

▼ Exercise 2

Data is in <https://drive.google.com/drive/folders/101uoeXxByvdccuiZXCZSuXYQHDjeZo5v?usp=sharing> Any one with UMD Mail ID can access the data

This notebook is in

https://colab.research.google.com/drive/1Xa6lahU9g_kIEaKKiQIB8VLbkk5JoWL8?usp=sharing Any one with UMD Mail ID can access the file

```
!pip install sklearn
```

▼ 1. Loading the data house_train.csv - the training set and house_test.csv - the test set

```
from google.colab import drive
drive.mount('/content/drive')
PATH = '/content/drive/MyDrive/ENPM808W/HW2 data/'
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call dr

import all the libraries needed for the prediction and read the training and testing data from the csv file to pandas dataframe

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score

#read the csv file
house_train = pd.read_csv(PATH + 'house_train.csv')
house_test = pd.read_csv(PATH + 'house_test.csv')
```

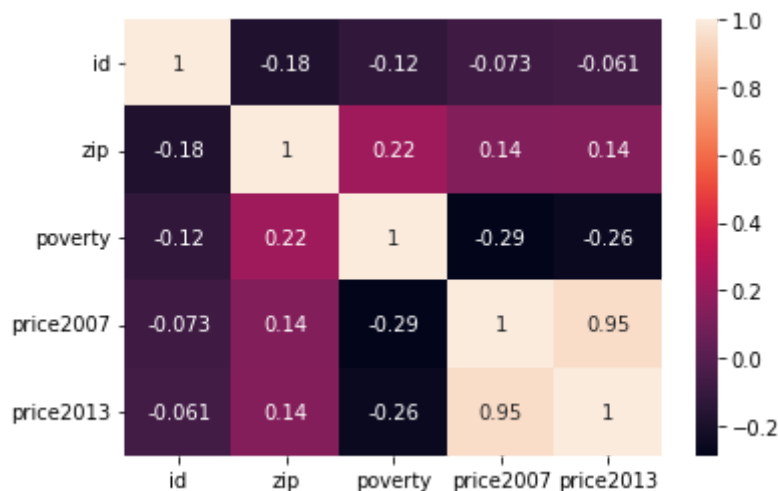
look for null values and plot a correlation heatmap to find the correlation columns price2013 and price2007 are highly correlated

```
house_train.isnull().sum() #found no null values
```

```
id          0
zip         0
state       0
county      0
poverty     0
price2007   0
price2013   0
dtype: int64
```

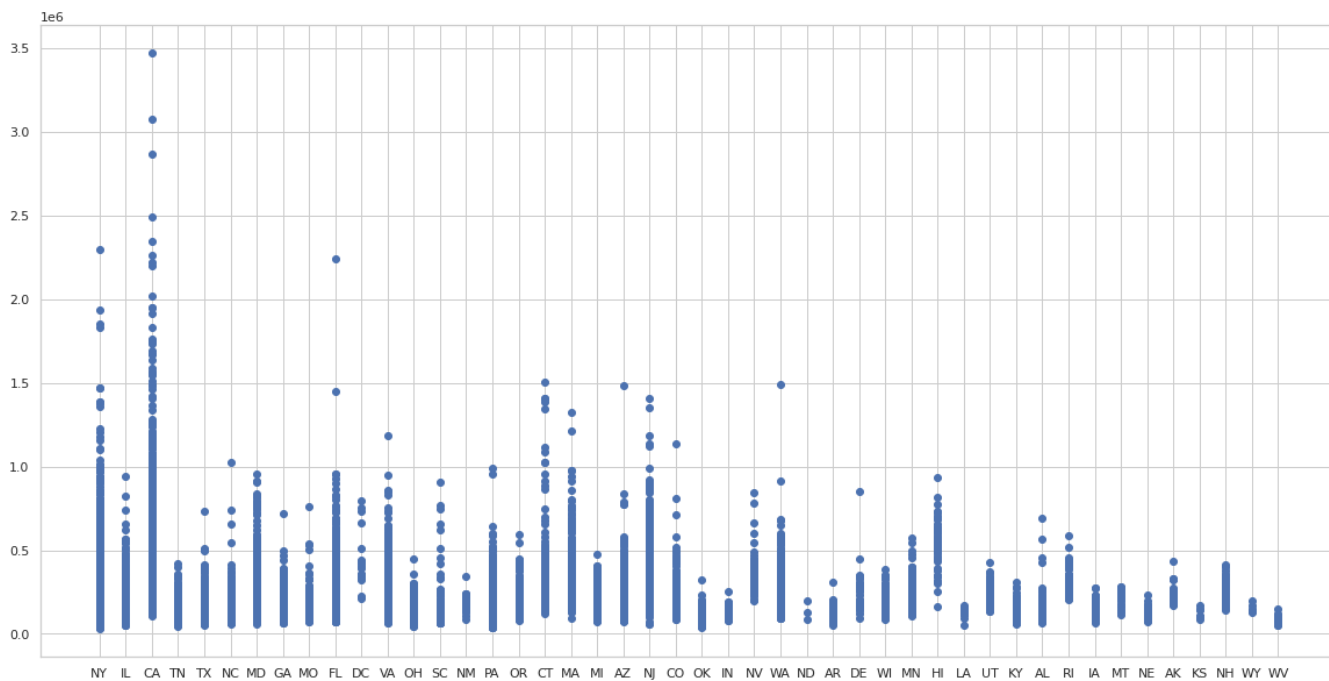
```
correlation_matrix=house_train.corr()
sns.heatmap(data=correlation_matrix,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa09464fe90>
```



