METRO COLLEGE
OF TECHNOLOGY

Project for Developing Machine Learning Model using Multiple Liner Regression in Python Programming to predict House Prices

---

Submitted by:

Mohammad Monjur-E-Elahi

Course: Data Mining- Advanced Statistical Modeling [DSA09]

Program: Data Science and Application - Advanced Diploma [6060]

Metro College of Technology

Date: 30 April, 2021

Introduction:

California Housing Prices.Median house prices for California districts derived from the 1990 census. Collected from Kaggle.org

This is the dataset serves as an excellent introduction to implementing machine learning algorithms because it requires rudimentary data cleaning, has an easily understandable list of variables and sits at an optimal size between being too toyish and too cumbersome.

The data contains information from the 1990 California census. So although it may not help you with predicting current housing prices like the Zillow Zestimate dataset, it does provide an accessible introductory dataset for learning about the basics of machine learning.

Context 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 data aren't cleaned so there are some preprocessing steps required! The columns are as follows, their names are pretty self explanitory:

longitude

latitude

housingmedianage

total_rooms

total_bedrooms

population

households

median_income

medianhousevalue

ocean_proximity

Detail of the featues:

```
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

Let us start by importing the most relevant libraries.

In [1]:
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

We should check the Current Working Directory at the beginning so that we do not have to provide full path while importing the dataset.

In [2]:
```
os.getcwd()
```

Out[2]:
```
'C:\\Users\\ruzdomain\\Desktop\\DMASM\\PROJECT\\LinearRegression'
```

Check the dataset in the current working direction.

In [3]:
```
!dir |  findstr -i housing.csv
```

```
09/22/2019  01:36 AM         1,423,529 housing.csv
```

Le us now import the dataset to python.

In [4]:
```
housing = pd.read_csv("housing.csv", na_values  = 'NA')
```

Let us investigate the head and tail of the dataset.

In [5]:
```
housing.head(10)
```

Out[5]:

	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
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	NEAR BAY
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0	NEAR BAY
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BAY
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0	NEAR BAY

In [6]:
```
housing.tail(10)
```

Out[6]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
20630	-121.32	39.29	11.0	2640.0	505.0	1257.0	445.0	3.5673	112000.0	INLAND
20631	-121.40	39.33	15.0	2655.0	493.0	1200.0	432.0	3.5179	107200.0	INLAND
20632	-121.45	39.26	15.0	2319.0	416.0	1047.0	385.0	3.1250	115600.0	INLAND
20633	-121.53	39.19	27.0	2080.0	412.0	1082.0	382.0	2.5495	98300.0	INLAND
20634	-121.56	39.27	28.0	2332.0	395.0	1041.0	344.0	3.7125	116800.0	INLAND
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND

Let us look into the detail of the dataset.

In [7]:
```
housing.info()
```

```
<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
```

In [8]:
```
housing.shape
```

Out[8]:
```
(20640, 10)
```

In [9]:
```
housing.columns
```

Out[9]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [10]:
```
housing.dtypes
```

Out[10]:
```
longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
ocean_proximity        object
dtype: object
```

There are 20640 instances and 'total_bedrooms' has only 20433 non-null values.So, some 207 values are missing in that column

We have only one categorical variable. Let us see how many categories are there.

In [11]:
```
housing['ocean_proximity'].value_counts()
```

Out[11]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Let us check all the numeric columns detail.

In [12]:
```
housing.describe()
```

Out[12]:

	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

In [13]:
```
housing.describe(include='all')
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	<1H OCEAN
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9136
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909	NaN
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874	NaN
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000	NaN
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000	NaN
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000	NaN
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000	NaN
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000	NaN

Let us analyze the missing values in the dataset.

In [14]:
```
housing.apply(lambda x: sum(x.isnull()))
```

Out[14]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      207
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

```
Let us see the missing values in percentage.
```

In [15]:
```
def percentage_of_miss(df):
  df1=df[df.columns[df.isnull().sum()>=1]]
  total_miss = df1.isnull().sum().sort_values(ascending=False)
  percent_miss = (df1.isnull().sum()/df1.isnull().count()).sort_values(ascending=False)
  missing_data = pd.concat([total_miss, percent_miss], axis=1, keys=['Number of Missing', 'Percentage'])
  return(missing_data)
```

In [16]:
```
percentage_of_miss(housing)
```

Out[16]:

	Number of Missing	Percentage
total_bedrooms	207	0.010029

Let us check duplicate rows(if any) in the dataset.

In [17]:
```
duplicateRowsHousing = housing[housing.duplicated()]
duplicateRowsHousing
```

Out[17]:

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity

```
So, there is no duplicate rwos in the dataset.
```

Let us replace the missing values of "total_bedrooms" column with mean of the values in that column.

In [18]:
```
housing.fillna(
            {
                'total_bedrooms': housing.total_bedrooms.mean(),

            },
            inplace = True
            )

housing.head(10)
```

Out[18]:

	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
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	NEAR BAY
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0	NEAR BAY
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BAY
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0	NEAR BAY

In [19]:
```
percentage_of_miss(housing)
```

Out[19]:

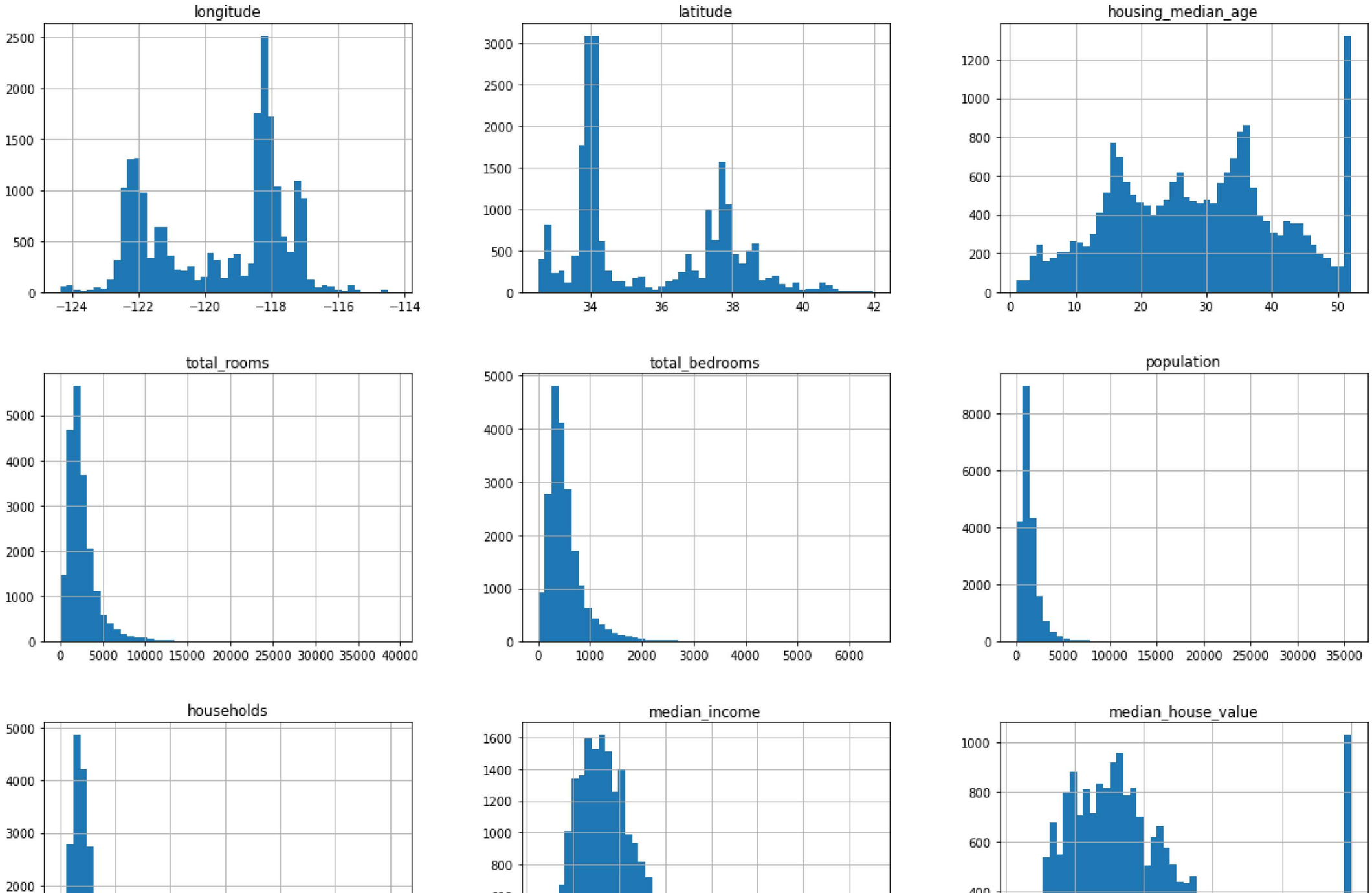
	Number of Missing	Percentage

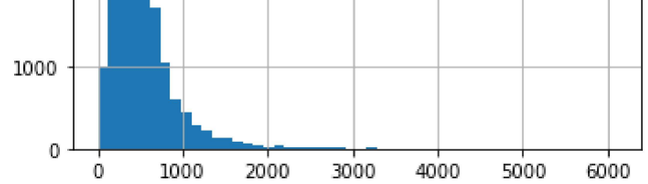
