HousingProj

September 12, 2021

[1]:
```
#%pylab inline
import pandas as pd
import numpy as np

df = pd.read_csv('Housing Project/housingproj.csv')
#print(df.head(5))
print(df.shape)




#FILL the missing Values
df.isnull().sum()
```

```
(20640, 10)
```

[1]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

Data Dictionary:

Features:'longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity'

Label: median_house_value

There are 20640 rows and 10 features. Since the resonse variable is continuous, it can be considered as linear regression problem.

[2]:
```
#df.describe().transpose()
#impute missing values with mean
#df['total_bedrooms'].head()
df.total_bedrooms.fillna(df.total_bedrooms.mean(),inplace=True)
df.total_bedrooms.head()
```

[2]:
```
0     129.0
1    1106.0
2     190.0
3     235.0
4     280.0
Name: total_bedrooms, dtype: float64
```

[3]:
```
#encoding the categorical variable
from sklearn.preprocessing import LabelEncoder

# Scikit Learn
lb_temp = LabelEncoder()

df['ocean_Dist'] = lb_temp.fit_transform(df.ocean_proximity)

#print(df['ocean_Dist'])
# Coding the labels. Converting the categories into the numbers.
print(df.head(10))
df1=df.drop(columns=['ocean_proximity'])
print(df1.head())
```

```
   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
5    -122.25     37.85                  52          919           213.0
6    -122.25     37.84                  52         2535           489.0
7    -122.25     37.84                  52         3104           687.0
8    -122.26     37.84                  42         2555           665.0
9    -122.25     37.84                  52         3549           707.0

   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
5         413         193         4.0368        NEAR BAY              269700
6        1094         514         3.6591        NEAR BAY              299200
```

```
7        1157          647         3.1200        NEAR BAY              241400
8        1206          595         2.0804        NEAR BAY              226700
9        1551          714         3.6912        NEAR BAY              261100

   ocean_Dist
0           3
1           3
2           3
3           3
4           3
5           3
6           3
7           3
8           3
9           3
   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  median_house_value  ocean_Dist
0         322         126         8.3252              452600           3
1        2401        1138         8.3014              358500           3
2         496         177         7.2574              352100           3
3         558         219         5.6431              341300           3
4         565         259         3.8462              342200           3
```

[4]:
```
#create dummy for Ocean_Dist

#dummy = pd.get_dummies(df1['ocean_Dist'],prefix='ocean_dist',drop_first=True)

#display(dummy)

df1 = df1.join(pd.
 ↪get_dummies(df1['ocean_Dist'],prefix='ocean_dist',drop_first=True))
display(df1)
df1.shape
```

```
       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
...          ...       ...                 ...          ...             ...
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  median_house_value  ocean_Dist  \
0             322         126         8.3252              452600           3
1            2401        1138         8.3014              358500           3
2             496         177         7.2574              352100           3
3             558         219         5.6431              341300           3
4             565         259         3.8462              342200           3
...           ...         ...            ...                 ...         ...
20635         845         330         1.5603               78100           1
20636         356         114         2.5568               77100           1
20637        1007         433         1.7000               92300           1
20638         741         349         1.8672               84700           1
20639        1387         530         2.3886               89400           1

       ocean_dist_1  ocean_dist_2  ocean_dist_3  ocean_dist_4
0                 0             0             1             0
1                 0             0             1             0
2                 0             0             1             0
3                 0             0             1             0
4                 0             0             1             0
...             ...           ...           ...           ...
20635             1             0             0             0
20636             1             0             0             0
20637             1             0             0             0
20638             1             0             0             0
20639             1             0             0             0

[20640 rows x 14 columns]
```

[4]:
```
(20640, 14)
```

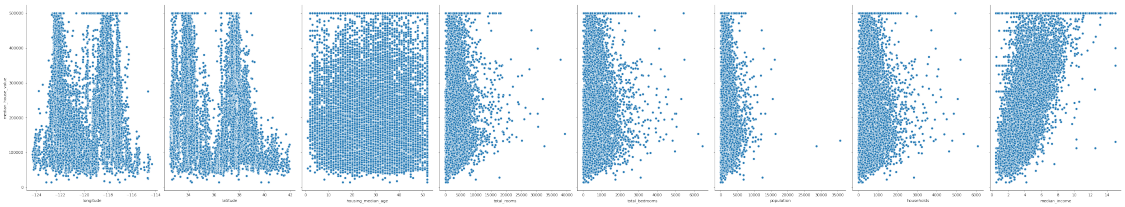
[5]:
```
#splitting to X and Y where Y is the lable or dependent variable
X = df1.drop(columns = ['median_house_value',])
Y = df1[['median_house_value']]
print(Y.head())
print(X.head())
list(X.columns)
list(Y.columns)
```

```
   median_house_value
0              452600
1              358500
2              352100
```

```
3              341300
4              342200
   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_Dist  ocean_dist_1  \
0         322         126         8.3252           3             0
1        2401        1138         8.3014           3             0
2         496         177         7.2574           3             0
3         558         219         5.6431           3             0
4         565         259         3.8462           3             0

   ocean_dist_2  ocean_dist_3  ocean_dist_4
0             0             1             0
1             0             1             0
2             0             1             0
3             0             1             0
4             0             1             0
```

[5]:
```
['median_house_value']
```

[6]:
```
#Analysis of variable against the dependent variable
import seaborn as sns

#allow plot to appear inline
%matplotlib inline

sns.pairplot(df1, x_vars =␣
 ↪['longitude','latitude','housing_median_age','total_rooms','total_bedrooms',
                           'population','households','median_income'],
             y_vars='median_house_value', height=7,aspect =0.7, kind='scatter');
```



1 Above result indicates that except for median income all the other variables has no linear corelation with dependent variable

[7]:
```
#Model Building splitting X and Y with test size of 30%
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train,X_test, Y_train,Y_test = train_test_split(X,Y, test_size=0.3,␣
 ↪random_state=42)
```

[9]:
```
# split is 70% for training and 30% for testing
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(14448, 13)
(14448, 1)
(6192, 13)
(6192, 1)
```

[5]:
```

```

	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	
5	-122.25	37.85	52	919	213.0	
6	-122.25	37.84	52	2535	489.0	
7	-122.25	37.84	52	3104	687.0	
8	-122.26	37.84	42	2555	665.0	
9	-122.25	37.84	52	3549	707.0	

	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	
5	413	193	4.0368	NEAR BAY	269700	
6	1094	514	3.6591	NEAR BAY	299200	
7	1157	647	3.1200	NEAR BAY	241400	
8	1206	595	2.0804	NEAR BAY	226700	
9	1551	714	3.6912	NEAR BAY	261100	

```
   ocean_Dist
0           3
1           3
2           3
3           3
4           3
5           3
6           3
7           3
8           3
9           3
   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  median_house_value  ocean_Dist
0         322         126         8.3252              452600           3
1        2401        1138         8.3014              358500           3
2         496         177         7.2574              352100           3
3         558         219         5.6431              341300           3
4         565         259         3.8462              342200           3
```

[10]:
```
#standardising of the variable to bring all the independent variables with in␣
 ↪standard range

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()



# i took a longer approach to standardise each and every variable instead of␣
 ↪entire dataset just for practice
#standardise formula (x- mean(x))/sd(x)

X_train.total_bedrooms = scaler.fit_transform(X_train[['total_bedrooms']])
X_train.total_rooms = scaler.fit_transform(X_train[['total_rooms']])
X_train.population = scaler.fit_transform(X_train[['population']])
X_train.households = scaler.fit_transform(X_train[['households']])
X_train.longitude = scaler.fit_transform(X_train[['longitude']])
X_train.latitude = scaler.fit_transform(X_train[['latitude']])
X_train.housing_median_age=  scaler.
 ↪fit_transform(X_train[['housing_median_age']])
X_train.median_income=  scaler.fit_transform(X_train[['median_income']])
```

```
X_test.total_bedrooms = scaler.fit_transform(X_test[['total_bedrooms']])
X_test.total_rooms = scaler.fit_transform(X_test[['total_rooms']]) # x is␣
↪column then (x- mean(x))/sd(x)
#display (X_train[:5])
X_test.population = scaler.fit_transform(X_test[['population']])
X_test.households = scaler.fit_transform(X_test[['households']])
X_test.longitude = scaler.fit_transform(X_test[['longitude']])
X_test.latitude = scaler.fit_transform(X_test[['latitude']])
X_test.housing_median_age=  scaler.fit_transform(X_test[['housing_median_age']])
X_test.median_income=  scaler.fit_transform(X_test[['median_income']])
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
7061    0.780934 -0.805682            0.509357    -0.113242       -0.337870
```

```
14689   1.245270 -1.339473           -0.679873    -0.213566       -0.013884
17323  -0.277552 -0.496645           -0.362745    -0.482639       -0.614210
10056  -0.706938  1.690024           -1.155565    -0.848339       -0.926284
15750  -1.430902  0.992350            1.857152     0.251071        0.400626

       population  households  median_income  ocean_Dist  ocean_dist_1  \
7061    -0.184117   -0.243508       0.133506           0             0
14689   -0.376191   -0.013267      -0.532218           4             0
17323   -0.611240   -0.565322       0.170990           4             0
10056   -0.987495   -0.949929      -0.402916           1             1
15750    0.086015    0.426285      -0.299285           3             0

       ocean_dist_2  ocean_dist_3  ocean_dist_4
7061              0             0             0
14689             0             0             1
17323             0             0             1
10056             0             0             0
15750             0             1             0


       median_house_value
7061               193800
14689              169700
17323              259800
10056              136100
15750              500001
```

