LinearRegressionForPricePredictionHousingCalifornia

April 4, 2020

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[3]:
```
housing_data = pd.read_csv('datasets/housing.csv')
#The difference compared to head method is that sample is random
housing_data.sample(5)
```

[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
1007	-121.75	37.68	35.0	1755.0	299.0	702.0	263.0	5.2443	183400.0	INLAND
17031	-122.25	37.50	44.0	348.0	79.0	154.0	73.0	4.7708	253800.0	NEAR OCEAN
1884	-119.99	38.93	23.0	1882.0	414.0	673.0	277.0	2.9091	141900.0	INLAND
5649	-118.29	33.74	41.0	1382.0	361.0	905.0	344.0	2.7500	238300.0	NEAR OCEAN
5090	-118.28	33.97	31.0	1068.0	271.0	1091.0	281.0	1.6890	102600.0	<1H OCEAN

[4]:
```
#drop data with missing field
housing_data = housing_data.dropna()
```

[5]:
```
#number of (entry,variable) after dropping empty entries
housing_data.shape
```

[5]:
```
(20433, 10)
```

[8]:
```
#this loc method is to count the number of entry that have certain value
#as you can see, there are 958 entries with median_house_value = 500001,
housing_data.loc[housing_data['median_house_value'] ==500001].count()
```

[8]:
```
longitude             958
latitude              958
housing_median_age    958
total_rooms           958
total_bedrooms        958
population            958
households            958
median_income         958
median_house_value    958
ocean_proximity       958
dtype: int64
```

[9]:
```
#This upper value of house value 500001 will scatter our data model, thus, we␣
 ↪need to remove using drop method
housing_data = housing_data.drop(housing_data.
 ↪loc[housing_data['median_house_value'] == 500001].index)
```

[10]:
```
housing_data.shape
```

[10]:
```
(19475, 10)
```

[11]:
```
housing_data.head()
```

[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41.0	880.0	129.0	
1	-122.22	37.86	21.0	7099.0	1106.0	
2	-122.24	37.85	52.0	1467.0	190.0	
3	-122.25	37.85	52.0	1274.0	235.0	
4	-122.25	37.85	52.0	1627.0	280.0	

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

[12]:
```
#ocean_proximity have discrete value, we need to convert
housing_data['ocean_proximity'].unique()
```

[12]:
```
array(['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND'],
      dtype=object)
```

[15]:
```
#using one hot encoding to numeric from using get_dumm from panda
housing_data = pd.get_dummies(housing_data, columns=['ocean_proximity'])
```

[17]:
```
#now it have 14 variable. 10-1+5 . 5 is 5 kind of value in ocean proximity
housing_data.shape
```

[17]:
```
(19475, 14)
```

[19]:
```
#discrete variable of ocean_proximity
housing_data.sample(5)
```

[19]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
4891	-118.25	34.02	33.0	1676.0	525.0	
1251	-122.17	39.31	35.0	2791.0	552.0	
12200	-117.21	33.61	7.0	7722.0	1324.0	
16148	-122.47	37.78	52.0	2042.0	378.0	
17370	-120.43	34.98	21.0	2725.0	514.0	

	population	households	median_income	median_house_value	\
4891	2564.0	515.0	2.1957	100800.0	
1251	1395.0	476.0	2.5625	62700.0	
12200	2975.0	1161.0	3.6273	150900.0	
16148	1153.0	408.0	4.1856	404700.0	
17370	1466.0	488.0	3.6639	128600.0	

	ocean_proximity_<1H OCEAN	ocean_proximity_INLAND	\
4891	1	0	
1251	0	1	
12200	1	0	
16148	0	0	
17370	1	0	

	ocean_proximity_ISLAND	ocean_proximity_NEAR BAY	\
4891	0	0	
1251	0	0	
12200	0	0	
16148	0	1	
17370	0	0	

	ocean_proximity_NEAR OCEAN
4891	0
1251	0
12200	0
16148	0
17370	0

[20]:
```
#now start to set feature and target in linear regression
#X is X axis, but dropping median_house_value because this is our target
#Y axis is median house value only
X = housing_data.drop('median_house_value', axis=1)
Y = housing_data['median_house_value']
```