Plot a scatter plot for State vs Price2007 and then with Price2013. Average house prices for 2007 and 2013 are the highest in CA

```
plt.figure(figsize=(20,10))
plt.scatter('state','price2007',data=house_train)
plt.show()
```



```
plt.figure(figsize=(20,10))
plt.scatter('state','price2013',data=house_train)
plt.show()
```

1e6

2. (40 points) Predict 2013 home prices using state information only.

Before taking state column for linear regression, we need to alter the state data as it is a categorical data column. I did so using 'pandas.get_dummies' method to get the dummy values. This will be the independent variable x. Our y variable is going to be price2013

```
x=pd.get_dummies(house_train['state'],drop_first=True)          # get dummies for categ
y=house_train['price2013']
print(x)
```

	AL	AR	AZ	CA	CO	CT	DC	DE	FL	GA	...	RI	SC	TN	TX	UT	VA	WA	\
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
...	
8968	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	
8969	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
8970	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	
8971	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
8972	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

	WI	WV	WY
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...
8968	0	0	0
8969	0	0	0
8970	0	0	0
8971	0	0	0
8972	0	0	0

[8973 rows x 45 columns]

```
modell = LinearRegression()
regla = modell.fit(x,y)
intercept = regla.intercept_
coef = regla.coef_
print("\n Intercept:", intercept)
```

```
print("\n Correlation Coefficient: \n", coef)
print("\n Score:", regla.score(x,y))
```

Intercept: 281729.99999999965

Correlation Coefficient:

```
[ -1.40892637e+05 -1.59320411e+05 -7.47779798e+04  2.30872896e+05
 -3.31445455e+04  6.70160377e+03  2.32558889e+05 -7.44633333e+04
 -1.06742121e+05 -1.39333721e+05  1.98581364e+05 -1.28573210e+05
 -1.06567771e+05 -1.46506471e+05 -1.43630000e+05 -1.35713333e+05
 -1.44257273e+05  5.77500000e+04  1.15519713e+02 -1.38182030e+05
 -7.54763277e+04 -1.27859197e+05 -5.57347619e+04 -1.32108019e+05
 -1.24863333e+05 -1.42711481e+05 -7.92850562e+04  2.47604645e+04
 -9.23758333e+04 -6.66660656e+04  1.70166176e+04 -1.64368283e+05
 -1.71825238e+05 -6.68951685e+04 -1.13104332e+05 -5.86692157e+04
 -1.35617786e+05 -1.50723089e+05 -1.21074828e+05 -4.37300000e+04
  1.83131280e+04 -1.52661217e+04 -1.18829479e+05 -1.83306923e+05
 -8.72500000e+04]
```

Score: 0.28677815383945593

```
max(coef)
```

232558.88888888978

```
min(coef)
```

-183306.92307692266

Above, we got a list of coefficients associated with each state with minimum and maximum coeffs. But I could not trace it back to the state from that format. Hence, below we are using statsmodel -> OLS to find least and most expensive average homes. You can see that intercept is same using both models.

```
from statsmodels.formula.api import ols
```

```
#Create a Model from a formula and dataframe.
```

```
#OLS measures the accuracy of a linear regression model.
```

```
#Assumption: A linear relationship exists between the dependent and predictor variabl
```

```
reglb = ols("price2013 ~ state", data=house_train).fit()
```

```
print('Intercept: ', reglb.params.Intercept)
```

```
reglb_state_params = reglb.params.drop(labels=['Intercept'])
```

```
print('Least Expensive Homes is in ', reglb_state_params.idxmin(), ' with coef: ', reg
```

```
print('Most Expensive Homes is in ', reglb_state_params.idxmax(), ' with coef: ', regl
```

```
#house_train.describe()
```

```
Intercept: 281730.00000000005
Least Expensive Homes is in state[T.WV] with coef: -183306.92307692376
Most Expensive Homes is in state[T.DC] with coef: 232558.88888888923
```

Answer these questions using all of the training data available.

a. What is the intercept? What does it correspond to? The intercept is the predicted value of Y when all independent variables (Xs) are set to 0. In other words, it is the value of Y when there is no independent variable present. The Intercept 281730.00000000005 here represents the estimated mean price 2013 given that state information is not available or in binary sense, 0.

b. How do you get this information from your regression?

I got this information about intercept from my linear regression model 'reg' made by making outcome/dependent variable as price2013 and independent/ variable used to predict price2013 as state information.

c. Based on your regression coefficients, what states have the most and least expensive average homes?

According to the coefficients of reg (reg.params) I found that the Least Expensive average homes are in state of WV (West Virginia) while the Most Expensive average homes are in state of DC (District of Columbia).

d. How do you get this information from your regression?

I got this information by finding the minimum and maximum from reg.params (after extracting the intercept coefficient) which lists the coefficients for each state individually along with State Initials. The states with the most expensive average homes have higher values for the intercept and all of the independent variables than the states with the least expensive average homes. This means that, on average, homes in these states are more expensive than homes in the states with the least expensive average homes.

e. What is the average price of homes in those states?

Average price of homes in WV => $281730 - 183306.9 = \$98423.1$

Average price of homes in DC => $281730 + 232558.9 = \$514288.9$

f. How do you get this information from your regression?

I got this information from the intercept of the overall linear model (reg.params.Intercept) and regression coefficients of the states (DC, WV) individually.

3. Predict 2013 home prices from state and county information.

```
#x2 = pd.get_dummies(data=house_train, columns=['state', 'county'], drop_first=True)
x2=pd.get_dummies(house_train[['state', 'county']],drop_first=True) # get dummies fo
y=house_train['price2013']
model2 = LinearRegression()
reg2a = model2.fit(x2,y)
```

```
#to find highest and lowest regression coefficients, we use OLS model
reg2b = ols("price2013 ~ state + county", data =house_train).fit()
reg2b_stateCounty_params = reg2b.params.drop(labels=['Intercept'])
print('Least Expensive Homes is in ', reg2b_stateCounty_params.idxmin(), ' with coef:
print('Most Expensive Homes is in ', reg2b_stateCounty_params.idxmax(), ' with coef:
```

```
Least Expensive Homes is in county[T.calaveras] with coef: -352624.3846659193
Most Expensive Homes is in county[T.pitkin] with coef: 857920.2646194387
```

a. What US counties have the highest and lowest regression coefficients? Why?

County Pitkin has highest and Calaveras has lowest regression coefficients

The counties with the highest regression coefficients are those with already high housing prices, whereas the counties with the lowest regression coefficients are those with already low housing prices. This is due to the fact that housing prices are a result of supply and demand, and there is less potential for them to rise in places where they are already high than there is in areas where they are already low

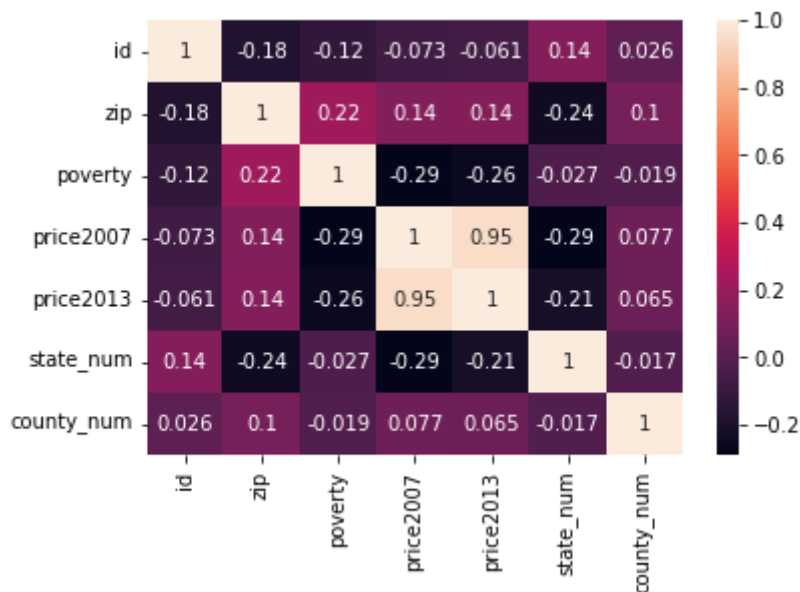
4. Challenge: Build a regressor that best predicts average home values in this dataset.

```
train = house_train.copy()
train.state = pd.Categorical(train.state)
train['state_num'] = train.state.cat.codes
train.county = pd.Categorical(train.county)
train['county_num'] = train.county.cat.codes
train.head()
```

	id	zip	state	county	poverty	price2007	price2013	state_num	county_
0	0	10467	NY	bronx	27.1	335200	294000	31	
1	1	11226	NY	kings	21.9	471500	471600	31	
2	2	60640	IL	cook	14.6	254600	174200	13	
3	3	94109	CA	san francisco	10.6	707100	822600	4	
4	4	11375	NY	queens	12.2	636400	681500	31	

```
correlation_matrix=train.corr()
sns.heatmap(data=correlation_matrix,annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa08f39d710>



Negative correlation indicates downward slope meaning as x increases/decreases, y decreases/increases and positive correlation indicates upward slope meaning as x increases/decreases, y increases/decreases

As you can see from the above heatmap, zip has positive correlation with price2013 : 0.14


poverty has negative correlation with price2013 : -0.26

price2007 has positive correlation with price2013 : 0.95

state has negative correlation with price2013 : -0.21

county has positive correlation with price2013 : 0.065

```
house_test.head()
#house_test['state'].unique()
```


	id	zip	state	county	poverty	price2007	
0	6	32162	FL	marion	13.0	265600	
1	13	78572	TX	hidalgo	34.0	79900	
2	20	11212	NY	kings	21.9	332000	
3	30	37042	TN	montgomery	12.7	98700	
4	37	85032	AZ	maricopa	12.9	266100	

```
print(house_test.isnull().sum())
```

```
id          0
zip         0
state       0
county      0
poverty     0
price2007   0
dtype: int64
```

```
#training data prep
train1 = house_train.copy()
#dummies_state = pd.get_dummies(train1['state'])
dummies_county = pd.get_dummies(train1['county'])
# Concatenate the dummies to original dataframe
merged_train = pd.concat([train1, dummies_county], axis=1)
# drop the values
merged_train.drop(['id', 'state', 'county', 'price2013'], axis=1, inplace=True)
merged_train.head()
```

	zip	poverty	price2007	adams	aiken	alachua	alamance	alameda	albany	al
0	10467	27.1	335200	0	0	0	0	0	0	
1	11226	21.9	471500	0	0	0	0	0	0	
2	60640	14.6	254600	0	0	0	0	0	0	
3	94109	10.6	707100	0	0	0	0	0	0	
4	11375	12.2	636400	0	0	0	0	0	0	

5 rows x 631 columns



```
#testing data prep
test = house_test.copy()
def missing_columns(col):
```

```

x = pd.get_dummies(test[col],drop_first=True)
missing_state = set(house_train[col]) - set(test[col])
for m in missing_state:
    x[m] = 0
return x
#dummies_state = missing_columns('state')
dummies_county = missing_columns('county')
dummies_county = dummies_county.reindex(sorted(dummies_county.columns), axis=1)
# Concatenate the dummies to original dataframe
merged_test = pd.concat([test, dummies_county], axis=1)
# drop the values
merged_test.drop(['id', 'state', 'county'], axis=1, inplace=True)
merged_test.head()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: PerformanceWarni
import sys

```

	zip	poverty	price2007	aiken	alachua	alamance	alameda	albany	albemarle
0	32162	13.0	265600	0	0	0	0	0	0
1	78572	34.0	79900	0	0	0	0	0	0
2	11212	21.9	332000	0	0	0	0	0	0
3	37042	12.7	98700	0	0	0	0	0	0
4	85032	12.9	266100	0	0	0	0	0	0

5 rows × 636 columns



training and testing data have uneven state and county rows when transformed using get dummies function (631 columns in training data and 636 columns in test data). So we need to drop some values from both dataframes to make it even

```

merged_train.drop(merged_train.columns.difference(merged_test.columns).tolist(), axis=1)
merged_train.head()

```

	zip	poverty	price2007	aiken	alachua	alamance	alameda	albany	albamarle
0	10467	27.1	335200	0	0	0	0	0	0
1	11226	21.9	471500	0	0	0	0	0	0
2	60640	14.6	254600	0	0	0	0	0	0

```
merged_test.drop(merged_test.columns.difference(merged_train.columns).tolist(), axis=
merged_test.head()
```

	zip	poverty	price2007	aiken	alachua	alamance	alameda	albany	albamarle
0	32162	13.0	265600	0	0	0	0	0	0
1	78572	34.0	79900	0	0	0	0	0	0
2	11212	21.9	332000	0	0	0	0	0	0
3	37042	12.7	98700	0	0	0	0	0	0
4	85032	12.9	266100	0	0	0	0	0	0

5 rows x 630 columns



```
#training the model
x_train = merged_train
y_train = train1['price2013']
model_train = LinearRegression()
reg_train = model_train.fit(x_train,y_train)
intercept = reg_train.intercept_
coef = reg_train.coef_
print("\n Intercept:", intercept)
print("\n Score:",reg_train.score(x_train,y_train))
```

Intercept: -67482.68880349316


Score: 0.9586468554922077

```
#testing the model
x_test = merged_test
test_pred = reg_train.predict(x_test)

f = open(PATH+'submission4_dradhagr.csv', 'w')
pred = pd.DataFrame(test_pred)
new_df = pd.concat([house_test['id'], pred], axis=1)
```

```
new_df.columns = ['id', 'prediction']
```

```
new_df
```

	id	prediction	
0	6	233566.526382	
1	13	85518.332049	
2	20	310279.995528	
3	30	73574.577777	
4	37	157242.839704	
...	
1058	9972	183892.343357	
1059	9991	185841.914289	
1060	9993	275994.573326	
1061	9996	676310.925843	
1062	10029	244232.915905	

1063 rows × 2 columns

```
new_df.to_csv(PATH+'submission4_dradhagr.csv', index=False)
```

**4a. Describe what you did to build the best predictor possible **

According to the heatmap, the best correlation for price2013 was with variable price2007 variables (0.95) along with other variables like zip and county giving positive correlations. state had a negative correlation with price2013, so I trained the model without the state.

4b. Give your best Kaggle score

48679.77371

**4c. Give your Kaggle username **

disharadhakrishna (Disha Radhakrishna) Apparently username cannot be modified by users. I shot a mail to the customer service guys to change it to dradhagr did. So please check for that incase you dont find disharadhakrishna. Thanks

5. Suppose you have 2 bags. Bag #1 has 1 black ball and 2 white balls. Bag #2 has 1 black ball and 3 white balls.

Suppose you pick a bag at random, and select a ball from

Answer: W = White B1 = Bag 1 B2 = Bag 2

You have a 50% or $1/2$ probability of choosing a bag out of 2. Considering independence between choosing a bag and selecting a white ball, $P(W \cap B1) = 2/3 * 1/2 = 1/3$

$$P(W \cap B2) = 3/4 * 1/2 = 3/8$$

By Law of total probability, $P(W)$ is the sum of these:

$$P(W) = P(W \cap B1) + P(W \cap B2) = 1/3 + 3/8 = 17/24 = 0.7083 \text{ or } 70.83\%$$

6. A soccer team wins 60% of its games when it scores the first goal, and 10% of its games when the opposing team scores first. If the team scores the first goal about 30% of the time, what fraction of the games does it win?

Answer: SF = Score First; OSF = Opposite Score First; W = win

Given that

$$P(W/SF) = 60\% = 60/100$$

$$P(W/OSF) = 10\% = 10/100$$

$$P(SF) = 30\% = 30/100$$

$$P(OSF) = 100\% - 30\% = 70\% = 70/100$$

Find $P(W)$

$$\text{By Law of Total Probability } P(W) = P(W/SF)P(SF) + P(W/OSF)P(OSF)$$

$$= (60/100 * 30/100) + (10/100 * 70/100)$$

$$= 18/100 + 7/100$$

$$= 25/100 \text{ or } 25\%$$

$$P(W) = 25/100$$

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 7:31 PM