In [20]:
```
housing.info()
```

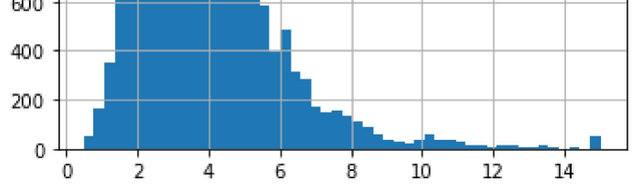
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20640 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
```

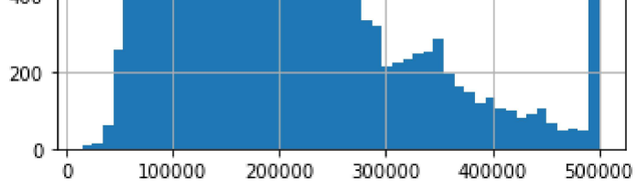
Let us go for some visualization for the dataset

In [21]:
```
housing.hist(bins=50, figsize=(20,15))
plt.show()
```









In [22]:
```
# From the above visualization we could have an idea about the distribution. Now, let us check correlation among all the numerical features
correlation_matrix = housing.corr()
correlation_matrix
```

Out[22]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069260	0.099773	0.055310	-0.015176	-0.045967
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066658	-0.108785	-0.071035	-0.079809	-0.144160
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.318998	-0.296244	-0.302916	-0.119034	0.105623
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.927253	0.857126	0.918484	0.198050	0.134153
total_bedrooms	0.069260	-0.066658	-0.318998	0.927253	1.000000	0.873910	0.974725	-0.007682	0.049454
population	0.099773	-0.108785	-0.296244	0.857126	0.873910	1.000000	0.907222	0.004834	-0.024650
households	0.055310	-0.071035	-0.302916	0.918484	0.974725	0.907222	1.000000	0.013033	0.065843
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007682	0.004834	0.013033	1.000000	0.688075
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049454	-0.024650	0.065843	0.688075	1.000000

We found that there are quite a few multi-colinearity among the independent features. Now, Let us see the correlation of numeric variables with out target or dependent variables which is "median_house_value".

In [23]:
```
correlation_matrix["median_house_value"].sort_values(ascending=False)
```

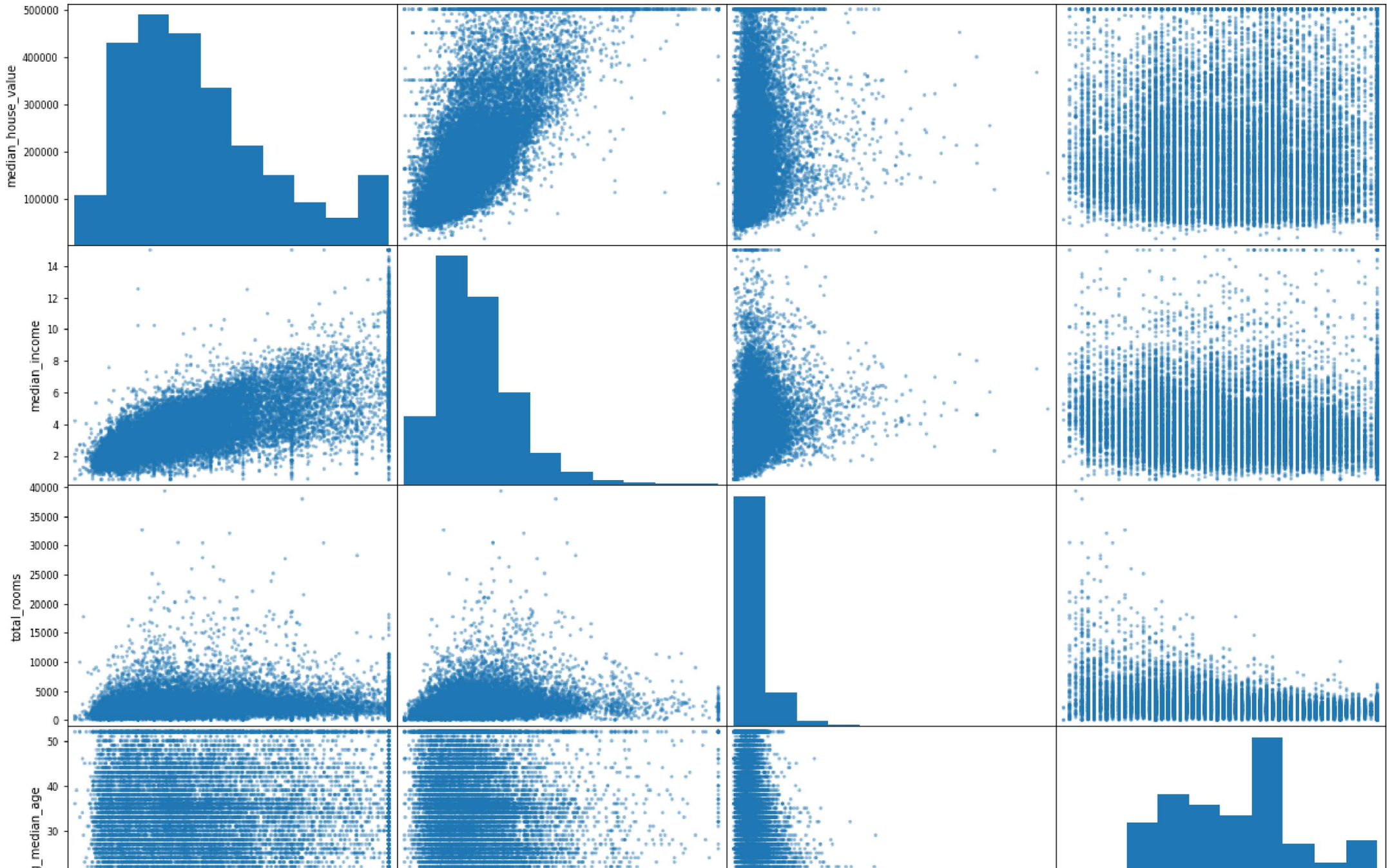
Out[23]:
```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049454
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

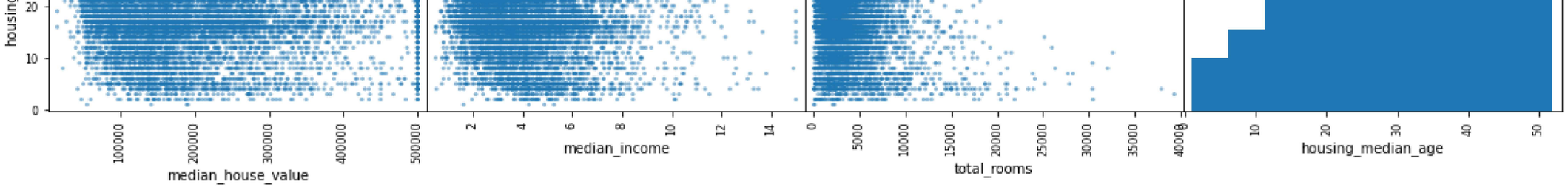
We find that median_income is the most prominant feature. Few other features have some positive correlation with the target column. Let us draw scatter plot using pandas library for

the variables that are showing positive substantial correlation.

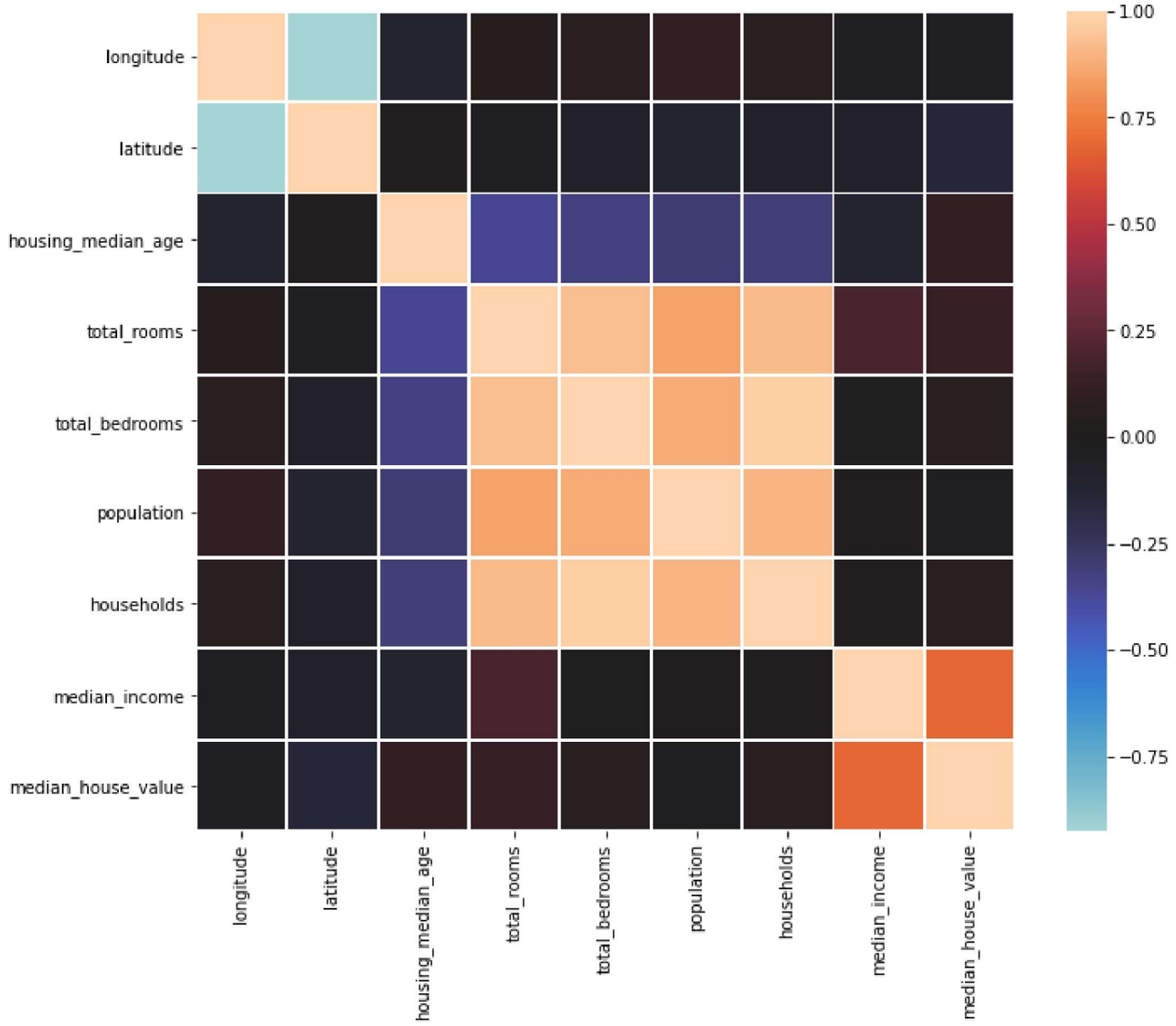
In [24]:
```
# We take the important fetures based on correlation with the target column
important_attributes = ["median_house_value", "median_income", "total_rooms", "housing_median_age"]
```

In [25]:
```
from pandas.plotting import scatter_matrix
important_scatter = scatter_matrix(housing[important_attributes],figsize=(20,15))
```




housing_median_age
0
10
20
100000
200000
300000
400000
500000
median_house_value
2
4
6
8
10
12
14
median_income
0
5000
10000
15000
20000
25000
30000
35000
40000
total_rooms
10
20
30
40
50
housing_median_age

In [26]:
```
# Strong linear correlation is shown between the target median_house_value and median_income
# Let us have a look at the heatmap as well
import seaborn as sns
plt.subplots(figsize=(11, 9))
hmap = sns.heatmap(housing.corr(), center=0, square=True, linewidths=.8)
```



In [27]:
```
# Same strong linear correlation is shown between the target median_house_value and median_income from the heat map as well
housing.head(10)
```

Out[27]:

	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
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	NEAR BAY
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0	NEAR BAY
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BAY
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0	NEAR BAY

Let us convert the categorical variable of "ocean_proximity" with dummy variables.

In [28]:
```
housing_dummies = pd.get_dummies(housing, ['ocean_proximity'], drop_first=True)
housing_dummies.head()
```

Out[28]:

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

Let us extract dependent and independent variables.

In [29]:
```
X = housing_dummies.drop(['median_house_value'],axis=1)
y = housing_dummies.median_house_value
```

In [30]:
```
X.head(10)
```

Out[30]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity_INLAND	ocean_proximity_ISLAND	ocean_proximity_NEAR BAY	ocean
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	0	0	1	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	0	0	1	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	0	0	1	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	0	0	1	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	0	0	1	
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	0	0	1	
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	0	0	1	
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	0	0	1	
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	0	0	1	
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	0	0	1	

In [31]:
```
y.head(10)
```

Out[31]:
```
0    452600.0
1    358500.0
2    352100.0
3    341300.0
4    342200.0
5    269700.0
6    299200.0
7    241400.0
8    226700.0
9    261100.0
Name: median_house_value, dtype: float64
```

Let us import OLS statsmodel to check the p-values of the X variable.

In [32]:
```
import statsmodels.api as sm
X2 = sm.add_constant(X)
ols = sm.OLS(y,X2)
lr = ols.fit()
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.645
Model:                            OLS   Adj. R-squared:                  0.645
Method:                 Least Squares   F-statistic:                     3130.
Date:                Tue, 04 May 2021   Prob (F-statistic):               0.00
Time:                        16:46:30   Log-Likelihood:            -2.5917e+05
No. Observations:               20640   AIC:                         5.184e+05
Df Residuals:                   20627   BIC:                         5.185e+05
Df Model:                          12
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                        -2.236e+06   8.75e+04    -25.552      0.000   -2.41e+06   -2.06e+06
longitude                    -2.646e+04   1014.035    -26.092      0.000   -2.84e+04   -2.45e+04
latitude                      -2.52e+04    999.908    -25.200      0.000   -2.72e+04   -2.32e+04
housing_median_age           1057.8638     43.704     24.205      0.000     972.200    1143.528
total_rooms                    -4.7725      0.772     -6.186      0.000      -6.285      -3.260
total_bedrooms                 72.2896      5.984     12.080      0.000      60.560      84.019
population                    -39.2618      1.064    -36.896      0.000     -41.348     -37.176
households                     76.9409      6.696     11.490      0.000      63.816      90.066
median_income                3.877e+04    332.447    116.635      0.000    3.81e+04    3.94e+04
ocean_proximity_INLAND      -3.972e+04   1736.412    -22.875      0.000   -4.31e+04   -3.63e+04
ocean_proximity_ISLAND        1.56e+05   3.08e+04      5.071      0.000    9.57e+04    2.16e+05
ocean_proximity_NEAR BAY    -3689.7507   1905.960     -1.936      0.053   -7425.582      46.081
ocean_proximity_NEAR OCEAN   4747.4021   1562.688      3.038      0.002    1684.410    7810.395
==============================================================================
Omnibus:                     5175.381   Durbin-Watson:                   0.968
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19749.153
Skew:                           1.212   Prob(JB):                         0.00
Kurtosis:                       7.133   Cond. No.                     7.22e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.22e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

backward feature elimination:

In [33]:
```
while (lr.pvalues.max()>0.05):
    X2.drop(lr.pvalues.idxmax(),axis=1,inplace=True)
    ols = sm.OLS(y,X2)
    lr = ols.fit()
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.645
Model:                            OLS   Adj. R-squared:                  0.645
Method:                 Least Squares   F-statistic:                     3413.
Date:                Tue, 04 May 2021   Prob (F-statistic):               0.00
Time:                        16:46:41   Log-Likelihood:            -2.5917e+05
No. Observations:               20640   AIC:                         5.184e+05
Df Residuals:                   20628   BIC:                         5.185e+05
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                        -2.198e+06   8.54e+04    -25.756      0.000   -2.37e+06   -2.03e+06
longitude                    -2.612e+04    999.231    -26.143      0.000   -2.81e+04   -2.42e+04
latitude                     -2.513e+04    999.301    -25.144      0.000   -2.71e+04   -2.32e+04
housing_median_age            1043.0871     43.035     24.238      0.000     958.734    1127.440
total_rooms                     -4.8298      0.771     -6.264      0.000      -6.341      -3.319
total_bedrooms                  71.9839      5.982     12.033      0.000      60.258      83.710
population                     -39.0977      1.061    -36.857      0.000     -41.177     -37.018
households                      76.9695      6.697     11.494      0.000      63.844      90.095
median_income                3.879e+04    332.420    116.677      0.000    3.81e+04    3.94e+04
ocean_proximity_INLAND      -3.909e+04   1705.920    -22.915      0.000   -4.24e+04   -3.57e+04
ocean_proximity_ISLAND       1.567e+05   3.08e+04      5.094      0.000    9.64e+04    2.17e+05
ocean_proximity_NEAR OCEAN   5448.6025   1520.233      3.584      0.000    2468.826    8428.379
==============================================================================
Omnibus:                     5145.355   Durbin-Watson:                   0.967
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19508.500
Skew:                           1.207   Prob(JB):                         0.00
Kurtosis:                       7.105   Cond. No.                     7.04e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.04e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [34]:
```
X2.head()
```

Out[34]:

	const	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity_INLAND	ocean_proximity_ISLAND	ocean_proximity_NEAR OCEAN
0	1.0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	0	0	0
1	1.0	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	0	0	0
2	1.0	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	0	0	0
3	1.0	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	0	0	0
4	1.0	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	0	0	0

we dont need constant column for sklearn package and hence let us drop it.

In [35]:
```
X=X2.drop('const',axis=1)
X.head()
```

Out[35]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity_INLAND	ocean_proximity_ISLAND	ocean_proximity_NEAR OCEAN
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	0	0	0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	0	0	0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	0	0	0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	0	0	0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	0	0	0

Let us use k-fold cross-validation.

In [38]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
cross_val_score(LinearRegression(),X,y,cv=5).mean()
```

Out[38]:
```
0.5814425872991967
```

Now, Let us perform For cross-validation using train-test split.

In [39]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 random_state=1,test_size=0.33)
```

In [40]:
```
# Let us build the model using using X_train and y_train and check the score using X_test and y_test
model = LinearRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[40]:
```
0.6370655724292966
```

Let us Check the R2_Square and other parameters

In [41]:
```
y_pred = model.predict(X_test)
from sklearn.metrics import r2_score,mean_squared_error
import math

print(f'R2 Score: {r2_score(y_test,y_pred)}')
print(f'Mean Squared Error, MSE: {mean_squared_error(y_test,y_pred)}')
print(f'RMSE: {math.sqrt(mean_squared_error(y_test,y_pred))}')
```

```
R2 Score: 0.6370655724292966
Mean Squared Error, MSE: 4747533703.419153
RMSE: 68902.3490413727
```

Let us check the actual values of median_house_value from the test dataset and their predicted values by the model.

In [42]:
```
y_test.head()
```

Out[42]:
```
4712     355000.0
2151      70700.0
15927    229400.0
82       112500.0
8161     225400.0
Name: median_house_value, dtype: float64
```

In [43]:
```
y_pred[:5]
```

Out[43]:
```
array([242155.90718898,  92675.46439436, 249763.0360777 , 163112.92242311,
       285761.89288663])
```

In [44]:
```
# Let us find the number of observations and number of features to be used in calculation for the adjusted R2
n = len(X_test)
k = len(X_test.iloc[0])
R2 = r2_score(y_test,y_pred)
R2
```

Out[44]:
```
0.6370655724292966
```

Let us now use Adjusted R^2 which is useful in multiple regression as it accounts for number of variables in the scoring.

In [45]:
```
Adj_R2 = 1 - ((n-1)*(1- R2)/(n-k-1))
print(f'The adjusted R2: {Adj_R2}')
```

```
The adjusted R2: 0.636478472619991
```

Inferences:

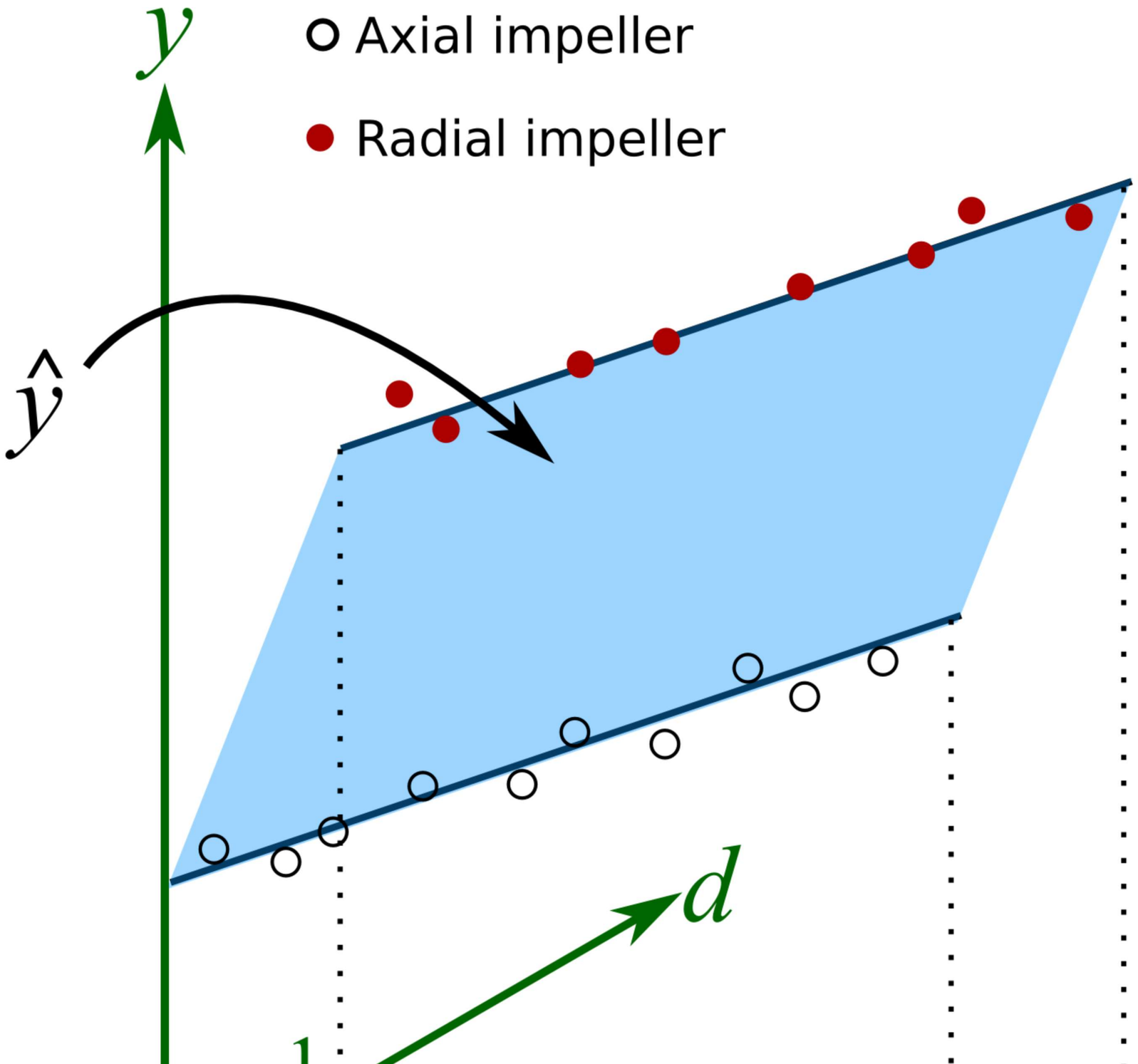
The formula of the model's regression line will be as below:

median_house_value = -2.198e+06 + (3.879e+04)*median_income* + *(1.567e+05)*ocean_proximity_ISLAND + (1043.0871)*housing_median_age* + *(71.9839)*total_bedrooms + (76.9695)*households* + *(5448.6025)*ocean_proximity_NEAR OCEAN + (-2.612e+04)*longitude* + *(-2.513e+04)*latitude + (-4.8298)*total_rooms* + *(-39.0977)*population + (-3.909e+04)*ocean_proximity_INLAND

The model seems to be a moderate one since the k-fold-cross-validation, R^2 value and Adjusted R^2 values are not substantially high.

Need to deal with the multi-collinearity among independent features to achieve better performance.

So, we might need to try for other machine learning models like decision tree to achieve better results.


y
Axial impeller
Radial impeller
ŷ
d


1
0
x_1