In [1]:

```
import pandas as  pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import mpl_toolkits
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [2]:

```
df = pd.read_excel("DS - Assignment Part 1 data set.xlsx")
```

In [3]:

```
df.shape
```

Out[3]:

```
(414, 9)
```

In [4]:

```
df.head()
```

Out[4]:

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)	House price of unit area
0	2012.916667	32.0	84.87882	10	24.98298	121.54024	1	575	37.9
1	2012.916667	19.5	306.59470	9	24.98034	121.53951	2	1240	42.2
2	2013.583333	13.3	561.98450	5	24.98746	121.54391	3	1060	47.3
3	2013.500000	13.3	561.98450	5	24.98746	121.54391	2	875	54.8
4	2012.833333	5.0	390.56840	5	24.97937	121.54245	1	491	43.1

In [5]:

```
df.dtypes
```

Out[5]:

```
Transaction date                            float64
House Age                                   float64
Distance from nearest Metro station (km)    float64
Number of convenience stores                  int64
latitude                                    float64
longitude                                   float64
Number of bedrooms                            int64
House size (sqft)                             int64
House price of unit area                    float64
dtype: object
```

In [6]:

```
df.isna().sum()
```

Out[6]:

```
Transaction date                            0
House Age                                   0
Distance from nearest Metro station (km)    0
Number of convenience stores                0
latitude                                    0
longitude                                   0
Number of bedrooms                          0
House size (sqft)                           0
House price of unit area                    0
dtype: int64
```

In [7]:
```
df.describe()
```

Out[7]:

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)	House price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	2013.148953	17.712560	1083.885689	4.094203	24.969030	121.533361	1.987923	931.475845	37.980193
std	0.281995	11.392485	1262.109595	2.945562	0.012410	0.015347	0.818875	348.910269	13.606488
min	2012.666667	0.000000	23.382840	0.000000	24.932070	121.473530	1.000000	402.000000	7.600000
25%	2012.916667	9.025000	289.324800	1.000000	24.963000	121.528085	1.000000	548.000000	27.700000
50%	2013.166667	16.100000	492.231300	4.000000	24.971100	121.538630	2.000000	975.000000	38.450000
75%	2013.416667	28.150000	1454.279000	6.000000	24.977455	121.543305	3.000000	1234.750000	46.600000
max	2013.583333	43.800000	6488.021000	10.000000	25.014590	121.566270	3.000000	1500.000000	117.500000

EDA

In [8]:
```
df['Number of bedrooms'].value_counts()
```

Out[8]:
```
1    141
2    137
3    136
Name: Number of bedrooms, dtype: int64
```

In [9]:
```
fig = px.scatter(df, x="House price of unit area", y="House size (sqft)", title="Price per SQFT")
fig.show()
```

In [10]:

```
fig = px.scatter(df, x="Number of bedrooms", y="House price of unit area",title="Price vs Bedroom")
fig.show()
```

In [11]:

```
fig = px.scatter(df, x="latitude", y="longitude", color="House price of unit area", title="Price by Location")
fig.show()
```

In [12]:

```
fig = px.scatter(df, x="House Age", y="House price of unit area", color="House price of unit area", title="Price by House Age")
fig.show()
```

In [13]:

```
fig = px.scatter(df, x="Distance from nearest Metro station (km)", y="House price of unit area", title="Price as per nearest Metro Station")
fig.show()
```

Train Test Split

In [14]:

```
x, y = df.drop(['Transaction date', 'House price of unit area'], axis=1), df['House price of unit area']
```

In [15]:

```
x.head()
```

Out[15]:

	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)
0	32.0	84.87882	10	24.98298	121.54024	1	575
1	19.5	306.59470	9	24.98034	121.53951	2	1240
2	13.3	561.98450	5	24.98746	121.54391	3	1060
3	13.3	561.98450	5	24.98746	121.54391	2	875
4	5.0	390.56840	5	24.97937	121.54245	1	491

```
In [16]: y.head()
```

```
Out[16]: 0    37.9
         1    42.2
         2    47.3
         3    54.8
         4    43.1
         Name: House price of unit area, dtype: float64
```

```
In [17]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
In [32]: models = []
         scores = []

         def ml_model(model):
             model.fit(x_train,y_train)
             pred = model.predict(x_test)
             print("RMSE:",np.sqrt(mean_squared_error(y_test,pred)))
             print("R2 Score:",r2_score(y_test,pred))
             models.append(str(model).split('(')[0])
             scores.append(r2_score(y_test,pred)*100)
```

Linear Regression

```
In [33]: ml_model(LinearRegression())
```

```
RMSE: 8.801207423458912
R2 Score: 0.5367845818290011
```

Decision tree

```
In [34]: ml_model(DecisionTreeRegressor())
```

```
RMSE: 9.105119439084804
R2 Score: 0.5042419350411532
```

Random Forest

```
In [35]: ml_model(RandomForestRegressor())
```

```
RMSE: 6.804801364330921
R2 Score: 0.723095892104537
```

Gradient Boosting

```
In [36]: params = {'n_estimators': 400, 'max_depth':5, 'min_samples_split':2, 'learning_rate':0.1, 'loss':'ls'}
```

```
In [37]: ml_model(GradientBoostingRegressor(**params))
```

```
RMSE: 6.6731630043995285
R2 Score: 0.7337056447867589
```

Performance Comparison of Models

In [42]:

```
model_performances = pd.DataFrame([models,scores]).T
model_performances.columns = ['Model','R2 Score(%)']
model_performances.set_index('Model',inplace=True)
model_performances = model_performances.sort_values('R2 Score(%)',ascending=False)
model_performances
```

Out[42]:

	R2 Score(%)
Model	
GradientBoostingRegressor	73.370564
RandomForestRegressor	72.309589
LinearRegression	53.678458
DecisionTreeRegressor	50.424194

In [ ]:

```

```