In [1]:
```
#Required imports for reading the file
import pandas as pd
```

In [2]:
```
#Problem 1: Prediction of house prices
```

In [3]:
```
```

Out[3]:

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)
0	2012.916667	32.0	84.87882	10	24.98298	121.54024	1	575
1	2012.916667	19.5	306.59470	9	24.98034	121.53951	2	1240
2	2013.583333	13.3	561.98450	5	24.98746	121.54391	3	1060
3	2013.500000	13.3	561.98450	5	24.98746	121.54391	2	875
4	2012.833333	5.0	390.56840	5	24.97937	121.54245	1	491
...	...	...	...	...	...	...	...	...
409	2013.000000	13.7	4082.01500	0	24.94155	121.50381	3	803
410	2012.666667	5.6	90.45606	9	24.97433	121.54310	2	1278
411	2013.250000	18.8	390.96960	7	24.97923	121.53986	1	503
412	2013.000000	8.1	104.81010	5	24.96674	121.54067	1	597
413	2013.500000	6.5	90.45606	9	24.97433	121.54310	2	1097

414 rows × 9 columns

Dependent Variable: House price of unit area

Type: Continuous

Regression is required

In [4]:
```
#Correlation matrix
```

Out[4]:

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms
Transaction date	1.000000	0.017542	0.060880	0.009544	0.035016	-0.041065	0.061985
House Age	0.017542	1.000000	0.025622	0.049593	0.054420	-0.048520	-0.008756
Distance from nearest Metro station (km)	0.060880	0.025622	1.000000	-0.602519	-0.591067	-0.806317	-0.046856
Number of convenience stores	0.009544	0.049593	-0.602519	1.000000	0.444143	0.449099	0.043638
latitude	0.035016	0.054420	-0.591067	0.444143	1.000000	0.412924	0.043921
longitude	-0.041065	-0.048520	-0.806317	0.449099	0.412924	1.000000	0.041680
Number of bedrooms	0.061985	-0.008756	-0.046856	0.043638	0.043921	0.041680	1.000000
House size (sqft)	0.068405	-0.060361	0.001795	0.033286	0.031696	0.009322	0.752276
House price of unit area	0.087529	-0.210567	-0.673613	0.571005	0.546307	0.523287	0.050265

House Age, Distance from metro, number of convenience store, lat, long are the most influencing factors on pricing.

In [5]:
```
#Split data into train and test
```

In [6]:
```

```

In [7]:
```
# Model 1: Linear regression
```

In [8]:
```
lr = LinearRegression()
```

In [9]:
```
#Evaluate
```

In [10]:
```

```

In [11]:
```

```

```
R2 Score:  0.4717938420680332
```

The model didn't perform well probably due to inter-correlation between independent variables

In [12]:
```
#SVM Regression: Uses kernel transformations to find the best fit
```

In [13]:
```
svr = SVR(kernel="linear")
```

In [14]:

In [15]:

```
R2 score:  0.47325318904682867
```

SVR doesn't perform well, the next model would randomforest regression, which is an ensemble model

In [16]:

In [17]:
```
rf = RandomForestRegressor()
```

In [18]:

In [19]:

```
R2 score:  0.5982682002170312
```

In [20]:
```
#Testing correlation again
```

Out[20]:

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms
Transaction date	1.000000	0.017542	0.060880	0.009544	0.035016	-0.041065	0.061985
House Age	0.017542	1.000000	0.025622	0.049593	0.054420	-0.048520	-0.008756
Distance from nearest Metro station (km)	0.060880	0.025622	1.000000	-0.602519	-0.591067	-0.806317	-0.046856
Number of convenience stores	0.009544	0.049593	-0.602519	1.000000	0.444143	0.449099	0.043638
latitude	0.035016	0.054420	-0.591067	0.444143	1.000000	0.412924	0.043921
longitude	-0.041065	-0.048520	-0.806317	0.449099	0.412924	1.000000	0.041680
Number of bedrooms	0.061985	-0.008756	-0.046856	0.043638	0.043921	0.041680	1.000000
House size (sqft)	0.068405	-0.060361	0.001795	0.033286	0.031696	0.009322	0.752276
House price of unit area	0.087529	-0.210567	-0.673613	0.571005	0.546307	0.523287	0.050265

In [21]:
```
#Normalising the data for better regression
```

In [22]:
```
```

In [23]:
```
mm_scaler = MinMaxScaler()
```

In [24]:
```
```

In [25]:
```
```

In [26]:
```
lr = LinearRegression()
lr = lr.fit(X_train,y_train)

y_pred_lr = lr.predict(X_test)
```

```
R2 Score:  0.5976102037734493
```

In [27]:
```
svr = SVR(kernel="linear")
svr = svr.fit(X_train,y_train)

y_pred_svr = svr.predict(X_test)
```

```
R2 score:  0.604292380041031
```

In [28]:
```
rf = RandomForestRegressor()
rf = rf.fit(X_train,y_train)

y_pred_rf = rf.predict(X_test)
```

```
R2 score:  0.7601839067632499
```

In [29]:
```
```

PART 2: PREDICTING SIMILAR TITLES

In [30]:
```
```

In [31]:
```
```

In [32]:

Out[32]:

	uniq_id	crawl_timestamp	product_url	product_na
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha S Wom Cycling Sh
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	FabHomeD Fabric Do Sofa
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Be
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha S Wom Cycling Sh
4	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-arn...	Sicon Purpose Ar Dog Sham
...	...	...	...	
19995	7179d2f6c4ad50a17d014ca1d2815156	2015-12-01 10:15:43 +0000	http://www.flipkart.com/walldesign-small-vinyl...	WALLDES SMALL VI STICI
19996	71ac419198359d37b8fe5e3fffdfee09	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-large-vinyl...	WALLMAN LARGE VI STICKI STICI
19997	93e9d343837400ce0d7980874ece471c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	EI COLLECT MED ACRY STICI
19998	669e79b8fa5d9ae020841c0c97d5e935	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	EI COLLECT MED ACRY STICI
19999	cb4fa87a874f715fff567f7b7b3be79c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	EI COLLECT MED ACRY STICI

20000 rows × 15 columns

In [33]:
```
import math
import re
from collections import Counter

WORD = re.compile(r"\w+")

#Gets cosine similarity for given pair of strings
def get_cosine(text1, text2):
    text1 = text1.lower()
    text2 = text2.lower()
    vec1 = text_to_vector(text1)
    vec2 = text_to_vector(text2)
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    sum1 = sum([vec1[x] ** 2 for x in list(vec1.keys())])
    sum2 = sum([vec2[x] ** 2 for x in list(vec2.keys())])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

#Converts text to vector using count method
def text_to_vector(text):
    words = WORD.findall(text)
```

In [34]:
```
def similar_title(title):
    scores = amazon_df["product_name"].apply(lambda x: get_cosine(x,title))
```

In [35]:
```
```

Out[35]:

	uniq_id	crawl_timestamp	product_url	product_name
28	171e0bcea390c17fd70e3ffa6c2cd187	2016-01-03 20:56:50 +0000	http://www.flipkart.com/fdt-women-s-leggings/p...	FDT WOMEN'S Leggings Pants

In [ ]:
```
```