project_housing

May 9, 2018

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

In [2]: housing_df=pd.read_csv("housing.csv")
        housing_df.tail()

Out[2]:        longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
        20635    -121.09     39.48                  25         1665           374.0
        20636    -121.21     39.49                  18          697           150.0
        20637    -121.22     39.43                  17         2254           485.0
        20638    -121.32     39.43                  18         1860           409.0
        20639    -121.24     39.37                  16         2785           616.0

               population  households  median_income ocean_proximity  \
        20635         845         330         1.5603          INLAND
        20636         356         114         2.5568          INLAND
        20637        1007         433         1.7000          INLAND
        20638         741         349         1.8672          INLAND
        20639        1387         530         2.3886          INLAND

               median_house_value
        20635               78100
        20636               77100
        20637               92300
        20638               84700
        20639               89400
```

Data preprocessing

```
In [3]: housing_df.isnull().any()

Out[3]: longitude             False
        latitude              False
        housing_median_age    False
        total_rooms           False
        total_bedrooms         True
```

```
population            False
households            False
median_income         False
ocean_proximity       False
median_house_value    False
dtype: bool
```

##
here we find the column total_bedrooms have null value

In [4]:
```
housing_df.isna().any()
```

Out[4]:
```
longitude             False
latitude              False
housing_median_age    False
total_rooms           False
total_bedrooms         True
population            False
households            False
median_income         False
ocean_proximity       False
median_house_value    False
dtype: bool
```

Impute mean of column value in place of missing value

In [5]:
```
housing_df.total_bedrooms=housing_df.total_bedrooms.fillna(housing_df.total_bedrooms.mea
housing_df.isna().any()
```

Out[5]:
```
longitude             False
latitude              False
housing_median_age    False
total_rooms           False
total_bedrooms        False
population            False
households            False
median_income         False
ocean_proximity       False
median_house_value    False
dtype: bool
```

Exploratory Data analysis

In [6]:
```
housing_df.describe()
```

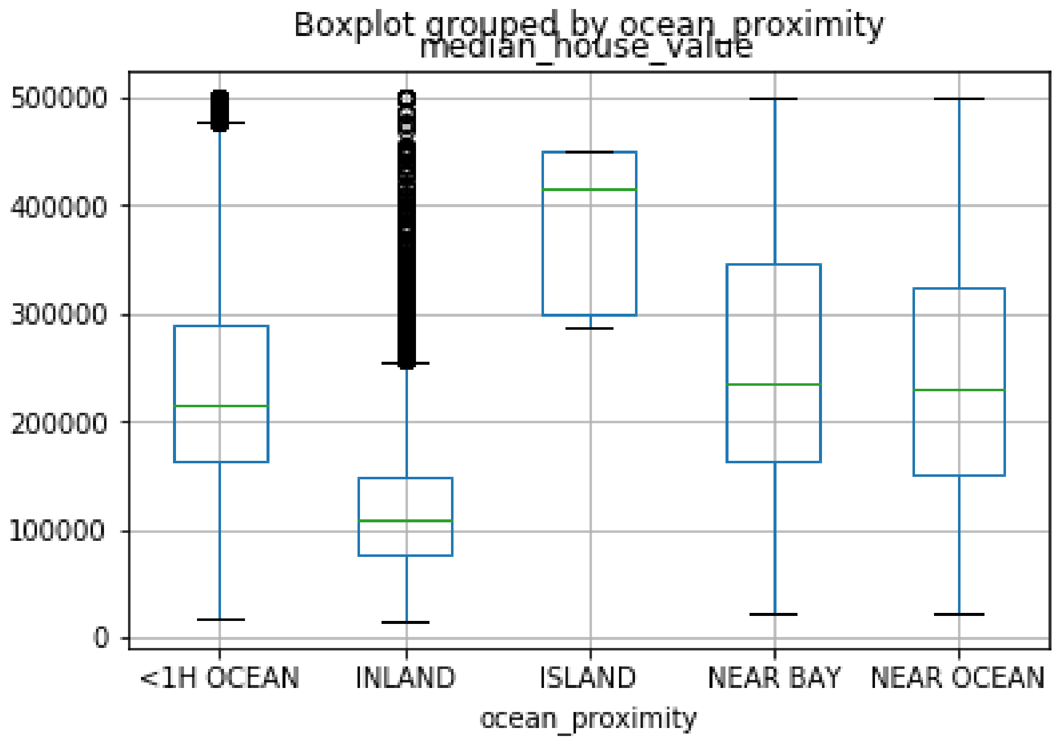
Out[6]:
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000   
mean    -119.569704     35.631861           28.639486   2635.763081   
std        2.003532      2.135952           12.585558   2181.615252   
```

```
min    -124.350000    32.540000            1.000000       2.000000
25%    -121.800000    33.930000           18.000000    1447.750000
50%    -118.490000    34.260000           29.000000    2127.000000
75%    -118.010000    37.710000           37.000000    3148.000000
max    -114.310000    41.950000           52.000000   39320.000000

       total_bedrooms    population    households  median_income  \
count    20640.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        419.266592   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        297.000000    787.000000    280.000000       2.563400
50%        438.000000   1166.000000    409.000000       3.534800
75%        643.250000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

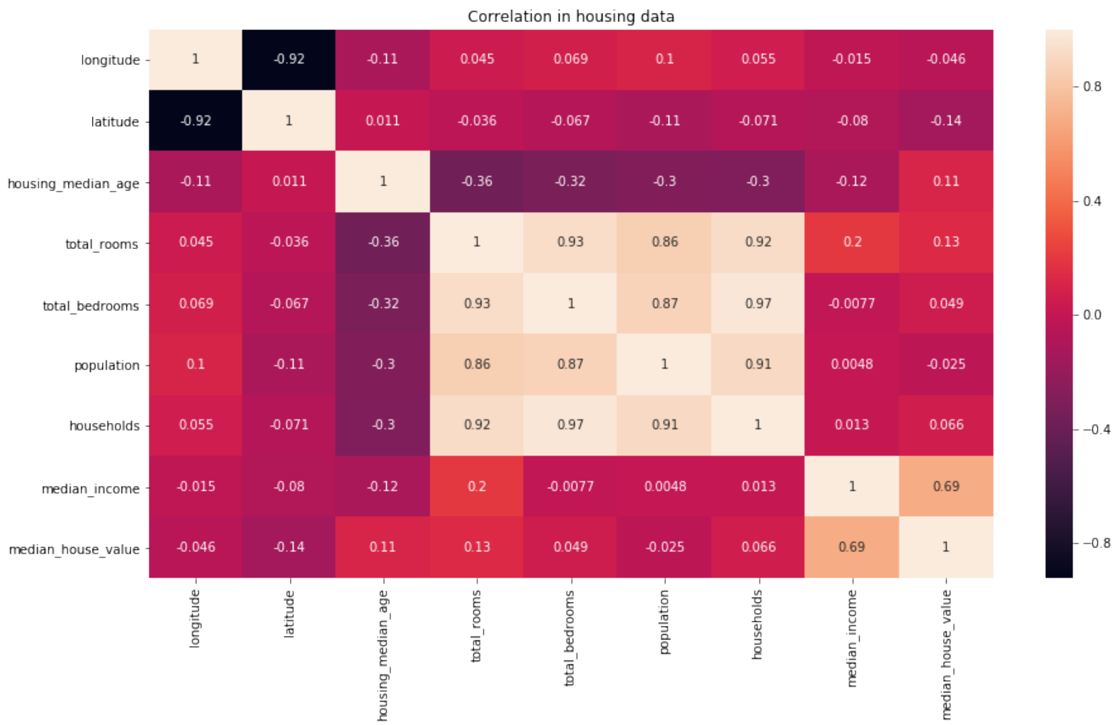
Bivariate analysis

In [7]:
```python
housing_df.boxplot(column='median_house_value',by='ocean_proximity')
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f531e00f748>
```



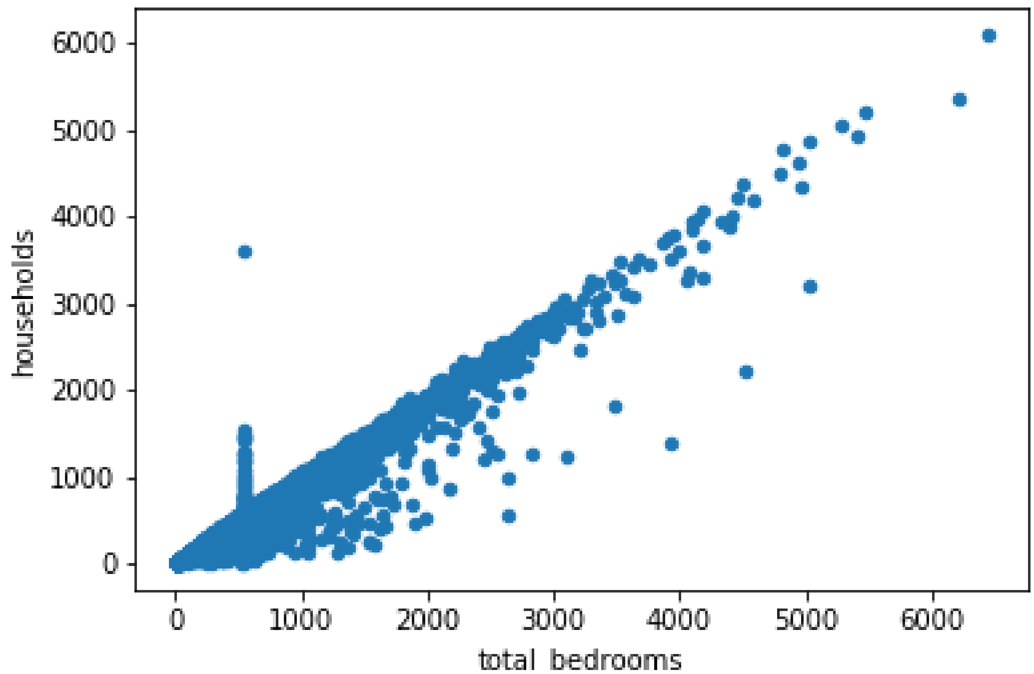

```
In [8]: import seaborn as sns

        plt.figure(figsize=(15,8))


        corr = housing_df.corr()
        sns.heatmap(corr,
                    xticklabels=corr.columns.values,
                    yticklabels=corr.columns.values, annot=True)
        plt.title('Correlation in housing data')
        plt.plot()

Out[8]: []
```



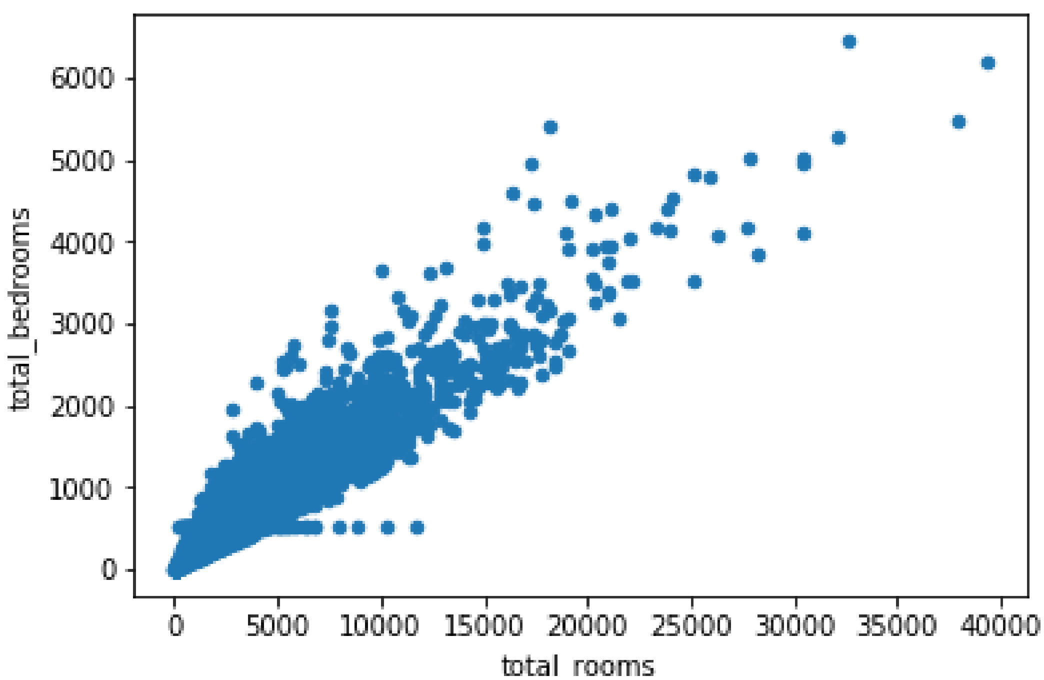
from above diagram we found that there is high correlation between

At this point we can drop households and total_rooms columns while creating model; since other columns are present which can convey similar information

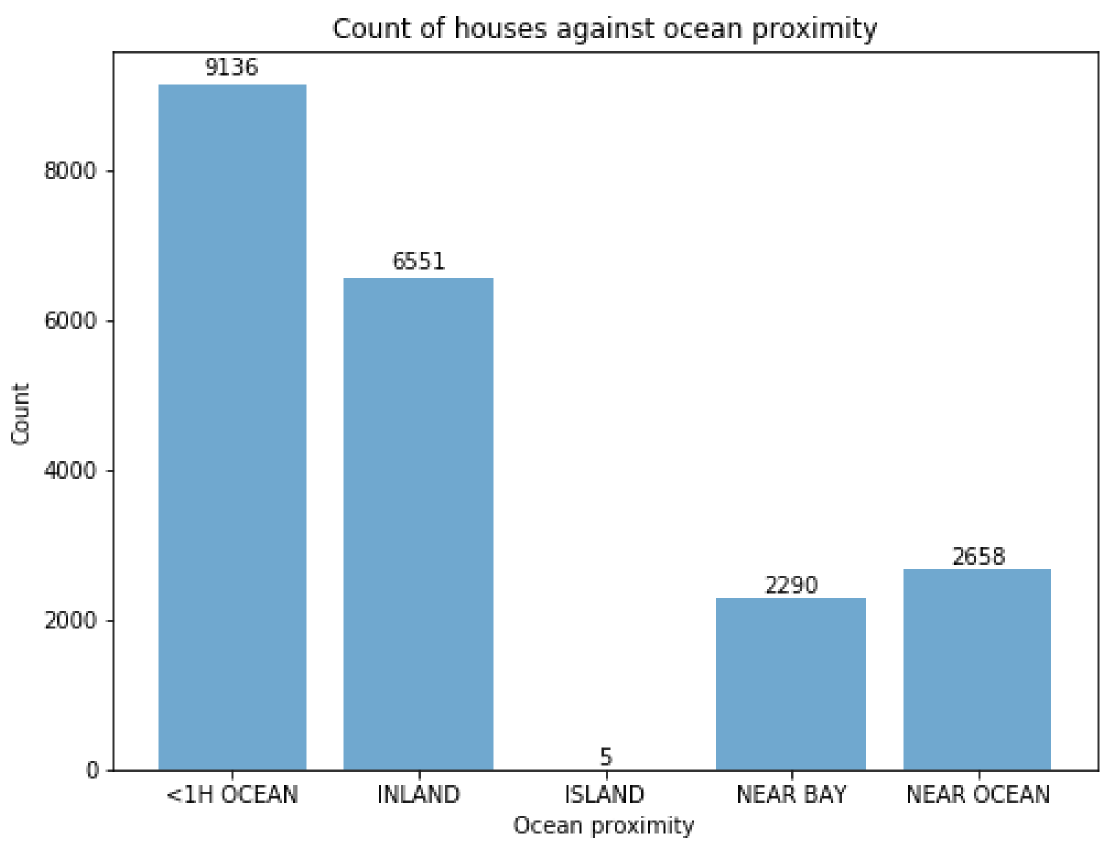
```
In [9]: housing_df.plot.scatter('total_bedrooms','households')

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f530b97ce48>
```




```
In [10]: housing_df.plot.scatter('total_rooms','total_bedrooms')

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f530b954c88>
```

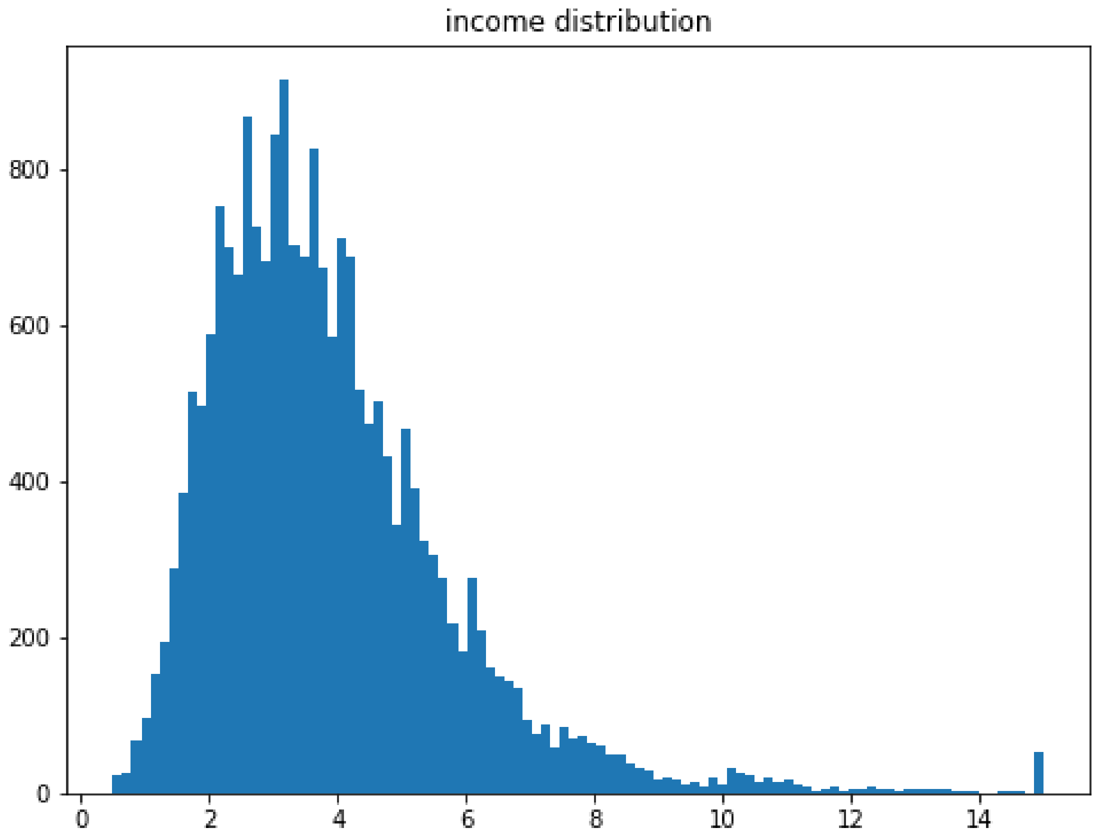


```
In [11]: def autolabel(rects):
             for rect in rects:
                 h = rect.get_height()
                 ax.text(rect.get_x()+rect.get_width()/2., 1.01*h, '%d'%int(h),
                         ha='center', va='bottom')

In [12]: fig=plt.figure(figsize=(8,6))
         ax=fig.add_subplot(111)
         s=housing_df.ocean_proximity.value_counts()


         rect=ax.bar(s.index,s,alpha=0.64)
         plt.title('Count of houses against ocean proximity')
         plt.xlabel('Ocean proximity')
         autolabel(rect)
         plt.ylabel('Count')
         plt.show()
```



In [13]:
```
plt.figure(figsize=(8,6))
plt.hist(housing_df.median_income,bins=100)
plt.title('income distribution')
plt.plot()
```

Out[13]: []



Encode the categorial data. So now instead of character values we will have numerical value

In [14]:
```
housing_df.head()
```

Out[14]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41	880	129.0	
1	-122.22	37.86	21	7099	1106.0	
2	-122.24	37.85	52	1467	190.0	
3	-122.25	37.85	52	1274	235.0	
4	-122.25	37.85	52	1627	280.0	

	population	households	median_income	ocean_proximity	median_house_value
0	322	126	8.3252	NEAR BAY	452600
1	2401	1138	8.3014	NEAR BAY	358500
2	496	177	7.2574	NEAR BAY	352100

```
3          558          219          5.6431        NEAR BAY                341300
4          565          259          3.8462        NEAR BAY                342200
```

In [15]:
```
from sklearn.preprocessing import LabelEncoder
x_labelencoder = LabelEncoder()
housing_df.ocean_proximity=x_labelencoder.fit_transform(housing_df.ocean_proximity)
housing_df.head()
```

Out[15]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                  41          880           129.0
1    -122.22     37.86                  21         7099          1106.0
2    -122.24     37.85                  52         1467           190.0
3    -122.25     37.85                  52         1274           235.0
4    -122.25     37.85                  52         1627           280.0

   population  households  median_income  ocean_proximity  median_house_value
0         322         126         8.3252                3              452600
1        2401        1138         8.3014                3              358500
2         496         177         7.2574                3              352100
3         558         219         5.6431                3              341300
4         565         259         3.8462                3              342200
```

In [16]:
```
x_labelencoder.classes_
```

Out[16]:
```
array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
      dtype=object)
```

Implementing OneHotEncoder to separate category variables into dummy variables.

In [17]:
```
from sklearn.preprocessing import OneHotEncoder
X_onehotencoder = OneHotEncoder (categorical_features = [8])
X = X_onehotencoder.fit_transform(housing_df)
```

In [18]:
```
x=X.toarray()
x[:,8:11]
```

Out[18]:
```
array([[ 880.,  129.,  322.],
       [7099., 1106., 2401.],
       [1467.,  190.,  496.],
       ...,
       [2254.,  485., 1007.],
       [1860.,  409.,  741.],
       [2785.,  616., 1387.]])
```

In [19]:
```
columns=housing_df.columns.tolist()

new_columns=[('proximity_'+i) for i in x_labelencoder.classes_]
new_columns[5:]=columns[0:-2]
new_columns.append(columns[-1])
new_columns
```

Out[19]:
```
['proximity_<1H OCEAN',
 'proximity_INLAND',
 'proximity_ISLAND',
 'proximity_NEAR BAY',
 'proximity_NEAR OCEAN',
 'longitude',
 'latitude',
 'housing_median_age',
 'total_rooms',
 'total_bedrooms',
 'population',
 'households',
 'median_income',
 'median_house_value']
```

In [20]:
```python
housing_df_new=pd.DataFrame(x,index=housing_df.index,columns=new_columns)

housing_df_new.head()
```

Out[20]:
```
   proximity_<1H OCEAN  proximity_INLAND  proximity_ISLAND  \
0                  0.0               0.0               0.0
1                  0.0               0.0               0.0
2                  0.0               0.0               0.0
3                  0.0               0.0               0.0
4                  0.0               0.0               0.0

   proximity_NEAR BAY  proximity_NEAR OCEAN  longitude  latitude  \
0                 1.0                   0.0    -122.23     37.88
1                 1.0                   0.0    -122.22     37.86
2                 1.0                   0.0    -122.24     37.85
3                 1.0                   0.0    -122.25     37.85
4                 1.0                   0.0    -122.25     37.85

   housing_median_age  total_rooms  total_bedrooms  population  households  \
0                41.0        880.0           129.0       322.0       126.0
1                21.0       7099.0          1106.0      2401.0      1138.0
2                52.0       1467.0           190.0       496.0       177.0
3                52.0       1274.0           235.0       558.0       219.0
4                52.0       1627.0           280.0       565.0       259.0

   median_income  median_house_value
0         8.3252            452600.0
1         8.3014            358500.0
2         7.2574            352100.0
3         5.6431            341300.0
4         3.8462            342200.0
```

Feature Engineering

Instead of having longitude and latitude as separate attribute we will put an attribute distance_from_california california is at Latitude 36.778259 Longitude -119.41793 we use the 'haversine' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface – giving an 'as-the-crow-flies' distance between the points (ignoring any hills they fly over, of course!).

In [21]:
```
from math import radians, cos, sin, asin, sqrt

def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c * r
```

In [22]:
```
housing_df_new['distance_from_california']=[haversine(-119.41793,36.778259 ,housing_df_
housing_df_new.head()
```

Out[22]:
```
   proximity_<1H OCEAN  proximity_INLAND  proximity_ISLAND  \
0                  0.0               0.0               0.0
1                  0.0               0.0               0.0
2                  0.0               0.0               0.0
3                  0.0               0.0               0.0
4                  0.0               0.0               0.0

   proximity_NEAR BAY  proximity_NEAR OCEAN  longitude  latitude  \
0                 1.0                   0.0    -122.23     37.88
1                 1.0                   0.0    -122.22     37.86
2                 1.0                   0.0    -122.24     37.85
3                 1.0                   0.0    -122.25     37.85
4                 1.0                   0.0    -122.25     37.85

   housing_median_age  total_rooms  total_bedrooms  population  households  \
0                41.0        880.0           129.0       322.0       126.0
1                21.0       7099.0          1106.0      2401.0      1138.0
2                52.0       1467.0           190.0       496.0       177.0
3                52.0       1274.0           235.0       558.0       219.0
4                52.0       1627.0           280.0       565.0       259.0
```

```
   median_income  median_house_value  distance_from_california
0         8.3252            452600.0                277.163423
1         8.3014            358500.0                275.422121
2         7.2574            352100.0                276.548120
3         5.6431            341300.0                277.346295
4         3.8462            342200.0                277.346295
```

##
NOW DELETE LONGITUDE AND LATITUDE COLUMN

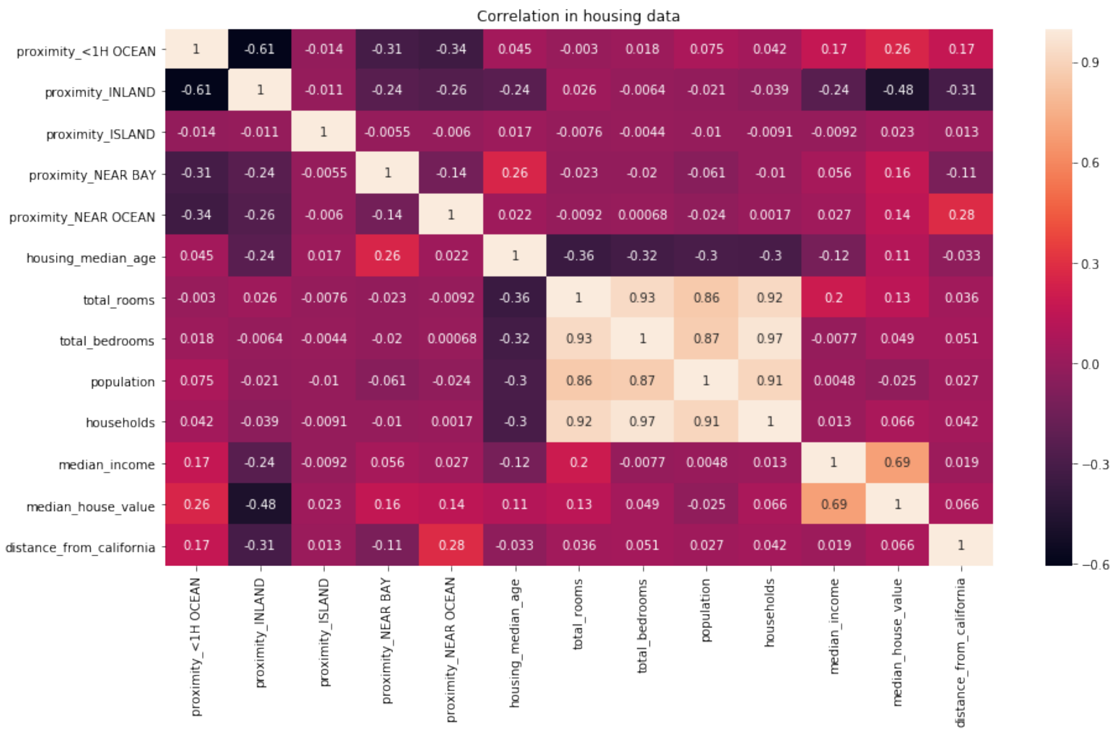
In [23]:
```
housing_df_new=housing_df_new.drop(columns=['longitude','latitude'])
housing_df_new.head()
```

Out[23]:
```
   proximity_<1H OCEAN  proximity_INLAND  proximity_ISLAND  \
0                  0.0               0.0               0.0
1                  0.0               0.0               0.0
2                  0.0               0.0               0.0
3                  0.0               0.0               0.0
4                  0.0               0.0               0.0

   proximity_NEAR BAY  proximity_NEAR OCEAN  housing_median_age  total_rooms  \
0                 1.0                   0.0                41.0        880.0
1                 1.0                   0.0                21.0       7099.0
2                 1.0                   0.0                52.0       1467.0
3                 1.0                   0.0                52.0       1274.0
4                 1.0                   0.0                52.0       1627.0

   total_bedrooms  population  households  median_income  median_house_value  \
0           129.0       322.0       126.0         8.3252            452600.0
1          1106.0      2401.0      1138.0         8.3014            358500.0
2           190.0       496.0       177.0         7.2574            352100.0
3           235.0       558.0       219.0         5.6431            341300.0
4           280.0       565.0       259.0         3.8462            342200.0

   distance_from_california
0                277.163423
1                275.422121
2                276.548120
3                277.346295
4                277.346295
```

In [24]:
```
import seaborn as sns

plt.figure(figsize=(15,8))


corr = housing_df_new.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
```

```
                    yticklabels=corr.columns.values, annot=True)
plt.title('Correlation in housing data')
plt.plot()
```

Out[24]: []



Standerdise our data

In [25]:
```
x_feature=housing_df_new.drop(columns='median_house_value')
y_target=housing_df_new['median_house_value']
```

In [26]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (x_feature, y_target, test_size = 1
                                                      random_state = 0)
from sklearn.preprocessing import StandardScaler
independent_scalar = StandardScaler()
x_train = pd.DataFrame(independent_scalar.fit_transform (x_train),index=x_train.index,c
x_test = pd.DataFrame(independent_scalar.transform (x_test),index=x_test.index,columns=
y_train=pd.DataFrame(y_train)
y_test=pd.DataFrame(y_test)
```

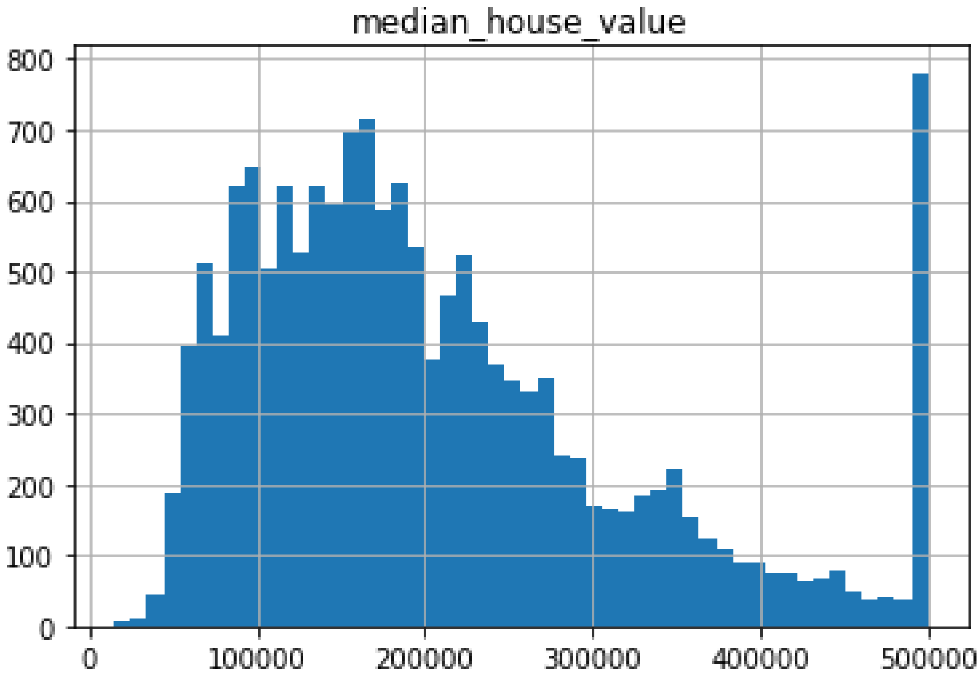
In [27]:
```
x_train.head()
```

Out[27]:
```
       proximity_<1H OCEAN  proximity_INLAND  proximity_ISLAND  \
19226             1.121845         -0.679323         -0.013923
```

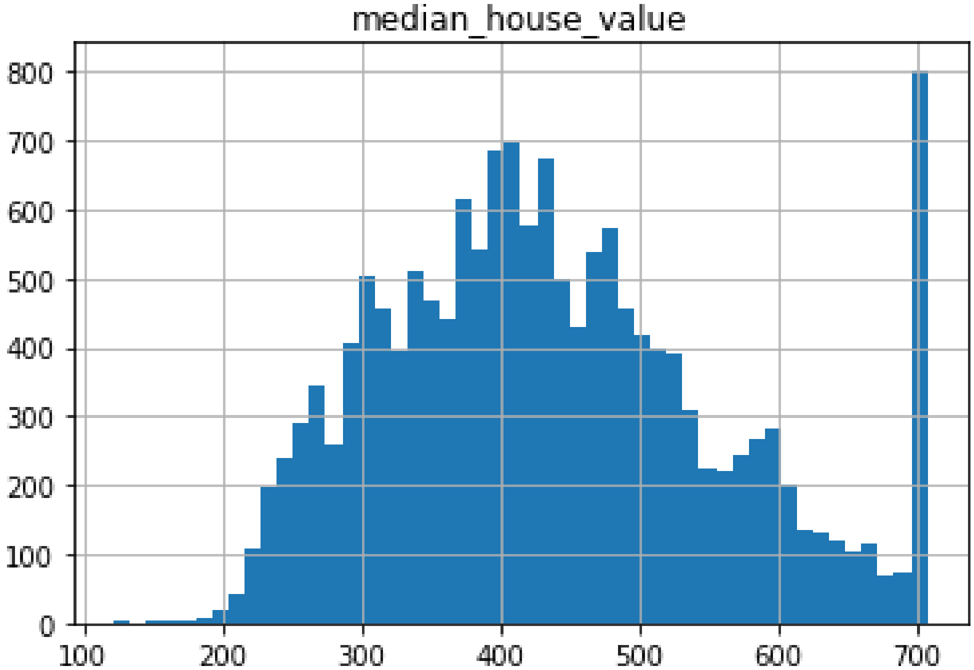
```
14549          -0.891389        -0.679323        -0.013923
9093           -0.891389         1.472053        -0.013923
12213           1.121845        -0.679323        -0.013923
12765          -0.891389         1.472053        -0.013923

       proximity_NEAR BAY  proximity_NEAR OCEAN  housing_median_age  \
19226           -0.353669             -0.386732           -0.764262
14549           -0.353669              2.585768           -0.843631
9093            -0.353669             -0.386732           -0.764262
12213           -0.353669             -0.386732           -1.240475
12765           -0.353669             -0.386732           -0.605525

       total_rooms  total_bedrooms  population  households  median_income  \
19226     1.068091        0.412186    0.436631    0.327101       1.808122
14549    -0.480400       -0.641939   -0.768275   -0.670119       1.097891
9093     -0.955697       -0.972692   -0.971859   -1.027760      -0.349490
12213    -1.084700       -1.179710   -1.141367   -1.194834       1.645924
12765     0.283095        0.535921    0.269744    0.616870      -0.717009

       distance_from_california
19226                  0.343862
14549                  1.599686
9093                  -0.521462
12213                  1.072802
12765                 -0.380512
```

In [28]:
```
y_train.hist(bins=50)
```

Out[28]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f530b6e1e10>]],
      dtype=object)
```

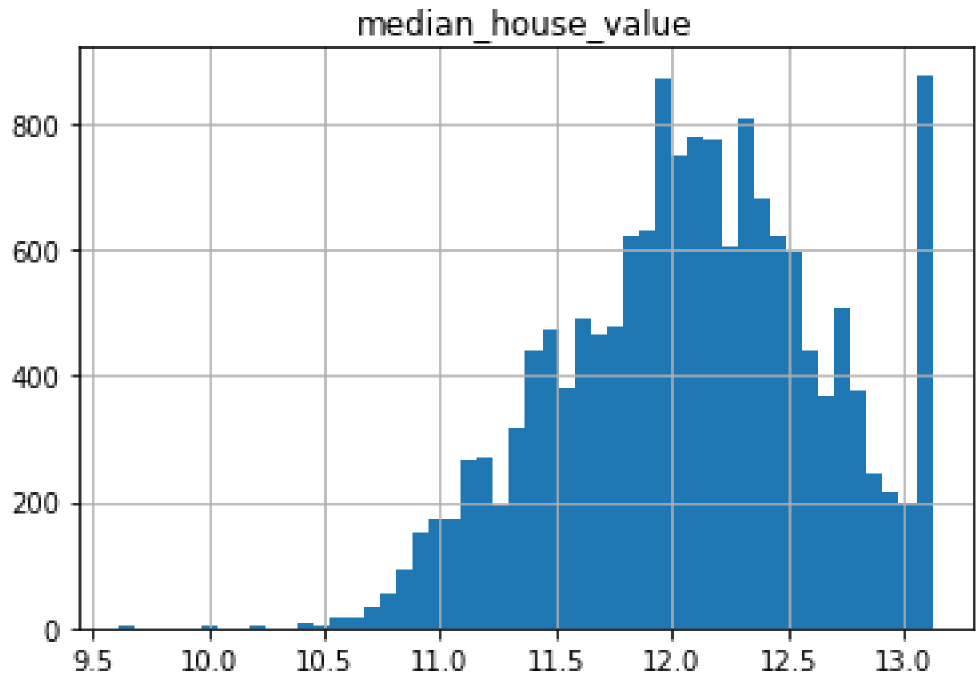



```
In [29]: np.sqrt(y_train).hist(bins=50)

Out[29]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f5308ab2f60>]],
          dtype=object)
```



In [30]: np.log(y_train).hist(bins=50)

Out[30]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f530899e0f0>]],
      dtype=object)



from above histograms we can say square root of y_train gives distribution close to normal distribution. hence we will take square root of y_train and y_test

```
In [31]: y_train=np.sqrt(y_train)
         y_test=np.sqrt(y_test)
```

```
In [32]: y_train.head()
```

```
Out[32]:        median_house_value
         19226          617.494939
         14549          573.846669
         9093           430.581003
         12213          707.107488
         12765          312.249900
```

```
In [33]: y_test.head()
```

```
Out[33]:        median_house_value
         14740          370.000000
```

```
10101          491.222964
20566          447.995536
2670           269.258240
15709          678.232998
```

Predictive Modeling
Linear Regression

```
In [34]: from sklearn.linear_model import LinearRegression

         linear_regressoragent = LinearRegression()
         linear_regressoragent.fit(x_train,y_train)

Out[34]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [35]: #predict the X_test
         predictValues = linear_regressoragent.predict(x_test)

In [36]: from sklearn.metrics import mean_squared_error
         np.sqrt(mean_squared_error(y_test, predictValues))

Out[36]: 72.33141914038299

In [37]: linear_regressoragent.score(x_test,y_test)

Out[37]: 0.6512403943232119
```

Decision tree Regression

```
In [38]: from sklearn.tree import DecisionTreeRegressor
         model_dtregress=DecisionTreeRegressor(max_depth=8,random_state=10,criterion='mse',preso
         model_dtregress.fit(x_train,y_train)

         predict_dtregress=model_dtregress.predict(x_test)
         np.sqrt(mean_squared_error(y_test,predict_dtregress))

Out[38]: 67.19112486979536

In [39]: model_dtregress.score(x_test,y_test)

Out[39]: 0.6990488345070018
```

Random forest Regression

```
In [40]: from sklearn.ensemble import RandomForestRegressor
         model_rfregressor=RandomForestRegressor(n_estimators=15,criterion='mse',min_samples_spl
         model_rfregressor.fit(x_train,np.array(y_train).ravel())
         predict_rfregressor=model_rfregressor.predict(x_test)
         np.sqrt(mean_squared_error(y_test,predict_rfregressor))

Out[40]: 59.624760503580426
```

In [41]:
```
model_rfregressor.score(x_test,y_test)
```

Out[41]:
```
0.7630124664153124
```

At this point we can drop households and total_rooms columns while creating model; since other columns are present which can convey similar information

In [42]:
```
x_train=x_train.drop(columns=['households','total_rooms'])
x_test=x_test.drop(columns=['households','total_rooms'])
x_test.head()
```

Out[42]:
```
       proximity_<1H OCEAN  proximity_INLAND  proximity_ISLAND  \
14740            -0.891389         -0.679323         -0.013923   
10101             1.121845         -0.679323         -0.013923   
20566            -0.891389          1.472053         -0.013923   
2670             -0.891389          1.472053         -0.013923   
15709            -0.891389         -0.679323         -0.013923   

       proximity_NEAR BAY  proximity_NEAR OCEAN  housing_median_age  \
14740           -0.353669              2.585768           -0.526156   
10101           -0.353669             -0.386732            0.267531   
20566           -0.353669             -0.386732            0.029425   
2670            -0.353669             -0.386732            0.664375   
15709            2.827503             -0.386732           -0.288050   

       total_bedrooms  population  median_income  distance_from_california  
14740       -0.330222    0.108974       0.146289                  2.042879  
10101       -0.332602   -0.113834       1.005470                  0.354755  
20566        0.024325    0.111595       0.250216                 -0.100769  
2670        -0.834680   -0.905454      -0.751370                  2.179562  
15709       -0.342120   -0.679152       0.596570                 -0.185641  
```

In [43]:
```
model_rfregressor1=RandomForestRegressor(n_estimators=15,criterion='mse',min_samples_sp
model_rfregressor1.fit(x_train,np.array(y_train).ravel())
predict_rfregressor1=model_rfregressor1.predict(x_test)
np.sqrt(mean_squared_error(y_test,predict_rfregressor1))
```

Out[43]:
```
59.98290152069001
```

In [44]:
```
model_rfregressor1.score(x_test,y_test)
```

Out[44]:
```
0.7601569459974005
```

Bonus Exercise:

In [45]:
```
x_train=x_train.median_income
x_test=x_test.median_income
```

In [46]:
```
x_train=np.array(x_train).reshape(-1,1)
x_test=np.array(x_test).reshape(-1,1)
y_train=np.array(y_train).reshape(-1,1)
y_test=np.array(y_test).reshape(-1,1)
```

```
In [47]: model_linear_regression=LinearRegression()
         model_linear_regression.fit(x_train,y_train)
```

```
Out[47]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [48]: predicted_test=model_linear_regression.predict(x_test)
         predicted_train=model_linear_regression.predict(x_train)
```

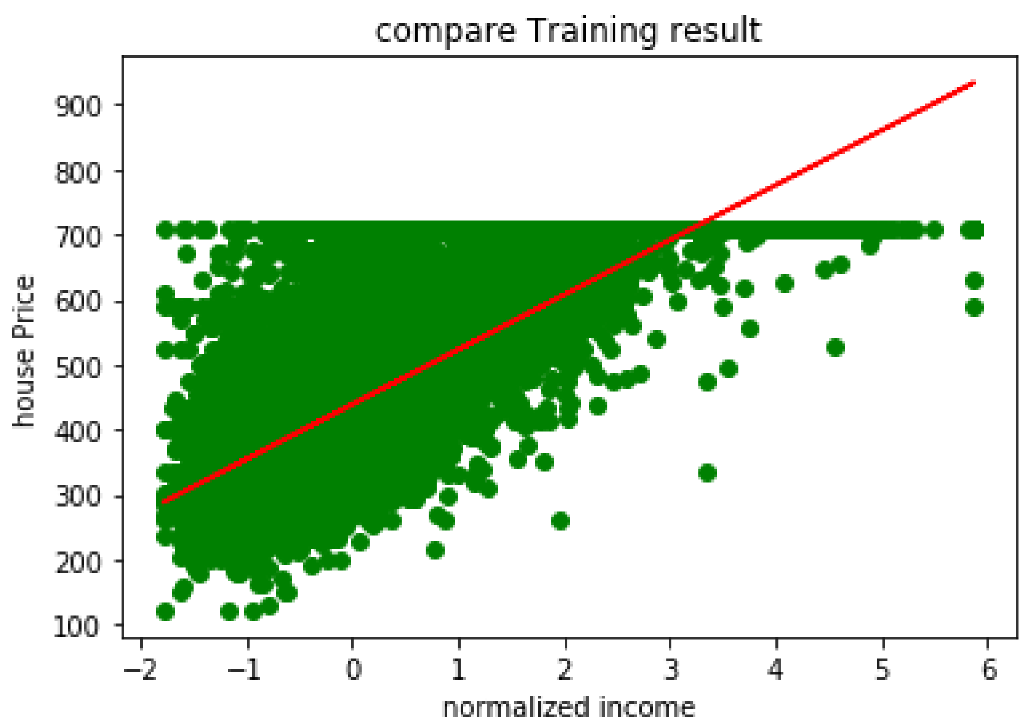
```
In [49]: model_linear_regression.score(x_test,y_test)
```

```
Out[49]: 0.44302338196197383
```

```
In [50]: np.sqrt(mean_squared_error(y_test,predicted_test))
```

```
Out[50]: 91.40761530242808
```

```
In [51]: plt.scatter(x_train, y_train, color = 'green')
         plt.plot (x_train, predicted_train, color = 'red')
         plt.title ('compare Training result')
         plt.xlabel('normalized income')
         plt.ylabel('house Price')
         plt.show()
```




```
In [52]: plt.scatter(x_test, y_test, color = 'green')
         plt.plot (x_test, predicted_test, color = 'red')
```

```
plt.title ('compare Test result')
plt.xlabel('normalized income')
plt.ylabel('house Price')
plt.show()
```