[11]:
```
display (X_train[:5])
display (Y_train[:5])
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
7061    0.780934 -0.805682            0.509357    -0.113242       -0.337870
14689   1.245270 -1.339473           -0.679873    -0.213566       -0.013884
17323  -0.277552 -0.496645           -0.362745    -0.482639       -0.614210
10056  -0.706938  1.690024           -1.155565    -0.848339       -0.926284
15750  -1.430902  0.992350            1.857152     0.251071        0.400626

       population  households  median_income  ocean_Dist  ocean_dist_1  \
7061    -0.184117   -0.243508       0.133506           0             0
14689   -0.376191   -0.013267      -0.532218           4             0
17323   -0.611240   -0.565322       0.170990           4             0
10056   -0.987495   -0.949929      -0.402916           1             1
15750    0.086015    0.426285      -0.299285           3             0

       ocean_dist_2  ocean_dist_3  ocean_dist_4
7061              0             0             0
14689             0             0             1
```

```
17323             0             0             1
10056             0             0             0
15750             0             1             0


       median_house_value
7061               193800
14689              169700
17323              259800
10056              136100
15750              500001
```

[11]:

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
7061    0.780934 -0.805682            0.509357    -0.113242       -0.337870
14689   1.245270 -1.339473           -0.679873    -0.213566       -0.013884
17323  -0.277552 -0.496645           -0.362745    -0.482639       -0.614210
10056  -0.706938  1.690024           -1.155565    -0.848339       -0.926284
15750  -1.430902  0.992350            1.857152     0.251071        0.400626

       population  households  median_income  ocean_Dist  ocean_dist_1  \
7061    -0.184117   -0.243508       0.133506           0             0
14689   -0.376191   -0.013267      -0.532218           4             0
17323   -0.611240   -0.565322       0.170990           4             0
10056   -0.987495   -0.949929      -0.402916           1             1
15750    0.086015    0.426285      -0.299285           3             0

       ocean_dist_2  ocean_dist_3  ocean_dist_4
7061              0             0             0
14689             0             0             1
17323             0             0             1
10056             0             0             0
15750             0             1             0


       median_house_value
7061           -0.113387
14689          -0.321603
17323           0.456831
10056          -0.611895
15750           2.532087
```

[12]:

```
display (X_test[:5])
display (Y_test[:5])
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
```

```
20046   0.262460  0.221006          -0.302678    -0.498771        0.010928
3024    0.037969 -0.209747           0.096730     0.147762        0.010928
15663  -1.448662  1.035689           1.854125     0.546562        0.010928
20484   0.407132 -0.612406          -0.941731     0.196319        0.010928
9814   -1.194239  0.483203           0.416256    -0.118405        0.010928

       population  households  median_income  ocean_Dist  ocean_dist_1  \
20046   -0.024915   -0.358030      -1.152099           1             1
3024     0.130364    0.230115      -0.701791           1             1
15663   -0.098516    1.220811      -0.199201           3             0
20484    0.256024   -0.002529       0.996622           0             0
9814    -0.320216   -0.177665      -0.069475           4             0

       ocean_dist_2  ocean_dist_3  ocean_dist_4
20046             0             0             0
3024              0             0             0
15663             0             1             0
20484             0             0             0
9814              0             0             1


       median_house_value
20046               47700
3024                45800
15663              500001
20484              218600
9814               278000
```

[13]:
```
# Linear Regression equation
from sklearn.linear_model import LinearRegression

# instantiate
linreg = LinearRegression()

# fit the model to the training data
linreg.fit(X_train, Y_train)

#print the intercept and the coeff
print (linreg.intercept_)
print (linreg.coef_)
```

```
[220287.15751609]
[[-53080.22484228 -53211.37198001  13949.87818887 -12701.71831577
   44378.88674127 -42559.67778653  16172.44726364  74802.55919045
    7264.35303288 -48340.41233654 121359.83972498 -27859.10578729
  -25884.39917967]]
```

[16]:
```
# predictions on the testing data
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
Y_pred = linreg.predict(X_test)

print ("Rsqd = ", r2_score(Y_pred, Y_test))
```

```
Rsqd =  0.4569503802211192
```

[18]:
```
#RMSE value
from sklearn.metrics import mean_squared_error
import numpy as np
print("RMSE = ", np.sqrt(mean_squared_error(Y_pred, Y_test)))
```

```
RMSE =  68795.0507977502
```

[19]:
```
#Display of Predictions or house Values on test population
print(Y_pred)
```

```
[[ 65207.76105097]
 [133991.57855677]
 [264449.05897032]
 …
 [282790.17340566]
 [112031.37450103]
 [217910.31945832]]
```

[ ]:
```

```