[22]:
```
X.columns #without median_house_income
```

[22]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity_<1H OCEAN', 'ocean_proximity_INLAND',
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
       'ocean_proximity_NEAR OCEAN'],
      dtype='object')
```

[43]:
```
#split dataset into test and train datasets. Train is to train the model, test␣
 ↪is to test the model, its also shuffling
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2) #test␣
 ↪is 20%, train is 80%
```

[44]:
```
x_train.shape, x_test.shape
```

[44]:
```
((15580, 13), (3895, 13))
```

[45]:
```
y_train.shape, y_test.shape
```

[45]:
```
((15580,), (3895,))
```

[46]:
```
#start the linear regression
from sklearn.linear_model import LinearRegression
#normalization scale all numerics in 0-1, greatly improve data processing speed.
 ↪ (preprocessing)
linear_model = LinearRegression(normalize=True).fit(x_train,y_train)
```

[47]:
```
#For regression, this is R^2
print("Training_score : " , linear_model.score(x_train, y_train))
```

```
Training_score :  0.6108857235416872
```

[48]:
```
predictors = x_train.columns
predictors #is list of coloumn of features
```

[48]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity_<1H OCEAN', 'ocean_proximity_INLAND',
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
```

```
       'ocean_proximity_NEAR OCEAN'],
      dtype='object')
```

[49]:
```
#calculate coefficient of all features using Panda Series
#as you can see, more + coef means that it contribute much to␣
 ↪median_house_price (Y axis)
coef = pd.Series(linear_model.coef_,predictors).sort_values()
print(coef)
```

```
ocean_proximity_INLAND       -25893.156018
longitude                    -25244.535316
latitude                     -23415.655937
population                      -33.061262
total_rooms                      -6.200343
households                       46.969096
total_bedrooms                   88.631976
housing_median_age              895.527336
ocean_proximity_NEAR BAY       6138.336354
ocean_proximity_<1H OCEAN     13223.281604
ocean_proximity_NEAR OCEAN    16683.737622
median_income                 37937.668447
ocean_proximity_ISLAND       205829.067605
dtype: float64
```

[50]:
```
#to predict the value (median_house_value) from x_test
y_pred = linear_model.predict(x_test)
```

[51]:
```
#dataframe that compare predicted value with actual value
df_pred_actual = pd.DataFrame({'predicted' : y_pred, 'actual' : y_test})
df_pred_actual.sample(10)
```

[51]:

	predicted	actual
12488	119136.251558	93800.0
15360	-541218.660948	134400.0
3634	197989.069984	176500.0
13256	252528.838365	239000.0
13195	137058.538164	139300.0
5541	289492.970134	411200.0
13418	147111.902534	207700.0
3081	78235.489272	92300.0
17278	293048.564099	340400.0
15953	238741.862648	260000.0

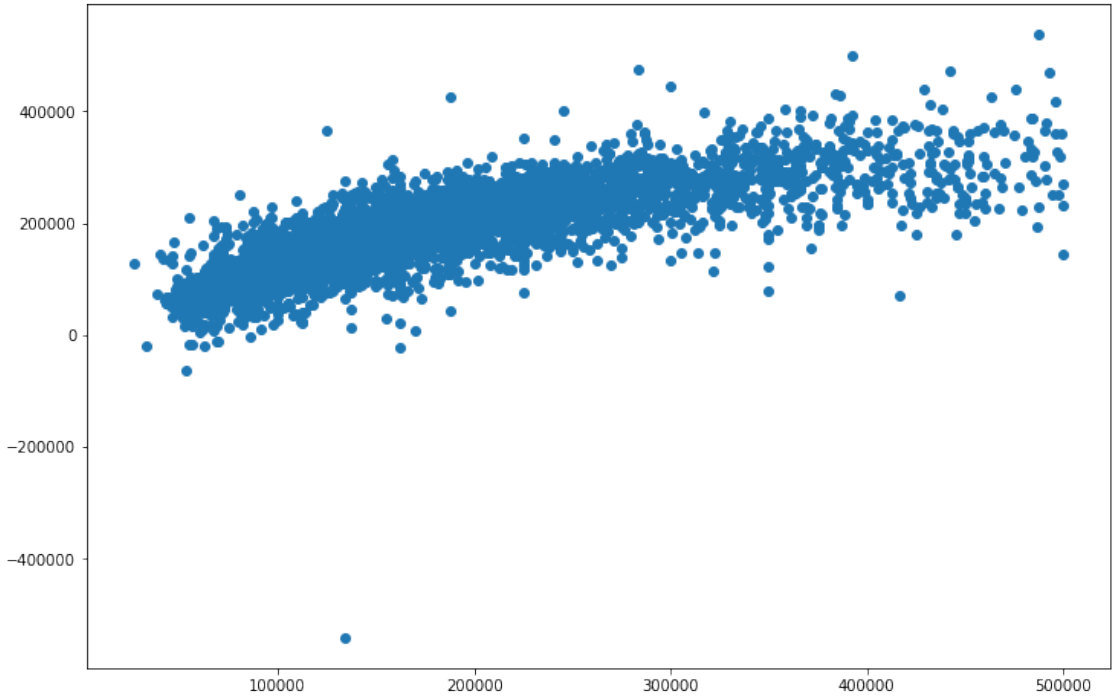
[54]:
```
#evaluate a model , calculate R2 score for test data
from sklearn.metrics import r2_score
```

```
print("Testing Score: ", r2_score(y_test, y_pred))#first input is actual value,␣
 ↪2nd input is predicted value
```

```
Testing Score:  0.619933485613736
```

