CALIFORNIA HOUSING PRICES DATASET

TASKS PERFORMED

1. DATA IMPORTING

2. DATA COLLECTION

3. UNDERSTANDING THE PROBLEM STATEMENT

4. EXPLORATORY DATA ANALYSIS

5. DATA CLEANING OR FEATURE ENGINEERING

1. DATA IMPORTING

Import Data and Required Packages

Importing Pandas, Numpy, Matplotlib, Seaborn and Warings Library and stats module such as scipy, shapiro and testmodules.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
import scipy
import matplotlib.image as mpimg
from pandas.plotting import scatter_matrix
from scipy.stats import norm
from numpy.random import randn
from statsmodels.stats.proportion import proportions_ztest
import plotly.graph_objects as go
from scipy.stats import spearmanr
from scipy.stats import shapiro
from scipy.stats import chi2_contingency

warnings.filterwarnings("ignore")

%matplotlib inline
```

1.1 reading the original dataset csv file in pandas dataframe format.

In [2]:

```
# comment - reading csv dataset of housing price.
# observation - using pandas library and converted file into DataFrames

df1 = pd.read_csv(r"C:\Users\Hp\Downloads\archive\housing.csv")
```

2. DATA COLLECTION

. The data pertains to the houses found in a given California district.

. The dataset is based on the 1990 census data.

. The dataset consists of 20640 rows and 10 columns.

. The data consists of both categorical and numerical data.

. Understanding the dataset only (for some cleaning).

3. PROBLEM STATEMENT

. Where the maximum population have houses near the ocean location and among population and households who has the maximum median income ?

. Which have Maximum count of houses from the highest group of population?

. Which location around ocean has highest median house value and where house is costlier?

. Where the population is dense near ocean locations ?

. How many peope have less than 10 (thousand USD) and still can afford the houses and where?

. How population manages to purchase houses even at less income?

. Where are the location of houses is highest near the ocean?

. Testing the relation between ocean different categories. (testing should we drop one category or not?)

. Testing independent and dependent features and also the normality. (to check how two different features are related)

. Testing multicollinearity among numerical and categorical features.

2.1 copying the original dataset into new DataFrame1.

In [3]:

```
# comment - copying the original dataset
# observation - this will ensure the intactness of the dataset and we can perform tasks in
df = df1.copy()
```

2.2 reading first five entries of the dataset.

In [4]:

```
# comment - head is the inbuilt function for having idea of the data for a quick glance.
# observation - head function by default provides only first five entries of the data only.
df.head()
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41.0	880.0	129.0	322.0	126
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259

2.3 detailed information of the dataset

In [5]:

```
# comment - this inbuilt info function provides complete understanding of the dataset.
# observation - checking the number of rows and columns,
                # null entries # datatypes # column names # memory usage or RAM of the data
# clear observation - there are 9 float features and 1 categorical or object features
                    # total_bedrooms has some missing values as the rows values doesnot ma
df.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
```

2.4 to check the size of the dataset .

In [6]:

```
# comment - checking the rows and column values.
# observation - there are 20640 rows and 10 columns in the dataset
df.shape
```

Out[6]:

```
(20640, 10)
```

2.5 missing values

solution : will be in data cleaning part

In [7]:

```
# comment - checking is there any null values in the dataset
# observation - total_bedroooms has 207 missing values out of 20640 rows entries.
df.isnull().sum()
```

Out[7]:

```
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
```

2.6 gathering all numerical data together

In [8]:

```
# comment - our data has only float as numerical datatypes
# observation - all 9 columns are float
df[df.dtypes[df.dtypes == 'float64'].index]
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20640 rows × 9 columns

problem detect:the total_rooms , total_bedrooms , population, households and income need to be of datatype int and not float

solution will be in data cleaning part.

2.7 Summary of the dataset

In [9]:

```
# comment - it provides the summary of only the numerical features.
# observation - # negative mean means that there are some negative values in the dataset of
                # std shows how much there is a deviation from the mean (can know the distr
                      # higher the std higher will be the deviation from the mean
                # min amd max are the lowest and highest values that a dataset can take
                # 25, 50 and 75 % of the data shows us the quartiles which also reflects th
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	5
longitude	20640.0	-119.569704	2.003532	-124.3500	-121.8000	-118.4
latitude	20640.0	35.631861	2.135952	32.5400	33.9300	34.2
housing_median_age	20640.0	28.639486	12.585558	1.0000	18.0000	29.0
total_rooms	20640.0	2635.763081	2181.615252	2.0000	1447.7500	2127.0
total_bedrooms	20433.0	537.870553	421.385070	1.0000	296.0000	435.0
population	20640.0	1425.476744	1132.462122	3.0000	787.0000	1166.0
households	20640.0	499.539680	382.329753	1.0000	280.0000	409.0
median_income	20640.0	3.870671	1.899822	0.4999	2.5634	3.5
median_house_value	20640.0	206855.816909	115395.615874	14999.0000	119600.0000	179700.0

2.6 having all the column names of the data

In [10]:

```
# comment - glance at all columns of the dataset.
# observation - there are total 10 columns.
df.columns
```

Out[10]:

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

4. EXPLORATORY DATA ANALYSIS

Feature Information

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. housing_Median_Age: Median age of a house within a block; a lower number is a newer building

4. total_Rooms: Total number of rooms within a block

5. total_Bedrooms: 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. median_Income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)

9. median_House_Value: Median house value for households within a block (measured in US Dollars)

10. ocean_Proximity: Location of the house w.r.t ocean/sea

4.1 segregating featues according to datatypes and have the details of the dataset

In [11]:

```
# comment - to segregate the data into categorical features
# observation - the only feature which contains category data is ocean_proximity which has
                # feature # it also has object datatype.

categorical_features = [fea for fea in df.columns if df[fea].dtypes == 'O']
print(f'we have {categorical_features} as our categorical feature')
```

```
we have ['ocean_proximity'] as our categorical feature
```

In [12]:

```
# comment - to segregate the data into numerical features.
# observation - the rest 9 columns are numerical and have float 64 as the datatype.
                  # might we can change the datatype if required to int.

numerical_features = [fea for fea in df.columns if df[fea].dtypes != 'O']
print(f'we have {numerical_features} as our numerical feature')
```

```
we have ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'tota
l_bedrooms', 'population', 'households', 'median_income', 'median_house_valu
e'] as our numerical feature
```

4.2 checking the index of the object feature

In [13]:

```
# comment - this will tell us what are the values and is there any null value.
# observation - there is no null value.
df[df.dtypes[df.dtypes == 'object'].index]
```

Out[13]:

	ocean_proximity
0	NEAR BAY
1	NEAR BAY
2	NEAR BAY
3	NEAR BAY
4	NEAR BAY
...	...
20635	INLAND
20636	INLAND
20637	INLAND
20638	INLAND
20639	INLAND

20640 rows × 1 columns

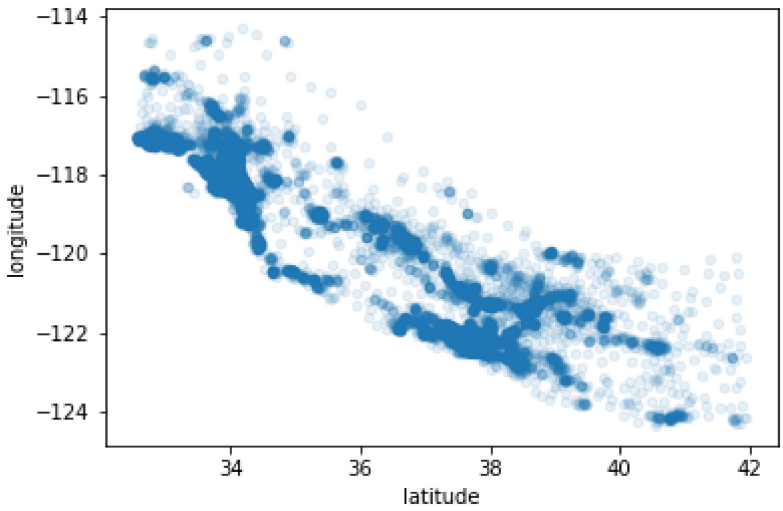
4.3 checking the scattering of latitude and longitude where it is high north or west.

In [14]:

```
# comment - population more scattered where shown by blue colour
# observation - dark blue portion for latitude is high at north and
                  # for longitude it is dark and scattered more at east region from -120 t
df.plot(kind = 'scatter', x = 'latitude', y = 'longitude', alpha = 0.1)
```

Out[14]:

```
<AxesSubplot:xlabel='latitude', ylabel='longitude'>
```



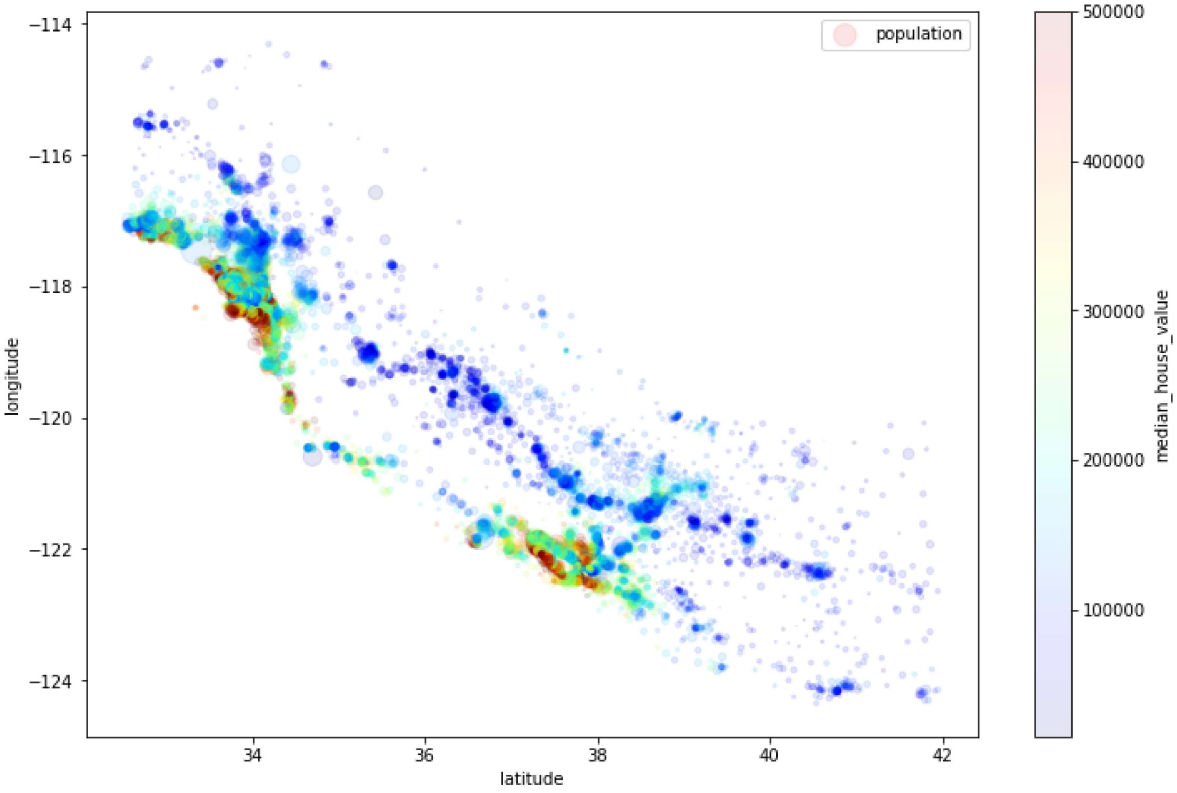
4.3.1 at these locations where population is dense and what is the median_house_value (costlier or cheap) at that area

In [15]:

```
# comment - to check population dense and where is the house is costlier
# observation - red colour shows the most dense and costly house.
                    # bubble is population and color indicates median_house_value
df.plot(kind = 'scatter', x = 'latitude', y = 'longitude', alpha = 0.1,
                  s = df['population']/100, label = 'population', figsize = (12,8), c = 'me
        cmap = plt.get_cmap('jet'), sharex = False)
plt.legend()
```

Out[15]:

```
<matplotlib.legend.Legend at 0x1f32af163d0>
```



4.3.2 maping the same analysis via California real map

In [16]:

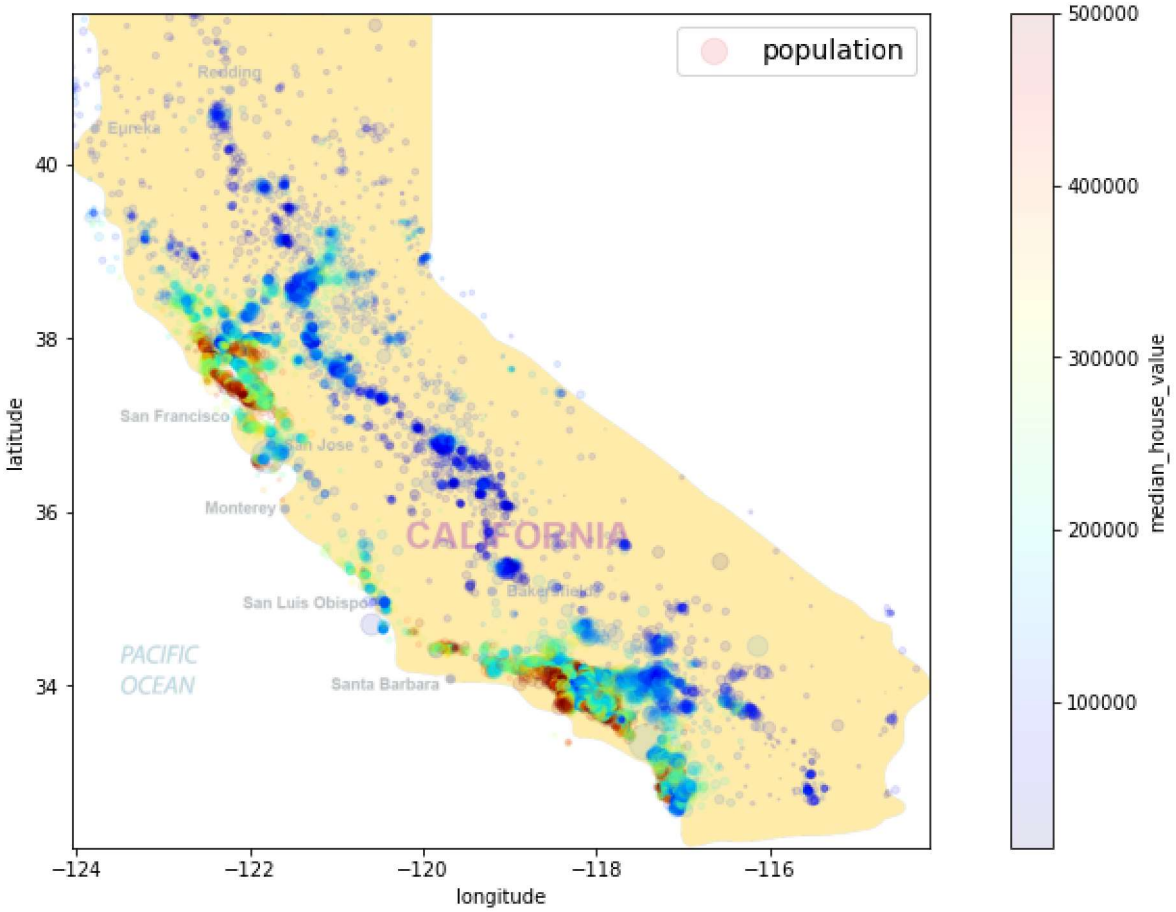
```
import matplotlib.image as mpimg
df.plot(kind = 'scatter', x = 'longitude', y = 'latitude', alpha = 0.1,
                  s = df['population']/100, label = 'population', figsize = (15,8),
             c = 'median_house_value' , cmap = plt.get_cmap('jet'), sharex = False)

california_img = mpimg.imread("california_map.png")

plt.imshow(california_img, extent = [-124.05 , -114.15, 32.13, 41.75], alpha = 0.5, cmap =

plt.ylabel = ('latitude')
plt.xlabel = ('longitude')

plt.legend(fontsize = 14)
plt.show()
```

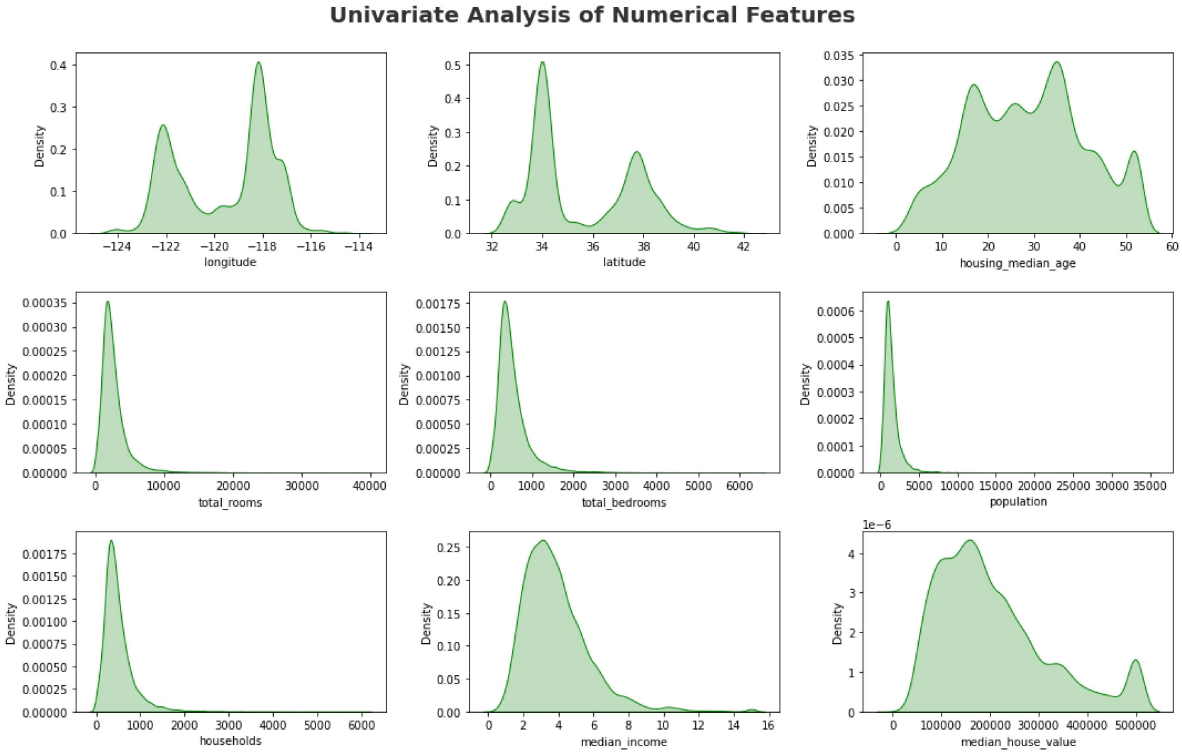


4.3 univariate analysis of numerical and categorical features

In [17]:

```
# comment - separate distribution of each and every numerical features
# observation - the index of numerical features from 3 to 7 are right skewed also known as
              # the index 0,1 are bimodal distributed and 2 is multimodal disrtibuted
plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20, fontweight='bold', a

for i in range(0, len(numerical_features)):
    plt.subplot(5, 3, i+1)
    sns.kdeplot(x=df[numerical_features[i]],shade=True, color='g')
    plt.xlabel = (numerical_features[i])
    plt.tight_layout()
```



problem detect : need to normalize the data of index [3:8] by changing scale

and transforming right skewed data to normally distributed

solution : will be in data cleaning part.

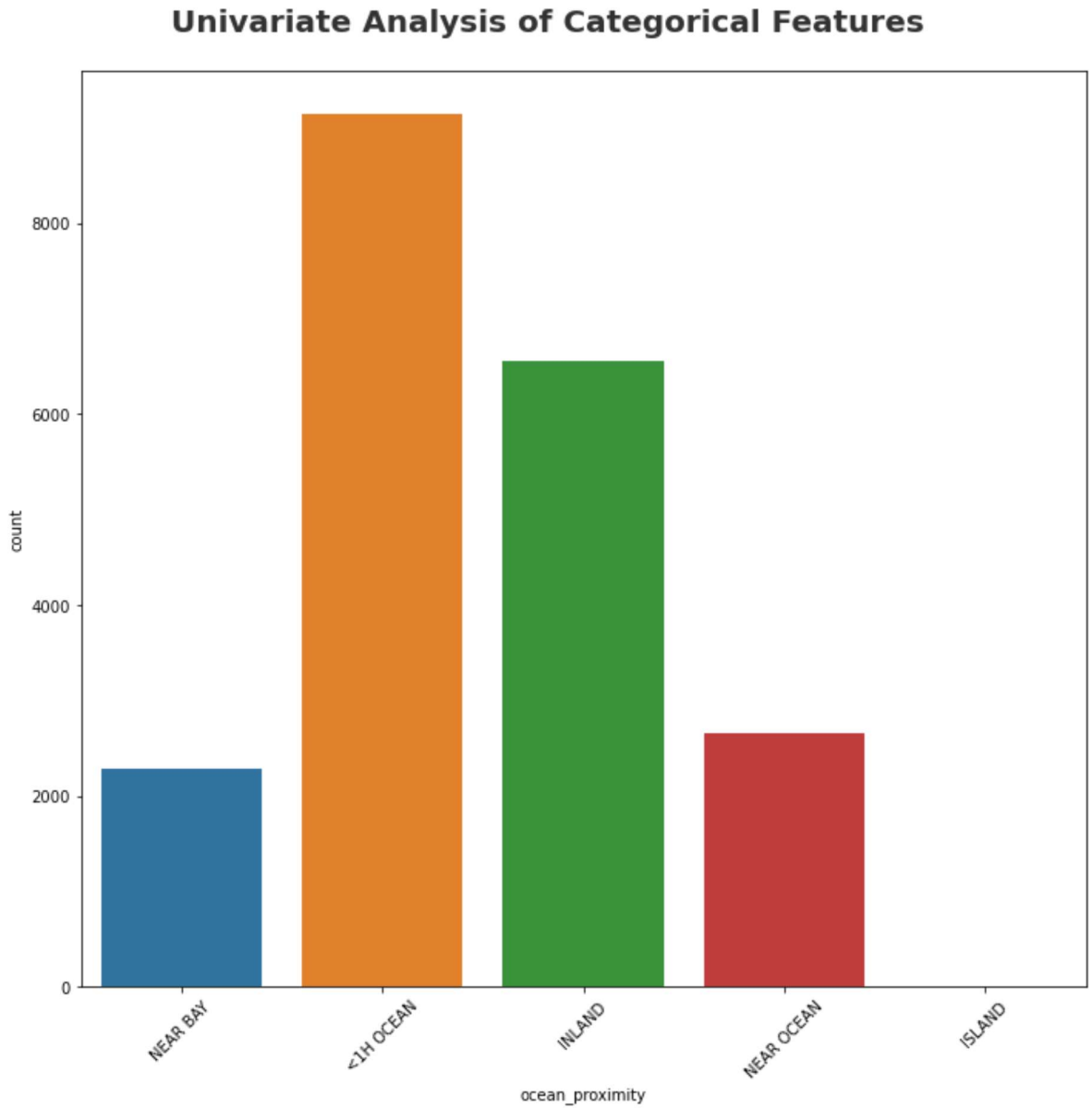
4.3.1 imbalance data also checked

solution : will be in data cleaning part.

In [18]:

```
# comment - checking where the location of houses is highest near the ocean_proximity
# observation - the analysis shows that population or households prefers <1H ocean location
                # second preference of people of California will be INLAND.
                # less to NEAR OCEAN and than NEAR BAY respectively
                # shows the data is imbalanced as island as less or nrgligible preferences.
plt.figure(figsize=(10, 10))
plt.suptitle('Univariate Analysis of Categorical Features', fontsize=20, fontweight='bold',

for i in range(0, len(categorical_features)):
    plt.subplot(1, 1, i+1)
    sns.countplot(x = df[categorical_features[i]])
    plt.xticks(rotation=45)
    plt.xlabel = (categorical_features[i])
    plt.tight_layout()
```



4.4 normalising the data of the ocean proximity feature for clear numerical picture

In [19]:

```
# comment - here we can easily analyse the percentage of the preferences of people for ocea
# observation - 44% Highest rank preference to <IH ocean location and so on.
for col in categorical_features:
    print(df[col].value_counts(normalize=True) * 100)
```

```
<1H OCEAN     44.263566
INLAND        31.739341
NEAR OCEAN    12.877907
NEAR BAY      11.094961
ISLAND         0.024225
Name: ocean_proximity, dtype: float64
```

4.5 BIvariate analysis of numerical and categorical features

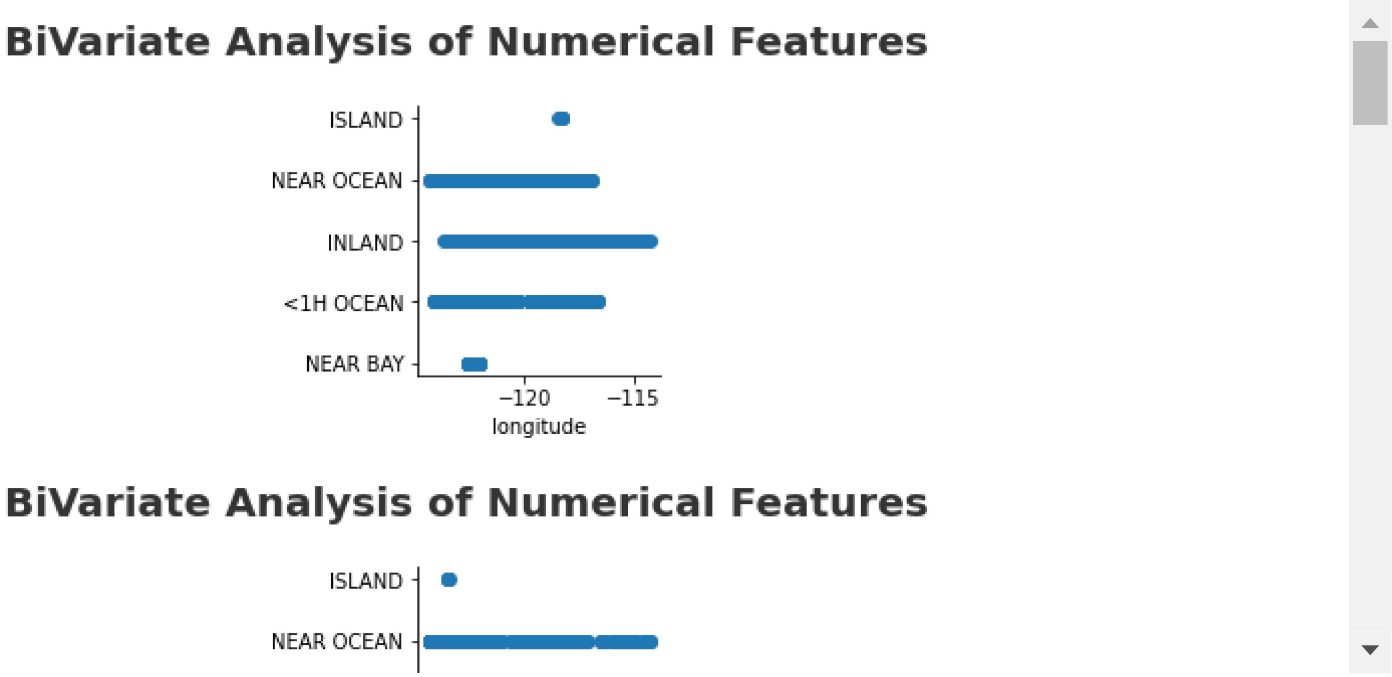
answering questions such as

1. at which ocean location houses are far from north and west (latitude and longitude)
2. is age of house differs according to ocean location
3. are rooms and bedrooms high where most population and households resides or vice versa
4. population is densely situated at which ocean location or which location popultaion of California prefers the most.
5. why does population prefers <1H ocean location only ?
6. does houses are cheaper at <1H ocean block?, as population prefers more this area!

solutions provided via observing graph under observation section below.

In [7]:

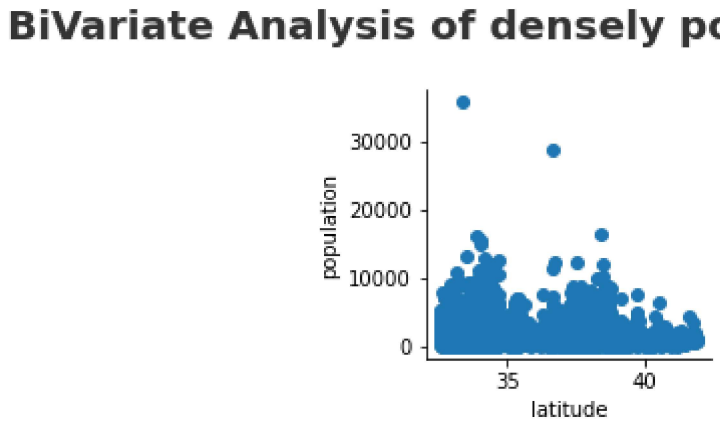
```
# comment - checking each numerical feature with categorical feature
# observation - # 1. longitude represents that in INLAND houses are farther west.
                # 1. latitude represents that in ISLAND houses are close to north
                      # and in NEAR WAY almost to near of north as compared to other locati
                # 2. housing median age reprsents that except in ISLAND almost at every loc
                # 3. total_rooms and total_bedroooms has high rooms in a block where most p
                # 4. popultaion and households are dense and prefers house in <1H ocean blo
                # 5. more income advantage near <1H ocean block and that's why most preferr
                # 6. the cost of the houses are same in <1H ocean block as others except IS
                     # ISLANDS are less preferrable, less population and still overpricing.
for i in range(0,len(numerical_features)):
    cat1 = ['ocean_proximity']
    for j in range(0,len(cat1)):
        sns.FacetGrid(df[numerical_features[i]]).map(plt.scatter, x = df[numerical_features
        plt.suptitle('BiVariate Analysis of Numerical Features', fontsize=20, fontweight='b
        plt.xlabel(numerical_features[i])
        plt.tight_layout()
```



4.5.1 At latitude the population is more that north or farther from north ?

In [8]:

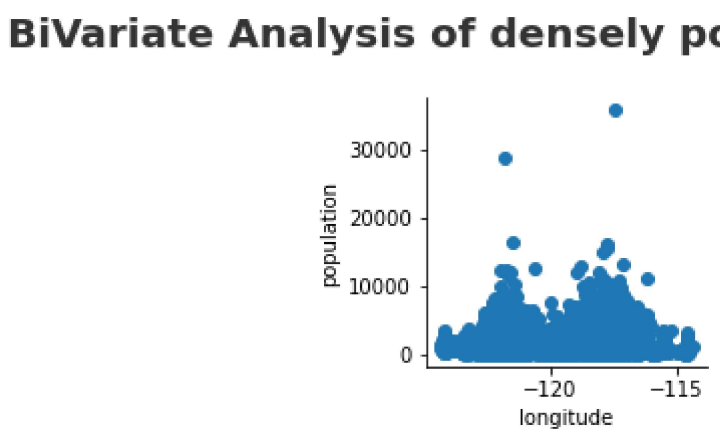
```
# comment - where population is dense at north or farther from north at latitude
# observation - the population is densely populated from 0 to 35 to north approx 15000 popu
                # and after 35 to 40 houses constructed farther from north and even les
sns.FacetGrid(df1[numerical_features]).map(plt.scatter, x = df1['latitude'], y = df1['popul
plt.suptitle('BiVariate Analysis of densely populated area', fontsize=20, fontweight='bold'
plt.xlabel('latitude')
plt.ylabel('population')
plt.tight_layout()
```



4.5.2 At longitude where the population is dense at west or farther from west?

In [9]:

```
# comment - where population is dense at north or farther from west at longitude
# observation - people of California prefers farther from west location but not near to wes
sns.FacetGrid(df[numerical_features]).map(plt.scatter, x = df['longitude'], y = df['populat
plt.suptitle('BiVariate Analysis of densely populated area', fontsize=20, fontweight='bold'
plt.xlabel('longitude')
plt.ylabel('population')
plt.tight_layout()
```



clear observation from 4.5.1 to 4.5.2 is that California population prefers the houses near north east and south east

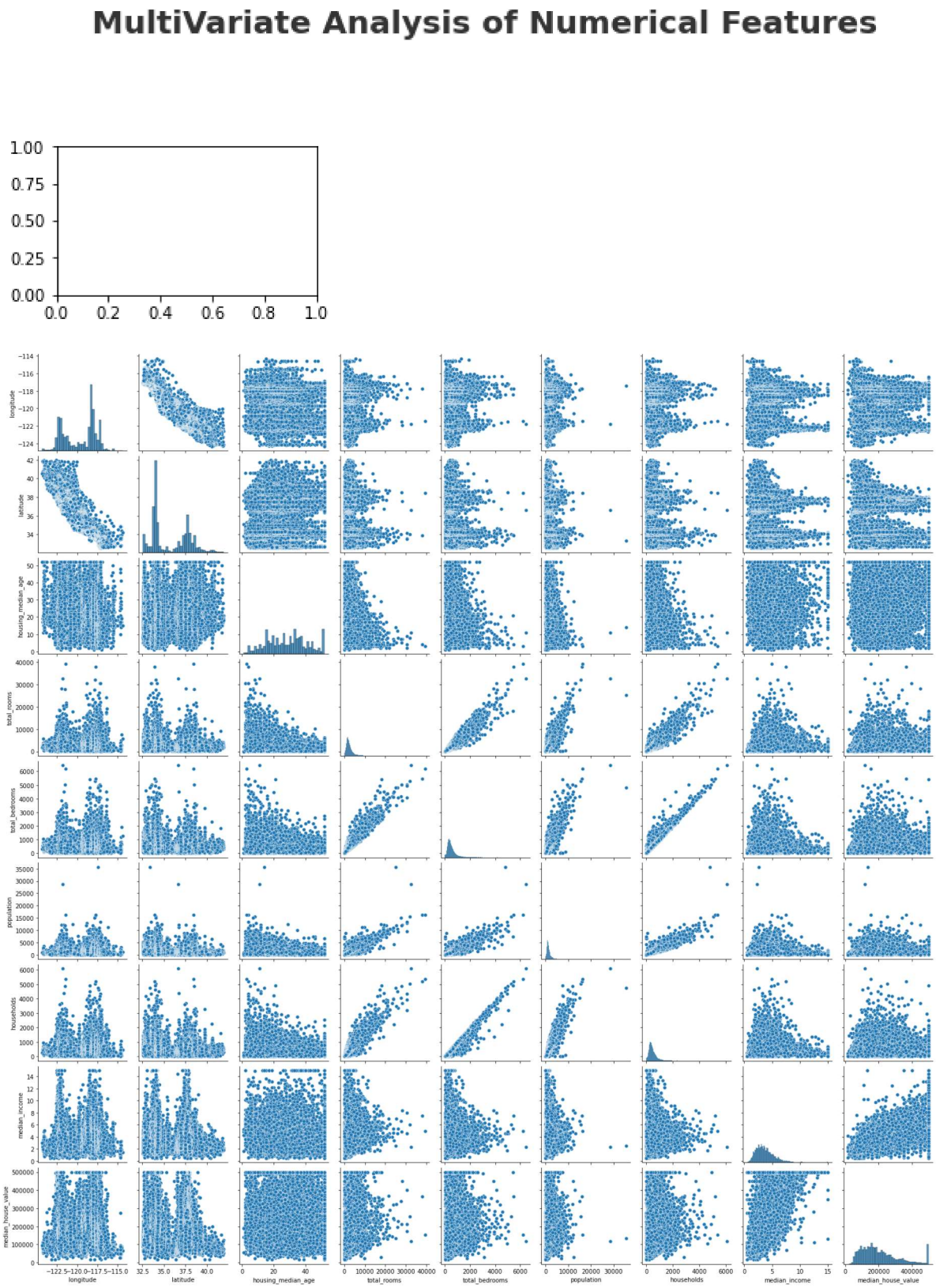
but farther from west and some what close choice to north and south.

4.6 Multivariate Analysis

Multivariate analysis is the analysis of more than one variable.

In [20]:

```
# comment - analysing more than one numerical features together
# observation - # by looking at graph some relations are clear at there is positive realtio
                    # relation of population with total_rooms , total_bedrooms and househol
                # the same features together shows normal or skewness distributions
plt.figure(figsize=(10, 10))
cat1 = ['ocean_proximity']
for i in range(0, len(cat1), len(numerical_features)):
    plt.subplot(5, 3, i+1)
    plt.suptitle('MultiVariate Analysis of Numerical Features', fontsize=20, fontweight='bo
    sns.pairplot(df)
```



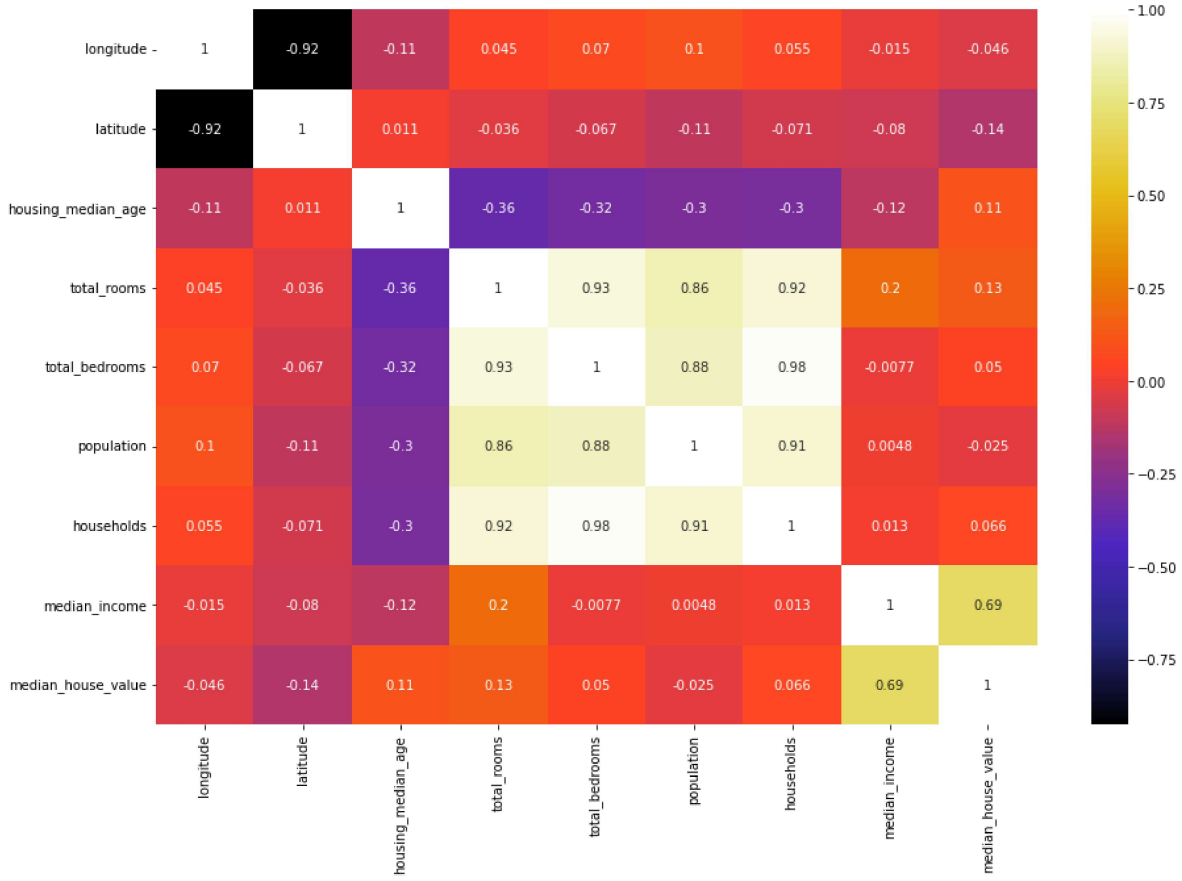
from 4.6 problem : cannot interpret much numerical behaviour

```
#### solution : we will use correlation along with heat map for be
tter understanding
#### correlation scale [-1(perfect negative) to 1 (perfect positiv
e)]
```

In [21]:

```
# comment - correlation of all numerical features together
# observation - # the diagonal are always one and sjows perfect correlation of +1
                # the black colour shows very weak negative correlatio
    # the light yellowish shade(in the centre) shows very strong positive relation between
                    # population (total_rooms, total_bedrooms and households)
                    # total_rooms and total_bedrooms
                    # households (total_rooms, total_bedrooms and population)

    # 0.69 strong positive correlation between median_house_value (USD) and median_income(t
plt.figure(figsize = (15,10))
sns.heatmap(df.corr(), cmap="CMRmap", annot=True)
plt.show()
```



4.6.2 understanding in detail more about median_house_value.

In [22]:

```
# comment - checking specially for median_house_value relation how they are correlated
# observation - # population consider income as a factor before purchasing a house (positiv
                    ## high income expensive house and vice versa
        #INTERESTING to see ----- weak negative relation with popultaion as less population
                          ## and high population, houses are less expensive
                                ### not perfect but somewhat values of house can alter with

corr_matrix = df.corr()
corr_matrix['median_house_value'].sort_values(ascending = False)
```

Out[22]:

```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

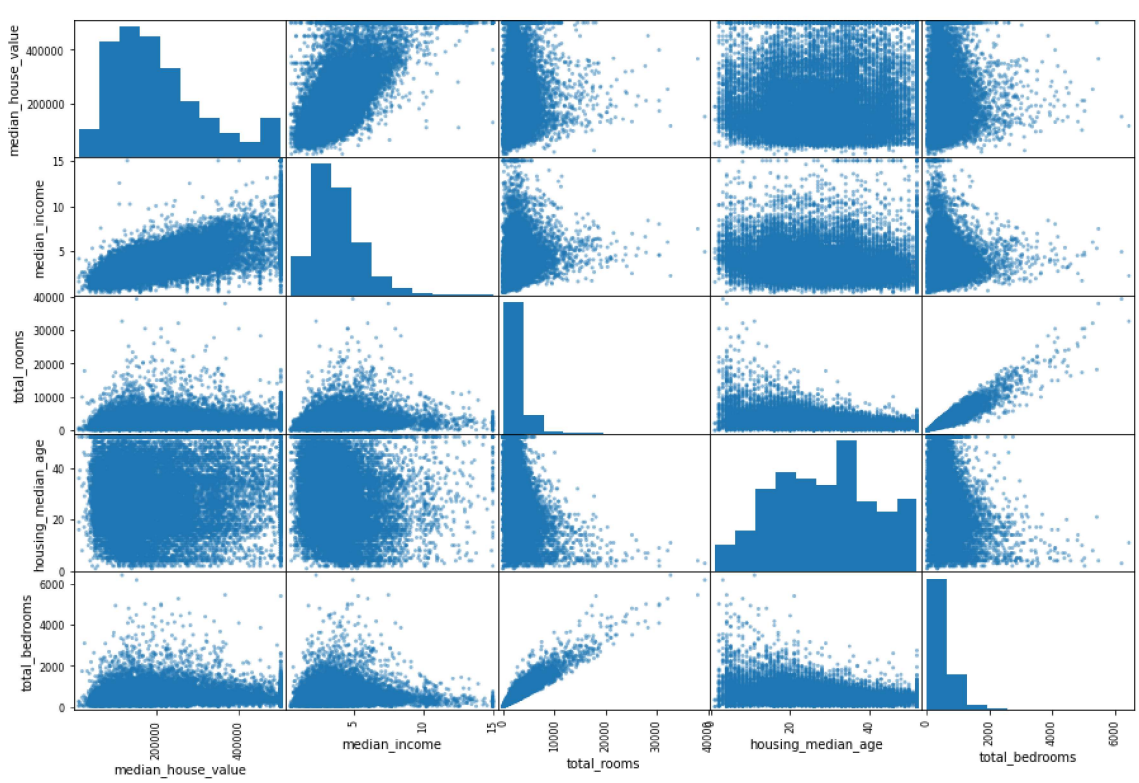
4.6.2.1 taking important attribute median_house_valuem and checking the relation via graph analysis

In [23]:

```
imp_attr = ['median_house_value', 'median_income', 'total_rooms', 'housing_median_age','tot
from pandas.plotting import scatter_matrix
scatter_matrix(df[imp_attr], figsize = (15,10))
```

Out[23]:

```
array([[<AxesSubplot:xlabel='median_house_value', ylabel='median_house_valu
e'>,
        <AxesSubplot:xlabel='median_income', ylabel='median_house_value'>,
        <AxesSubplot:xlabel='total_rooms', ylabel='median_house_value'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='median_house_valu
e'>,
        <AxesSubplot:xlabel='total_bedrooms', ylabel='median_house_value'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='median_income'>,
        <AxesSubplot:xlabel='median_income', ylabel='median_income'>,
        <AxesSubplot:xlabel='total_rooms', ylabel='median_income'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='median_income'>,
        <AxesSubplot:xlabel='total_bedrooms', ylabel='median_income'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='total_rooms'>,
        <AxesSubplot:xlabel='median_income', ylabel='total_rooms'>,
        <AxesSubplot:xlabel='total_rooms', ylabel='total_rooms'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='total_rooms'>,
        <AxesSubplot:xlabel='total_bedrooms', ylabel='total_rooms'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='housing_median_ag
e'>,
        <AxesSubplot:xlabel='median_income', ylabel='housing_median_age'>,
        <AxesSubplot:xlabel='total_rooms', ylabel='housing_median_age'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='housing_median_ag
e'>,
        <AxesSubplot:xlabel='total_bedrooms', ylabel='housing_median_age'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='total_bedrooms'>,
        <AxesSubplot:xlabel='median_income', ylabel='total_bedrooms'>,
        <AxesSubplot:xlabel='total_rooms', ylabel='total_bedrooms'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='total_bedrooms'>,
        <AxesSubplot:xlabel='total_bedrooms', ylabel='total_bedrooms'>]],
      dtype=object)
```



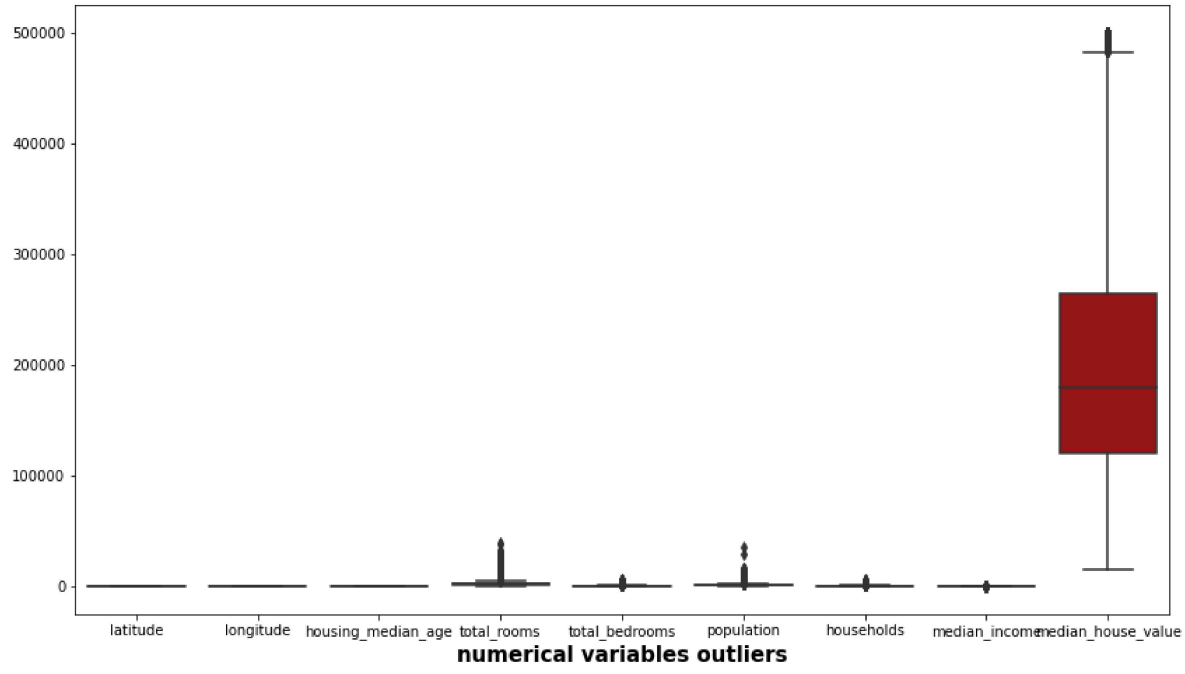
4.7 checking the outliers in the data of numerical features

In [24]:

```
# comment - Plotting boxplot for numerical features
# observation - numerical_features index [3:9] have outliers in the data and starting three
plt.figure(figsize = (14,8))
ax = sns.boxplot(data = df[['latitude', 'longitude','housing_median_age', 'total_rooms', 't
                 palette="OrRd")
ax.set_xlabel('numerical variables outliers', weight="bold", fontsize=15)
```

Out[24]:

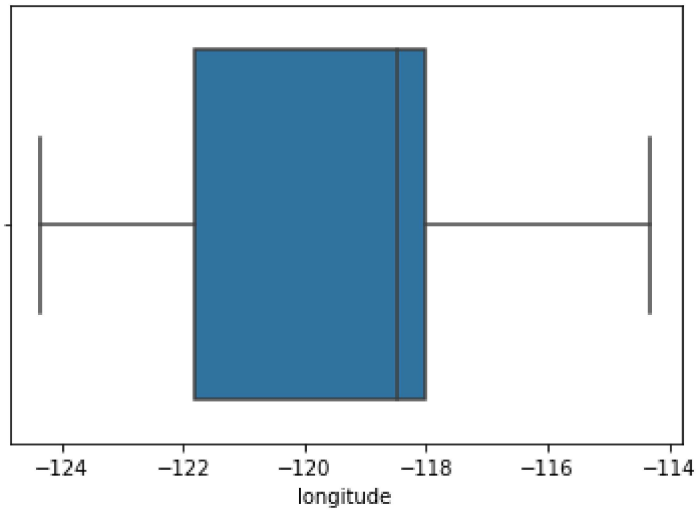
```
Text(0.5, 0, 'numerical variables outliers')
```



4.7.1 clear picture for range of outliers separately of each numerical features.

In [25]:

```
# comment - Plotting boxplot for numerical features
# observation - numerical_features index [3:9] have outliers in the data and starting three
for i in numerical_features:
    sns.boxplot(df[i])
    plt.show()
```



problem detect : outliers in the dataset from last 6 features.

solution : will be in data cleaning part.

analysis till now for data collection

1. missing value in total_bedrooms feature

2. have to change the datatypes of some columns to int

3. have to handle the outliers

4. need to normalise the data (scaling) and some transformation is required

5. imbalance data

A. testing independency, normality and mulicollinearity

A.1 checking multicollinearity that how features are related to numericala and categorical

testing hypothesis for multicollinearity via chi square test

```
H0 (null hypothesis) :  shows correlative relationships when independent variables
H1 (alternative hypothesis) :  not shows correlative relationships when independent
variables
```

In [26]:

```
# comment - chi square test shows the relation between features and here we are checking fo
# observation - for index 0,1,2 and 8 we are able to reject H0 (which is good) as independe
                ## for index 3 to 7 we fail to reject as these factors might affect the rel
chi2_test = []
for feature in numerical_features:
    if chi2_contingency(pd.crosstab(df['ocean_proximity'], df[feature]))[1] < 0.05:
        chi2_test.append('Reject Null Hypothesis')
    else:
        chi2_test.append('Fail to Reject Null Hypothesis')
result = pd.DataFrame(data=[numerical_features, chi2_test]).T
result.columns = ['Column', 'Hypothesis Result']
result
```

Out[26]:

	Column	Hypothesis Result
0	longitude	Reject Null Hypothesis
1	latitude	Reject Null Hypothesis
2	housing_median_age	Reject Null Hypothesis
3	total_rooms	Fail to Reject Null Hypothesis
4	total_bedrooms	Fail to Reject Null Hypothesis
5	population	Fail to Reject Null Hypothesis
6	households	Fail to Reject Null Hypothesis
7	median_income	Fail to Reject Null Hypothesis
8	median_house_value	Reject Null Hypothesis

problem detect : we need to remove multicollinearity among features

solution : will be in data cleaning.

A.2 checking the dependent and independent variables or features via test and graph

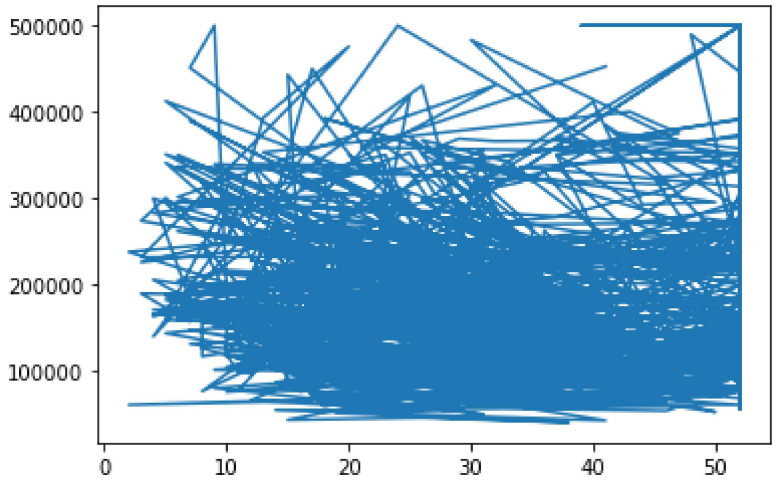
assumptions of multicollinearity is that it doesnot assume corr
elation

```
setting dependent variable and independent variable meaning
dependent variable - there is a relation
independent variable - there is no relation.
```

In [27]:

```
# comment - taking sample size for better analysis of the data.
# observation - considering data sample size upto 1500 for two numerical features
                        ## and will check will they affect each other or no affect
    ### graph shows that there is no dependency between these features for each other.

first_sample = df[0:1500]['housing_median_age']
second_sample = df[0:1500]['median_house_value']
plt.plot(first_sample,second_sample)
plt.show()
```



let us check the relation via testing spearman correlation.

In [28]:

```
# comment - spearman will help to test the dependency or independency of features.
# observation - independent variables which means that housing age has no realtion in deter
stat , p = spearmanr(first_sample,second_sample)
print('stat=%.3f, p=%5f' % (stat,p))
if p > 0.05:
    print('independent sample')
else:
    print('dependent sample')
```

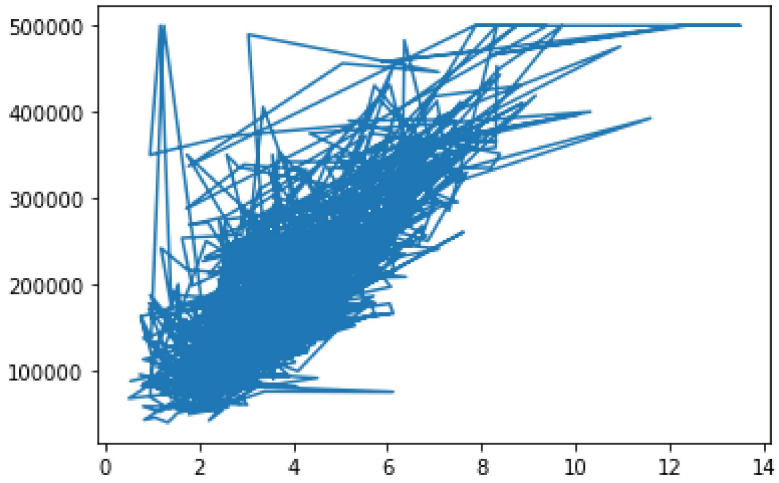
```
stat=0.024, p=0.343514
independent sample
```

A.2.1 some more spearsman test on different features

In [29]:

```
# comment - taking sample size for better analysis of the data.
# observation - considering data sample size upto 1500 for two numerical features
                        ## and will check will they affect each other or no affect
    ### graph shows that there is positive dependency between these features for each other

first_sample1 = df[0:1500]['median_income']
second_sample1 = df[0:1500]['median_house_value']
plt.plot(first_sample1,second_sample1)
plt.show()
```



In [30]:

```
# comment - spearman will help to test the dependency or independency of features.
# observation - dependent variable shows that there is affect of each other while consideri
                    ## income will helps us to see how much value of house can be afforded
                                ### higher the income higher the purchase of house value an
stat , p = spearmanr(first_sample1,second_sample1)
print('stat=%.3f, p=%5f' % (stat,p))
if p > 0.05:
    print('independent sample')
else:
    print('dependent sample')
```

```
stat=0.729, p=0.000000
dependent sample
```

A.3 checking the distributions of mean with two numerical features and one categorical feature via ANNOVA test

```
#### ANOVA TEST HYPOTHESIS
H0 : if the means of all are near then means are equal
H1 : not near and nor equal
```

In [31]:

```
# comment - random integer sample for the data
# observation - 10000 sample space data for these features.
median_income = np.random.randn(10000)
median_house_value = np.random.randn(10000)
ocean_proximity = np.random.randn(10000)
```

In [32]:

```
# comment - the means of these features are almost samely distributed
# observation -
tstat, p = scipy.stats.f_oneway(median_income,median_house_value, ocean_proximity)
print('stat=%.3f, p=%.3f' % (tstat,p))
if p > 0.05:
    print('different distribution of means')
else:
    print('same distribution of means')
```

```
stat=0.749, p=0.473
different distribution of means
```

problem detect : the data is not equally distributed around mean for various features.

A.4 checking subcategories of ocean proximity relation

In [33]:

```
# RANDOM VARIABLE
norm.ppf(0.95)
```

Out[33]:

```
1.6448536269514722
```

In [34]:

```
# comment - the 10000 observation sample is drawn.
# observation - for ztest huge sample is required.
data_sample = randn(10000)
data_sample
```

Out[34]:

```
array([-1.85840568,  0.03007812, -0.05857958, ..., -1.36451585,
       -1.95650666, -0.12883557])
```

```
## hypothesis testing
H0 : there is no difference in proportion of ocean_proximity of <1H ocean and Island
H1 : there is a difference in proportion of ocean_proximity of <1H ocean and Island
```

In [35]:

```
# comment - whether we can reject the island data (less data) or it has some impact
# observation - we reject H0 which means there is a differnce and island can have separate
                    # not removing from data.
number_of_success = np.array([4400, 240])
total_sample_size = np.array([10000, 10000])
(t_stat, p_value) = proportions_ztest(number_of_success, total_sample_size, alternative = '
print(f'the computed ztest_statistic is {t_stat}')
print(f'p value is {p_value}')

if p_value < 0.05:
    print('reject null hypothesis')
else:
    print('fail to reject null hypothesis')
```

```
the computed ztest_statistic is 69.68731476445666
p value is 0.0
reject null hypothesis
```

A.5 checking which features shows normally distribution or not

normality test

```
## hypothesis testing
H0: the data is normally distributed
H1: the data is not normally distributed
```

In [37]:

```
# comment - with p value the data is checked for normality
# observation - every feature here has p value less than 0.05 so they all features has no e
data_to_test = df['housing_median_age']
stat , p = shapiro(data_to_test)
print('stat=%.2f, p=%.3f' % (stat,p))
if p > 0.05:
    print('housing_median_age is NORMAL DISTRIBUTION')
else:
    print('fail to show NORMALITY')
```

```
stat=0.98, p=0.000
fail to show NORMALITY
```

In [38]:

```
data_to_test1 = df['median_house_value']
stat , p = shapiro(data_to_test1)
print('stat=%.2f, p=%.30f' % (stat,p))
if p > 0.05:
    print('median_house_value is NORMAL DISTRIBUTION')
else:
    print('fail to show normality')
```

```
stat=0.91, p=0.000000000000000000000000000000
fail to show normality
```

In [39]:

```
data_to_test2 = df['latitude']
stat , p = shapiro(data_to_test2)
print('stat=%.2f, p=%.30f' % (stat,p))
if p > 0.05:
    print('latitude is NORMAL DISTRIBUTION')
else:
    print('fail to show normality')
```

```
stat=0.88, p=0.000000000000000000000000000000
fail to show normality
```

problem detect : that none of the feature is normally distributed

solution will be in data cleaning.

A.6 want to check the distribution of latitude and longitude is same or not.

In [40]:

```
# comment - creating random samples
latitude = np.random.randn(10000)
longitude = np.random.randn(10000)
```

```
assumption : does not assume that the data should be normally only.
```

In [41]:

```
# comment - checking the two features has same distribution or not
# observation - same distribution implies same skewness of these two features.
zstat, p = scipy.stats.mannwhitneyu(latitude, longitude)
print('stat=%.3f, p=%.3f' % (zstat,p))
if p > 0.05:
    print('same distribution')
else:
    print('different distribution')
```

```
stat=50104636.000, p=0.798
same distribution
```

note- for t-test sample size is too small which results not clear understanding of the relation

and that's why used 10000 observations and used z test for clear and accurate result to population

```
### ------------------ data cleaning and problem set solutions on next ipynb file -------
-----------.
```

In [ ]:

```

```

In [ ]:

```

```