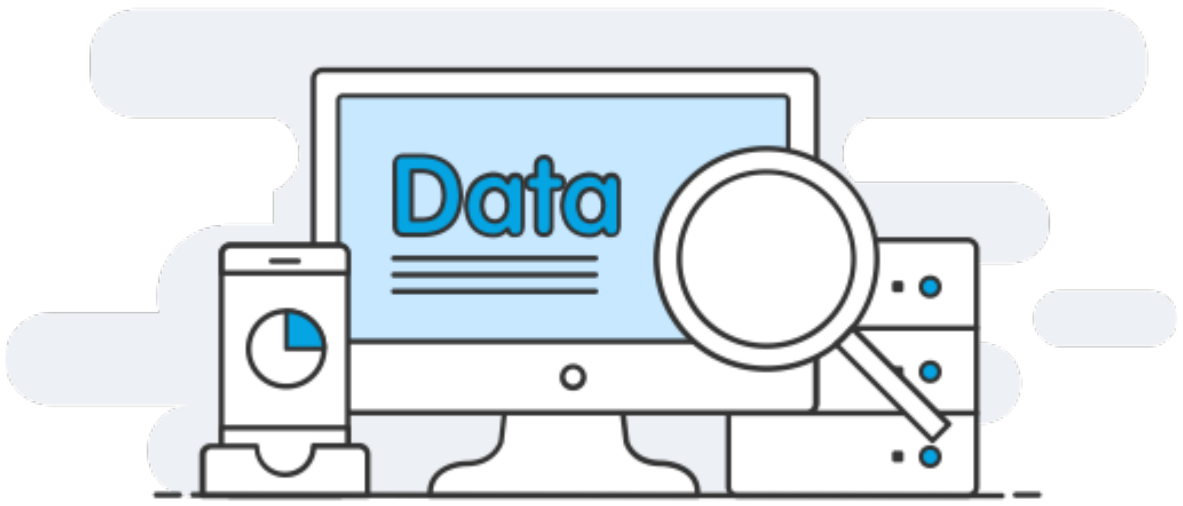
California Housing Prices

Median house prices for California districts



Source: https://www.kaggle.com/camnugent/california-housing-prices#housing.csv

This data set covers a domain for exercising the Finance and Housing data set. The data contains information from the 1990 California census. This data set has 10 types of metrics such as the population, median housing prices, income, location etc. for each block group in the state of California. 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.

The columns are as follows, their names are pretty self explanatory:

1. longitude
2. latitude
3. housing_median_age
4. total_rooms
5. total_bedrooms
6. population
7. households
8. median_income
9. median_house_value
10. ocean_proximity

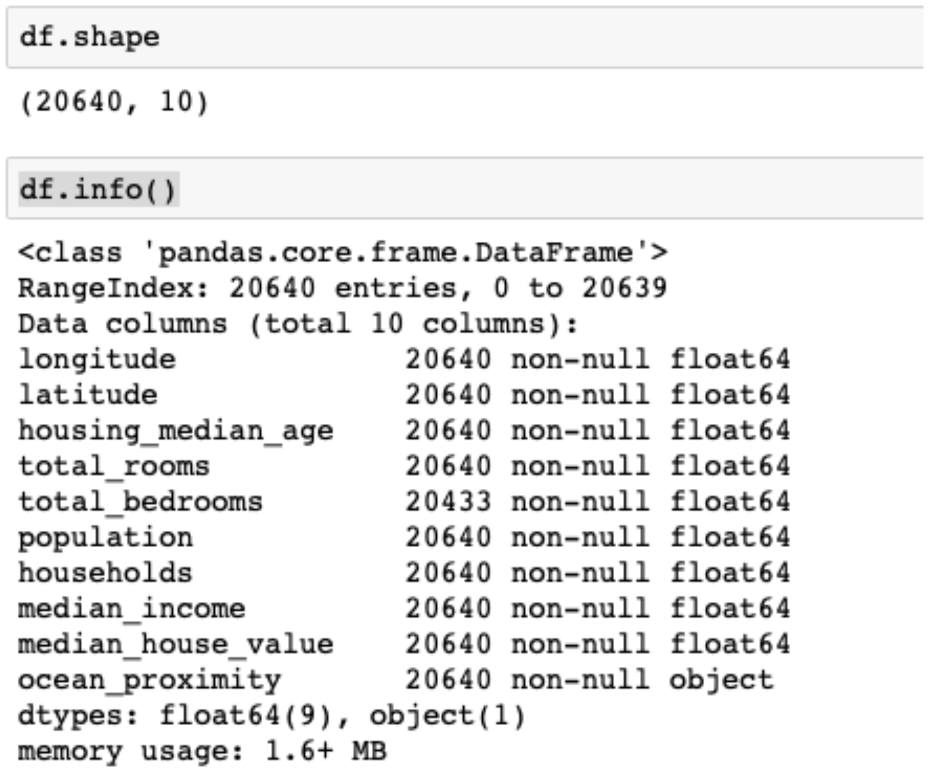
Problem Statement/Goal

To predict the housing price in the State of california.

SIGNIFICANCE OF THE PROBLEM

To buy a house is on everyone' checklist. One of the most important factors in buying a house is the income or the money each household have. Besides this other factors include location, distance from work, the size of the house etc. Buyer consider these factors while looking for a house in the market. Prediction of the housing value is therefore dependent on these factors. Considering these values as the predictor for the housing prices will also help the government, private companies, insurance companies and real estate agent to invest money accordingly.

Exploratory Data Analysis



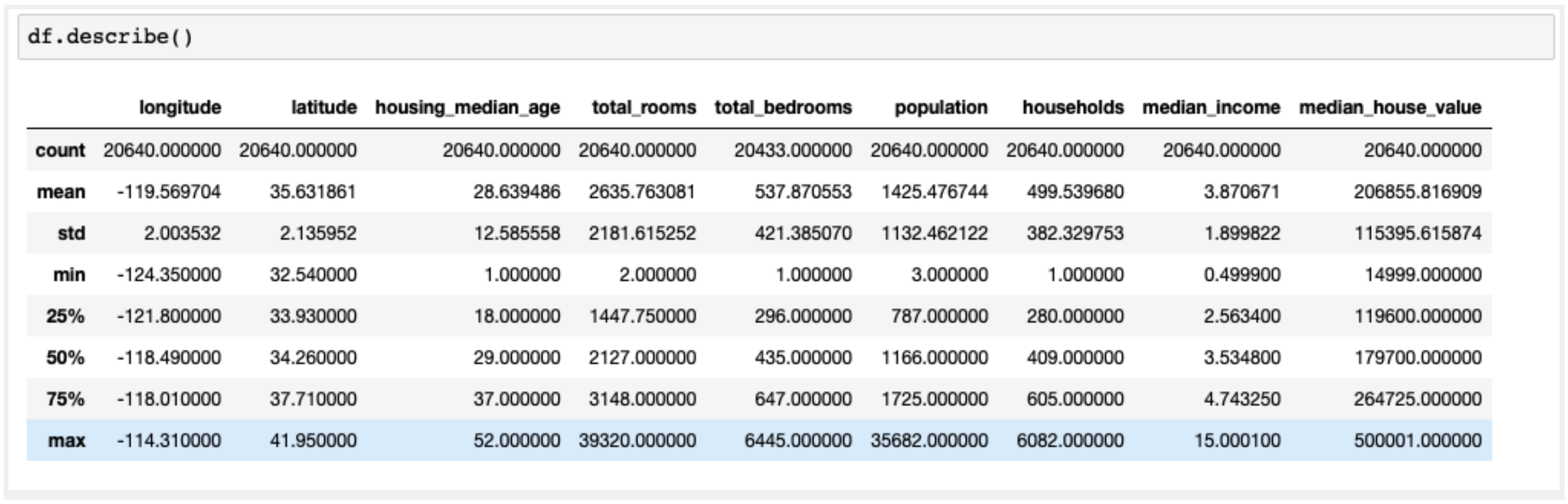
```
df.shape
```

```
(20640, 10)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

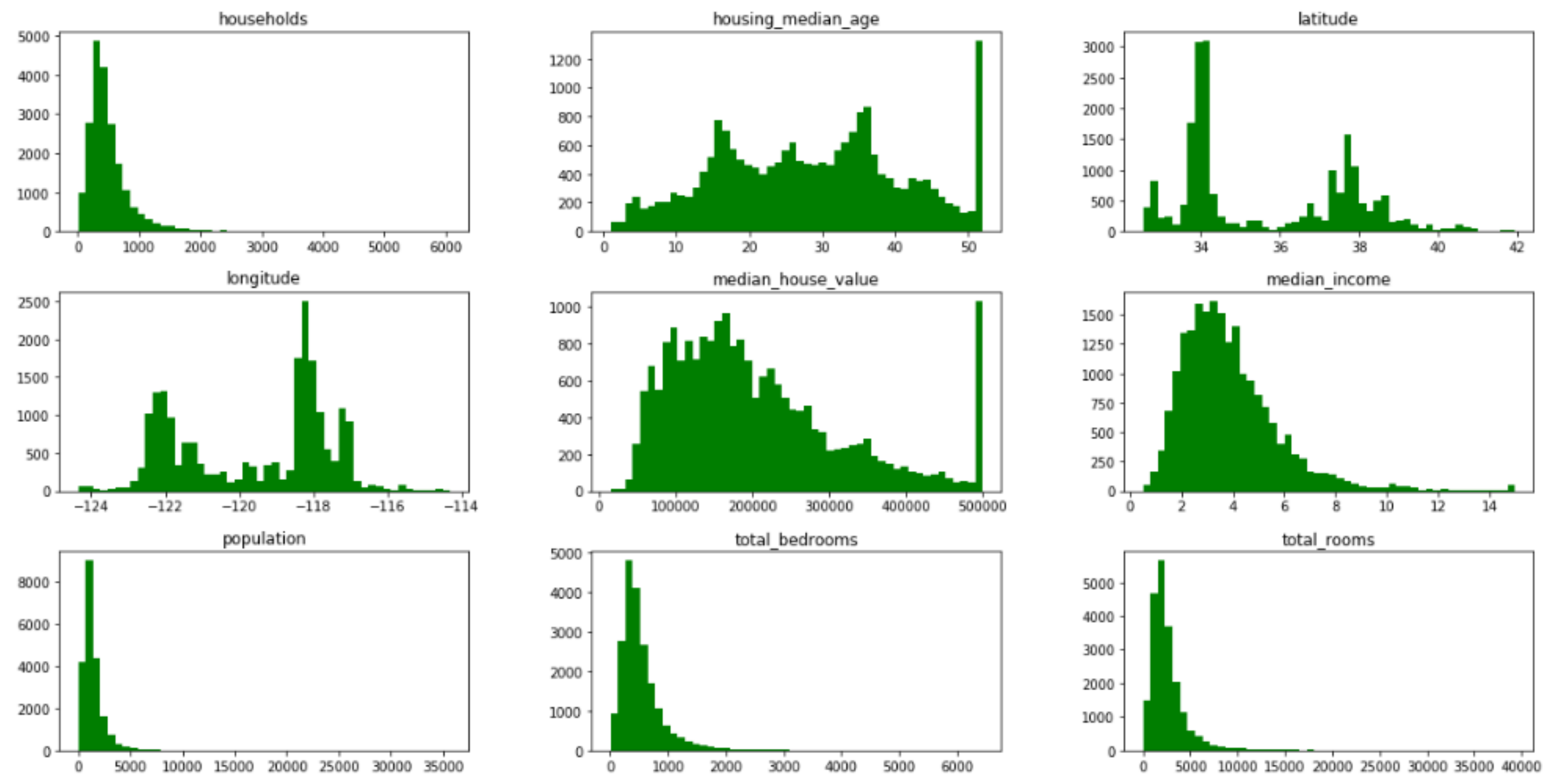
- 10 variables
- 9 Numerical and 1 categorical variable
- Total Entries = 20640
- One variable has missing values (n=207)
- To proceed further with the data, one has to take care of these missing values. This can be fixed either by (i) removing the rows with the missing values or (ii) filling the missing values with the mean value. As the missing values are approx 1% of the total values, we can move further with option one i.e removing the rows with the missing values. Another important point to note the presence of outliers in this variable. If we look at the box plot this variable has many outliers, in this scenario, filling missing values with mean is not appropriate.



```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

- The count, mean, min, and max rows are self-explanatory.
- Standard deviation (std) - dispersion of data points.
- The 25%, 50%, and 75% show the corresponding percentiles
- There is a huge difference in the 75%percantile and max values, probability of having significant outliers in the variables : total_rooms, population, median income, bedroom.



OBSERVATION

- Just like describe , we use histogram or box plot as a visual tool to look at the numerical data
- For example, you can see that slightly over 900 districts have a median_house_value equal to about $500,000

- Median income : No units describes, it seems that it is expressed value*10000. e.g. if 12, that means 120, 000 or 120K.
- **NOTE** - It is not necessary to have the units for the data points, but it is better to have , so that we can correlate logically while making the models.
- Many histograms are tail heavy: they extend much farther to the right of the median than to the left. We might need some transforming later.
- Total rooms and total bedroom, we can also make a new variable like rooms per household , something like that
- population — most districts have population below 3000
- bedrooms — looks like most districts have between 300–600 bedrooms
- total rooms — most districts have around 3000 rooms

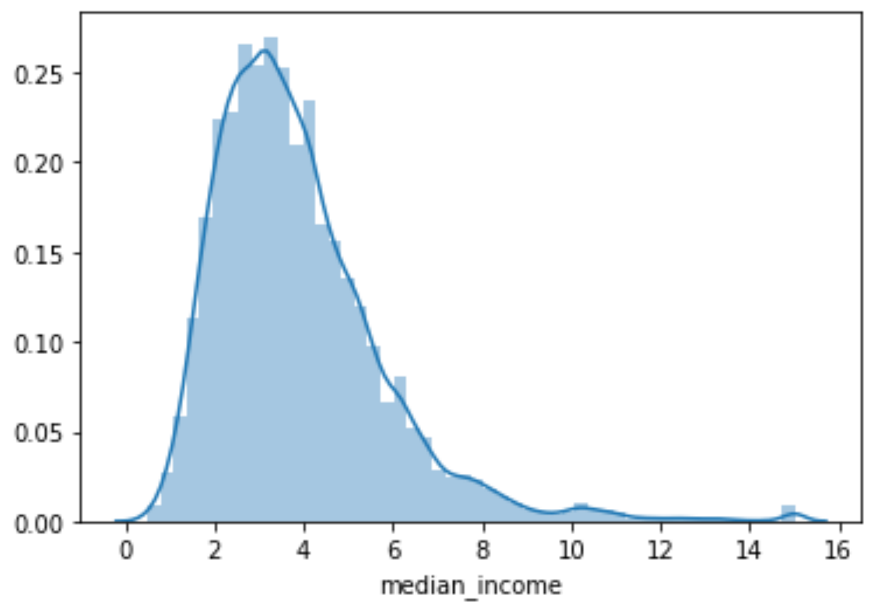
****

What is the first thing that comes to mind when we think of buying anything?

MONEY - SAVINGS - INCOME

****

`Let'S TAKE A CLOSURE LOOK AT THE MEDIAN INCOME`



OBSERVATION

- Right tail shift
- Majority of population have

income between 2-6 (

- 20,000−
- 20,000−60,000)
- Some numbers go beyond 8

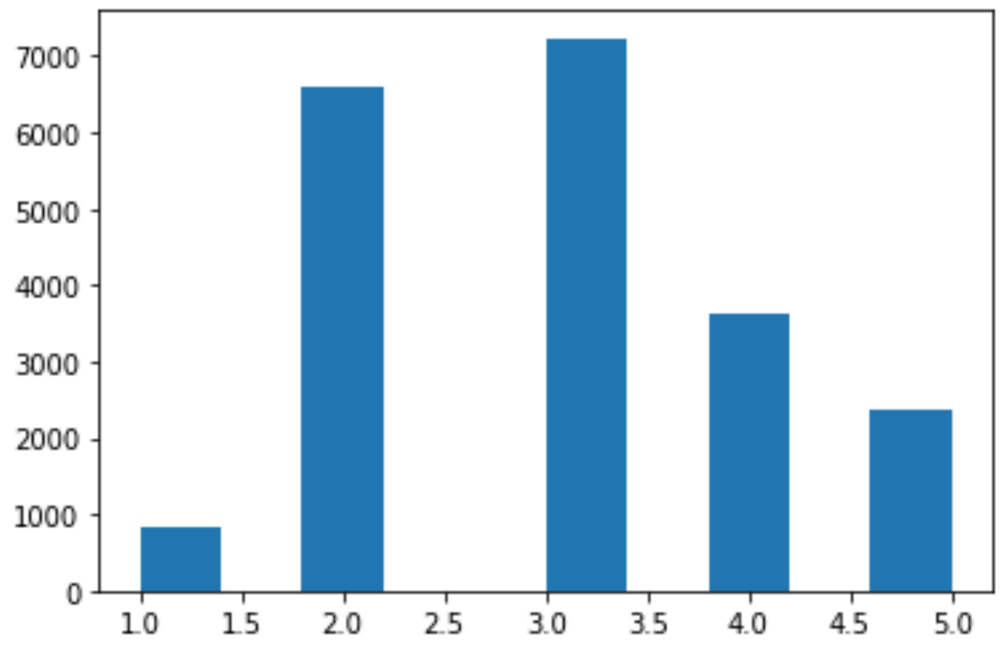
and max at $150,000

`TO DO`

Median income data is continuous. So let's make it discrete

```
df["income_cat"] = pd.cut(df["median_income"],
                          bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                          labels=[1, 2, 3, 4, 5])
```

```
df["income_cat"].hist()
plt.grid(False)
```



****

MACHINE LEARNING

****

- ❖ Before we explore data more, lets split the data into test and train data set

Slicing housing data set into a training set and test set.

- ❖ test_size = 0.2 means we split test and train in a ratio of 20 to 80

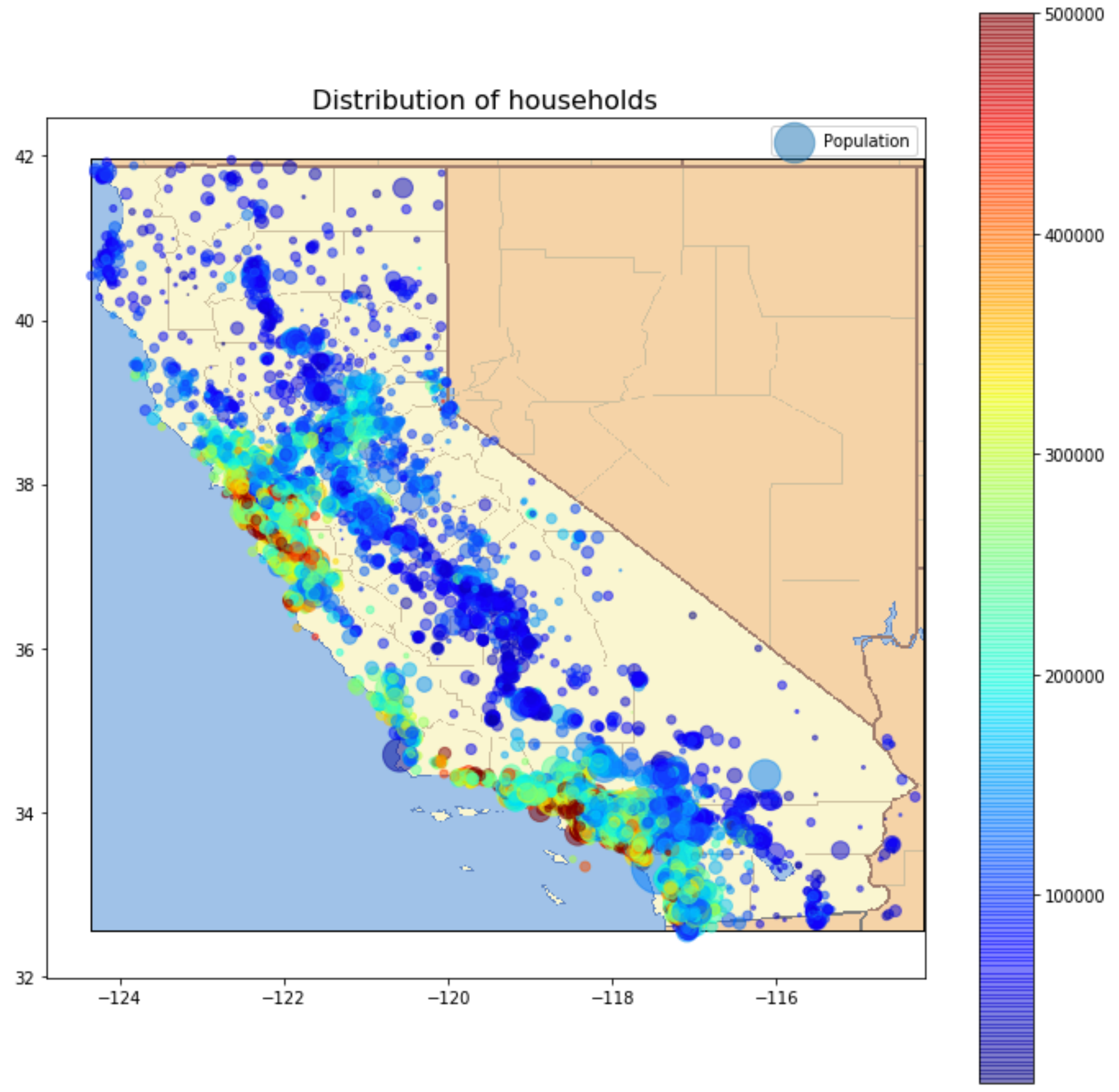
SPLITTING DATA INTO TRAINING AND TESTING SET

The training dataset and testing dataset, usually have the same predictors or variables. They differ on the observations and specific values in the variables. If one fits the model on the training dataset, then the chances are high to minimize error. The fitted model always provides a good prediction on the training dataset. Firstly, test the model made on the test dataset without the dependent variable which is called validation set. One builts few models and test them on validation test. The models which is best with the validation test, will be tested with the `test_data_seti` as that data_set has never seen the model before. This helps to prevent overfitting or underfitting of the data.

Now based on income categories, we'll split our entire data into train (0.8) and test (0.2)
We are using stratified sampling technique here since we derived a new attribute called income category.

****

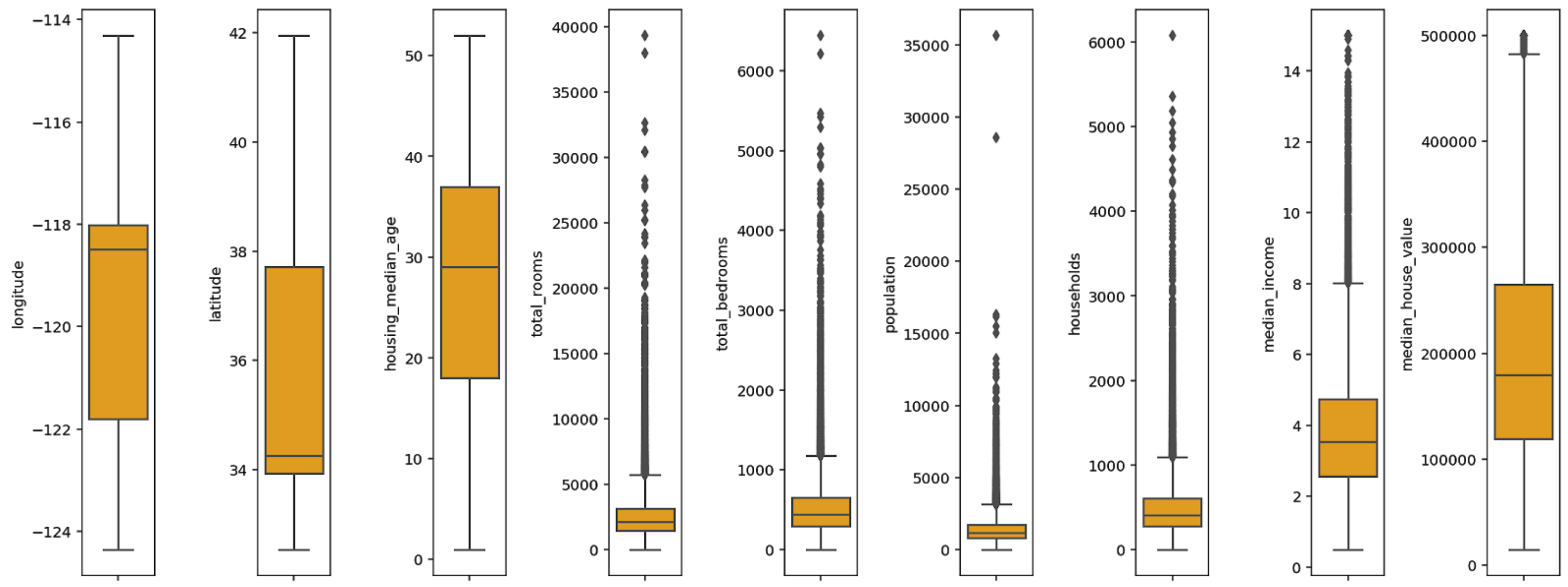
Scatter Plot: Data - Train Set



Inference: The scatter plot between longitude and latitude was plotted with the population density. Overlapping of scatter plot with the State of California map (as shown above) clearly demonstrated that the housing price near the bay area are expensive compared to the rest of the part.

This scatter plot gave us a rough idea about the distribution of the houses and their prices.

****
Distribution of data points in Variables - Outliers | Transformation | Normality

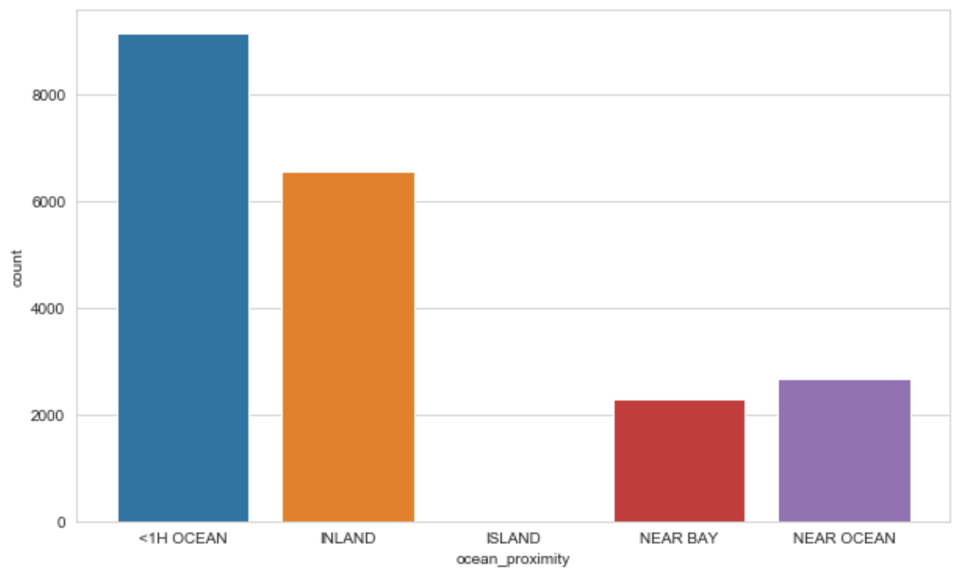


Inference

- Box Plots are the best way to get an idea about the outliers.
- 5 variables have outliers
- Transformation or making a new variable can be helpful.

****
Categorical Variable

There is only one categorical variable. Counts are plotted below.

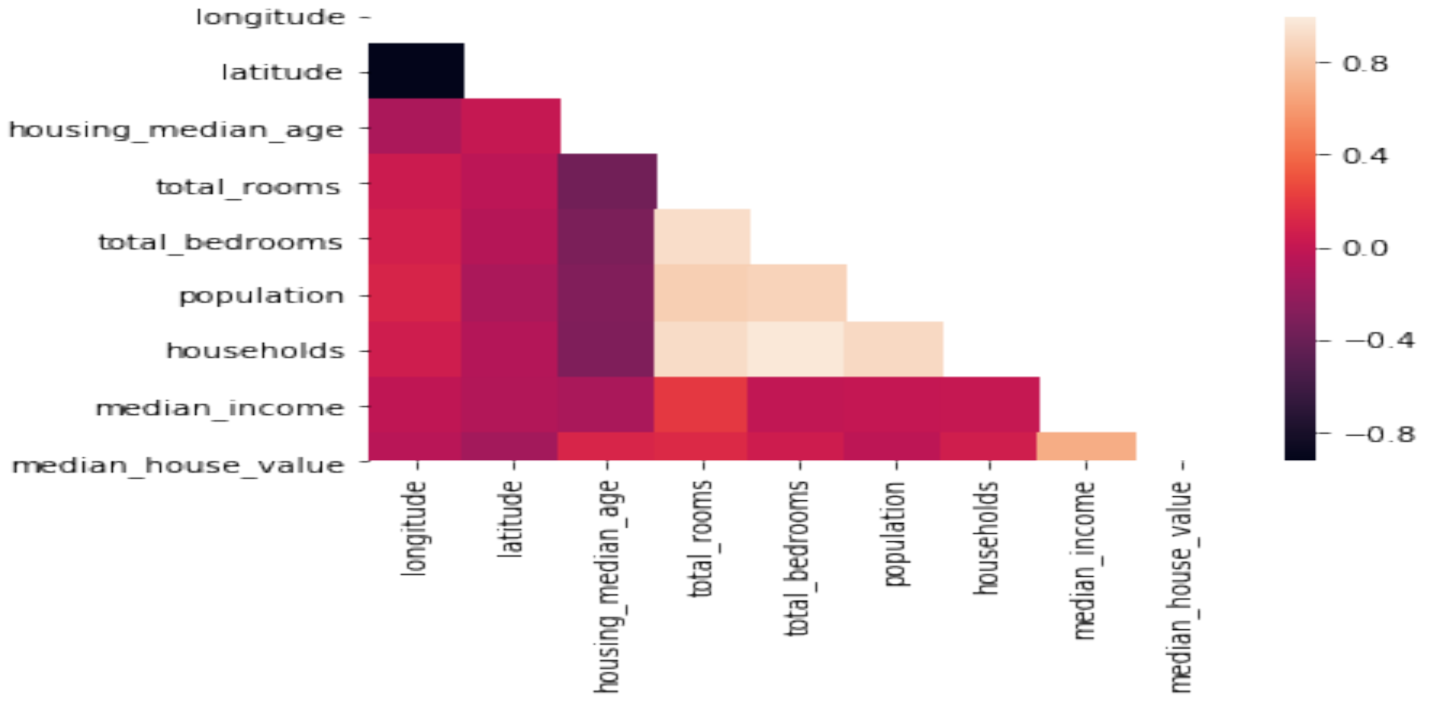


<1H Ocean = 9136
Inland = 6551
Near Ocean = 2658
Near bay = 2290
Island = 5

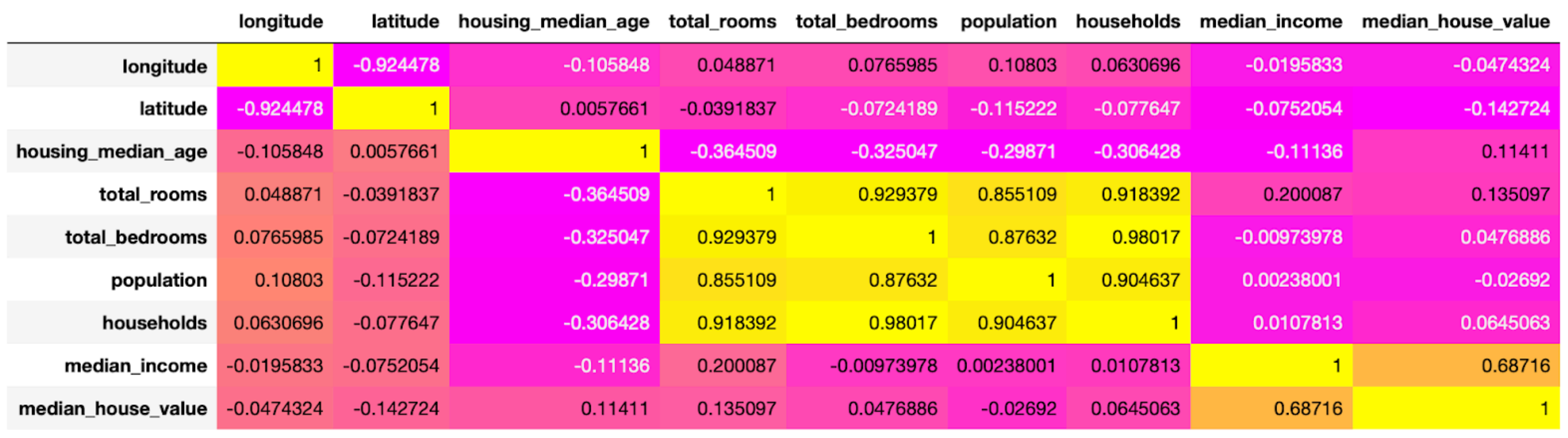
Note: Did not explore this variable in this ML project, however one can explore this categorical variable, to explore the house prices in different locations and it varies. Island , have only five data points, one can delete these or can merge then with “Near Ocean’.

****

Correlation Matrix :



Alternative



	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
longitude	1	-0.924478	-0.105848	0.048871	0.0765985	0.10803	0.0630696	-0.0195833	-0.0474324
latitude	-0.924478	1	0.0057661	-0.0391837	-0.0724189	-0.115222	-0.077647	-0.0752054	-0.142724
housing_median_age	-0.105848	0.0057661	1	-0.364509	-0.325047	-0.29871	-0.306428	-0.11136	0.11411
total_rooms	0.048871	-0.0391837	-0.364509	1	0.929379	0.855109	0.918392	0.200087	0.135097
total_bedrooms	0.0765985	-0.0724189	-0.325047	0.929379	1	0.87632	0.98017	-0.00973978	0.0476886
population	0.10803	-0.115222	-0.29871	0.855109	0.87632	1	0.904637	0.00238001	-0.02692
households	0.0630696	-0.077647	-0.306428	0.918392	0.98017	0.904637	1	0.0107813	0.0645063
median_income	-0.0195833	-0.0752054	-0.11136	0.200087	-0.00973978	0.00238001	0.0107813	1	0.68716
median_house_value	-0.0474324	-0.142724	0.11411	0.135097	0.0476886	-0.02692	0.0645063	0.68716	1

Highly correlated features:

- (+) Total bedroom, population, household
- (-) latitude and longitude (Which is kind off obvious)

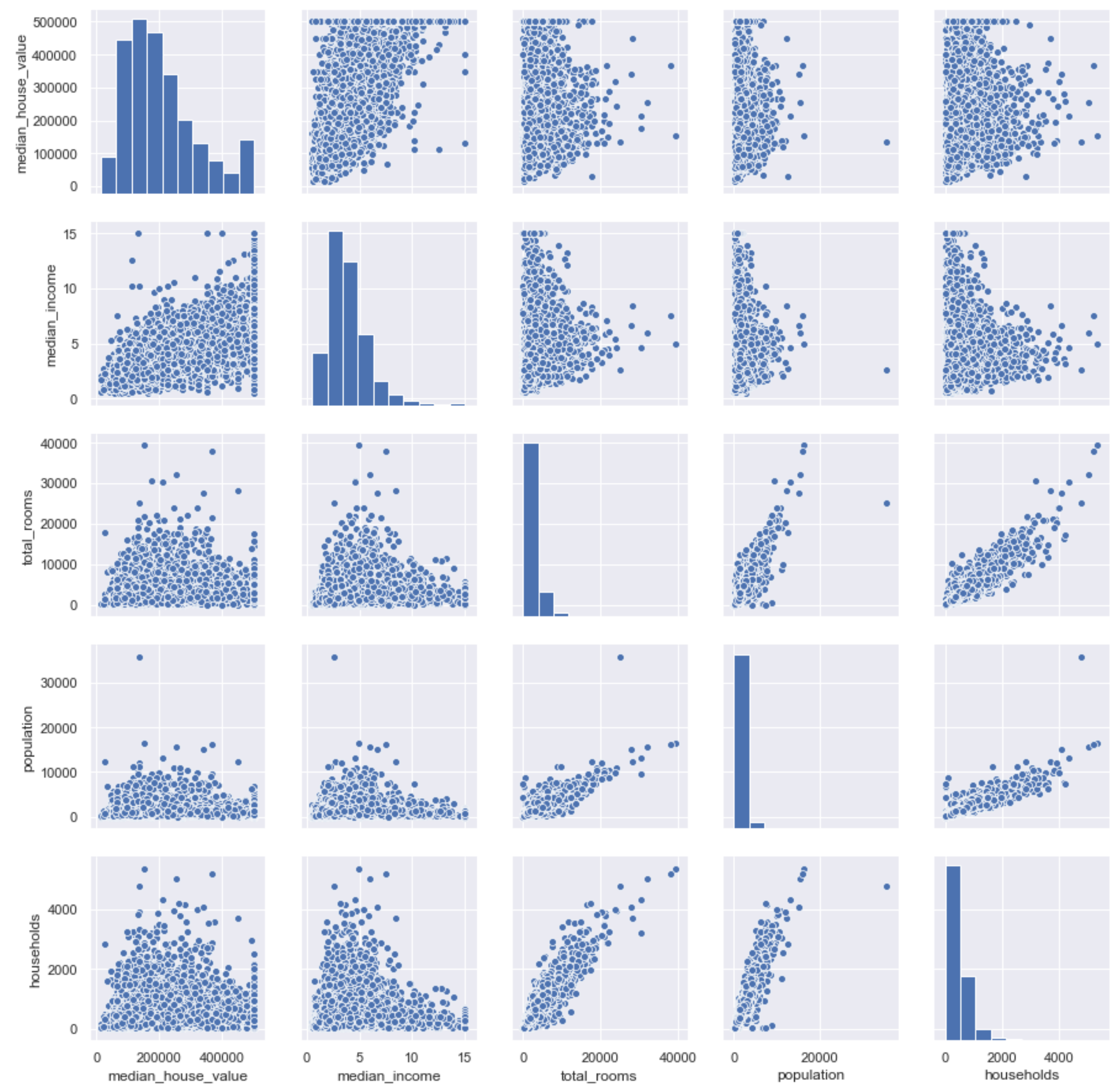
Summary:

- Median income and house value are positively correlated. (Median Income is the most promising attribute to get Median Housing Price)
- Latitude and longitude are negatively correlated
- Total rooms and total bedrooms in a house look very similar, so we can drop either of them

****

PairPlot

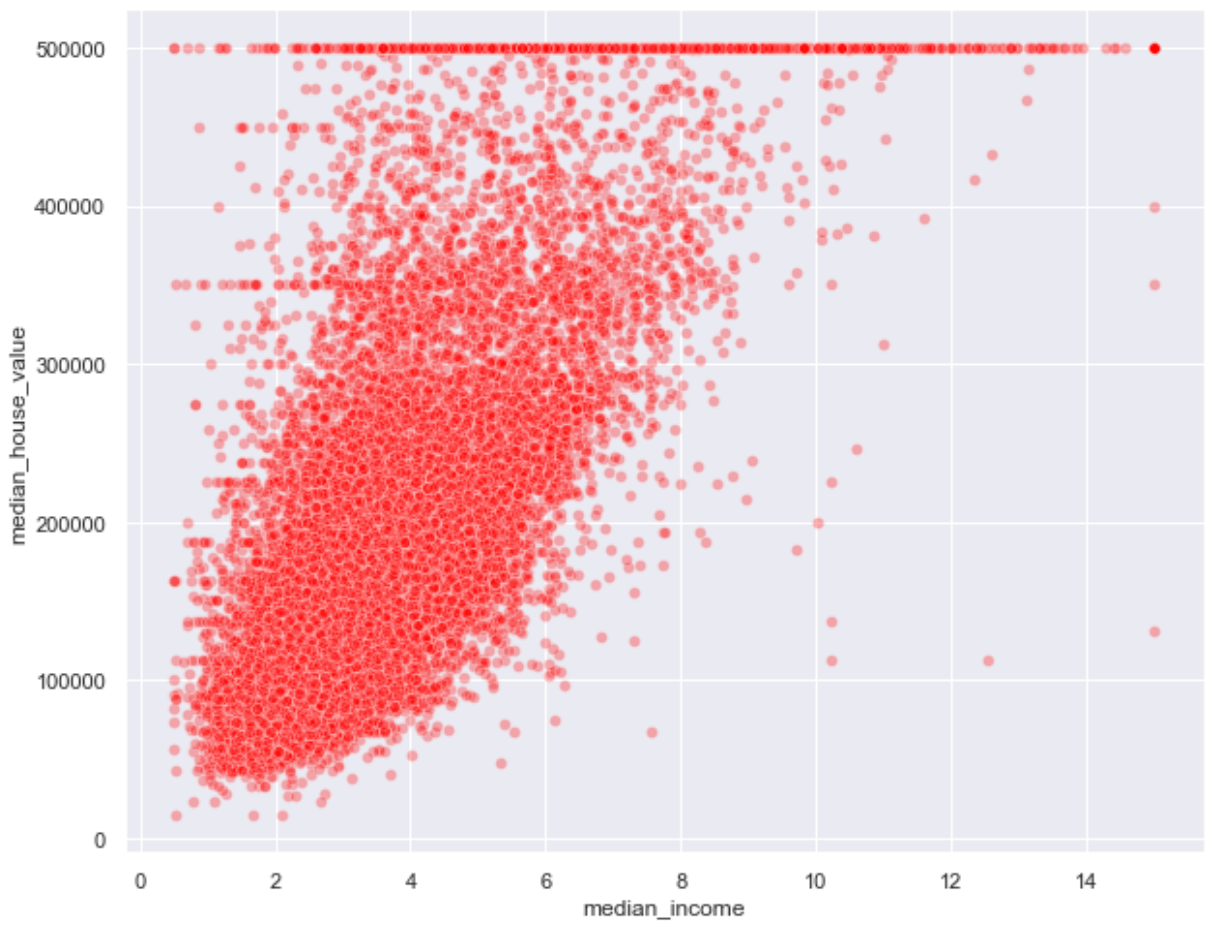
As we see in the correlation matrix plot, only five variables seem to correlated. So next, pairplot was plotted. The most associated attribute to predict the median house value is the median income.



****

Considering median house value and the median income are well associated, explored this association a bit more.

Below is the scatter plot between median house values vs house income.



INFERENCE

- Correlation is strong (0.686117) as per the table above
- Clear Horizontal lines at 500,000, 450,000, 350,000, perhaps one around 280,000, and a few more below that.
- This kind of data may degrade the performance of model.

****

Note : As mentioned before that, total rooms and total bedrooms, are quite similar and do not add much. We can remove one variable and, then we can also make a new variable like rooms per household or total bedrooms per household.

****

`NEW ATTRIBUTES`

```
housing['rooms_per_household'] = housing['total_rooms']/housing['households']
housing['bedrooms_per_room'] = housing['total_bedrooms']/housing['total_rooms']
housing['population_per_household'] = housing['population']/housing['households']
housing.head()
```

Correlation with new variables

Observations : Bedrooms variable is better correlated compared to total bedrooms

```
median_house_value          1.000000
median_income               0.687160
rooms_per_household         0.146285
total_rooms                 0.135097
housing_median_age          0.114110
households                  0.064506
total_bedrooms              0.047689
population_per_household   -0.021985
population                 -0.026920
longitude                  -0.047432
latitude                   -0.142724
bedrooms_per_room          -0.259984
Name: median_house_value, dtype: float64
```

****

REMEMBER - WHAT IS OUR AIM "TO PREDICT THE HOUSING PRICE"__

****Independent variable (Predictor) and Dependent variable (Label/Target)***

__IN THIS DATA SET OUR DEPENDENT VARIABLE IS MEDIAN_HOUSE_VALUE__

Before moving to Machine Learning Modelling, let's split the data into test and drop the dependent variable (Validation Data_Set).

****

CLEANING THE DATA

__ After EDA and splitting the data , now I will clean the data.

**** Remember : Total_bedrooms had some missing values**

****How to fix the Missing Values****

- Remove the rows of missing values "(housing.dropna(subset=["total_bedrooms"])"
- Remove the entire attribute "housing.drop("total_bedrooms", axis=1)"
- Set missing value to some value like zero, median or mean

```
*****Scikit-Learn provides a function to take care of missing values:
`SimpleImputer`******
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median") ## We are filling NA with MEDIAN
housing_num = housing.drop('ocean_proximity', axis=1) #Median can only be calculated
for the numerical variables, so drop Ocean_proximity
housing_num.describe()
```

****
NOTE: After taking care of missing values, one can use this data for making different models. Else, one can remove outliers or transform the variables. Here in this project, I will be using data without any transformation and removing the outliers.

****

PREDICTIVE MODELING
There are different machine learning algorithms to predict housing prices. In this project I used three regression algorithms to train machine learning models : Linear Regression, Random Forest Regression, and Decision tree Regressor. There are many factors that affect house prices, such as number of bedrooms, location, income, etc. In addition, one should keep this in mind that choosing different combinations of parameters can also affect the predictions.

Linear Regression
Linear regression is a statistical approach for modeling the relationship between a dependent variable with a given set of independent variables.

Random Forest Regressor
The random forest model is good with numerical and categorical features. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target.

Decision Tree Regressor

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

****
****

Prediction Errors with three different models :

- Linear Model = 69052 ± 2731
- Decision Tree Regressor = 70770 ± 2342.
- Random Forest Regressor = 52836 ± 2081

****
****

<u>OBSERVATION</u>
Looking at the values above it is clear that the Random Forest Regressor method is the best fit among three.

****

<u>FINE TUNING</u>
I tried to do the finetuning of the Model using GridSearch . This method tells our model about the hyperparameters. Also, Grid search is good, when we have a small number of hyperparameters and samples.

OUTPUT

- *Mean Score* _49934.70__
- *Compare this score vs the one before fine tuning* __49934 vs 52836 __FINE TUNING IMPROVES THE ERROR

****

<u>CONCLUSIONS</u>

<u>TIME TO RUN TEST_DATA_SET</u>

We have reached the point where we tested three models and fine tuned the best. Now this is time to test this model with the TEST_DATA_SET

After running the test data set the values we got is 47707.902082224886, which demonstrates that our Model after fine tuning is performing well.

HOWEVER

There are few other ways we can explore the data set.

- **Instead of filling median in place of missing values, one can delete the rows with the missing values**
- **Remove outliers and run the similar analysis**
- **Make new attributes, just like we did for rooms per household**