[56]:
```
#use scatter plot to visualize predicted vs actual in test series

fig, ax = plt.subplots(figsize=(12,8))
plt.scatter(y_test,y_pred)
plt.show()
```



[57]:
```
#sample 100
df_pred_actual_sample = df_pred_actual.sample(100)
df_pred_actual_sample = df_pred_actual_sample.reset_index()
```

[60]:
```
df_pred_actual_sample.head()
```

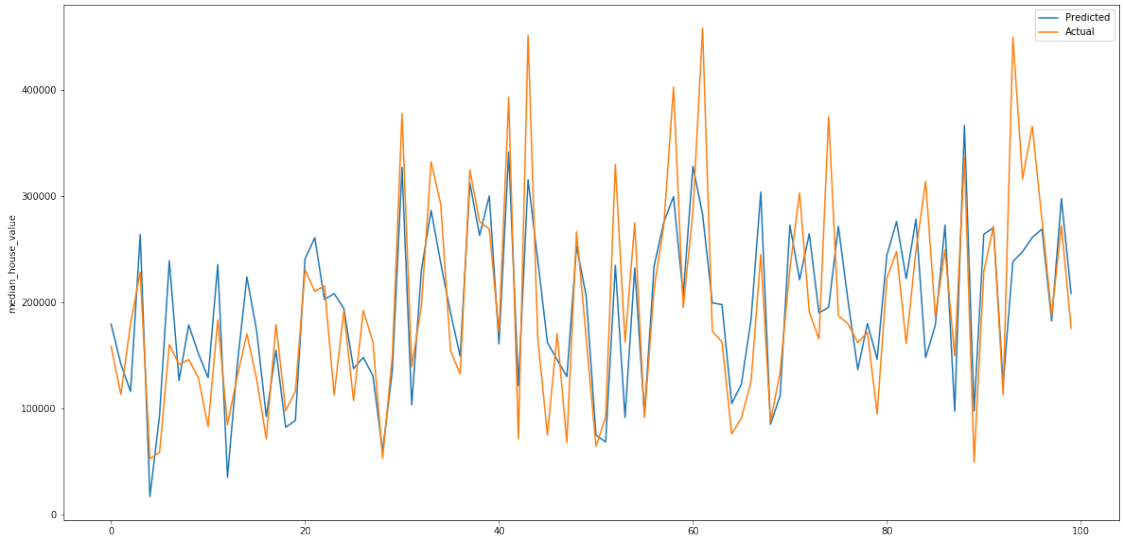
[60]:

	index	predicted	actual
0	15478	179328.355353	158500.0
1	12254	141857.426030	113100.0
2	19700	115999.487984	179700.0
3	16035	264011.682350	228600.0
4	9665	16853.180884	52600.0

[63]:
```
plt.figure(figsize = (20,10))

plt.plot(df_pred_actual_sample['predicted'], label='Predicted')
plt.plot(df_pred_actual_sample['actual'], label='Actual')

plt.ylabel('median_house_value')
plt.legend()
plt.show()
```



[ ]:

[ ]:

[ ]:

[ ]: