```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataframe = pd.read_csv("/content/Real estate.csv")
```

```
dataframe.head()
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8

```
dataframe.shape
```

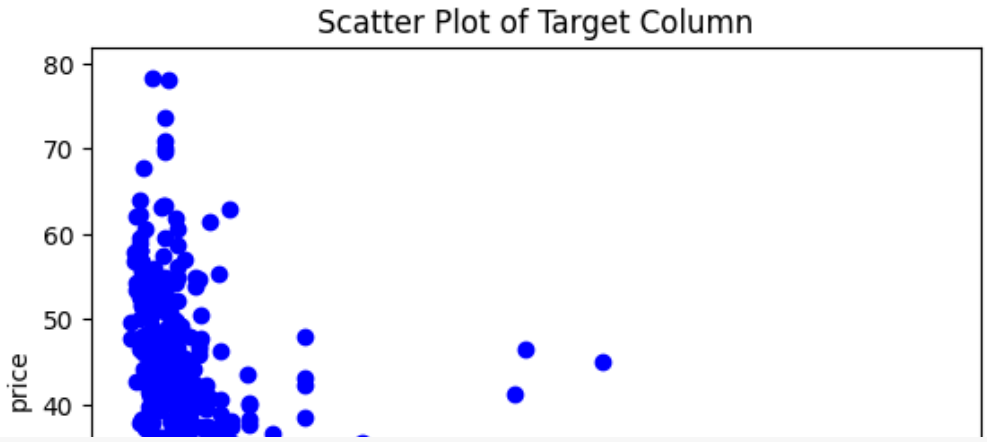
```
(414, 8)
```

```
dataframe.columns
```

```
Index(['No', 'X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

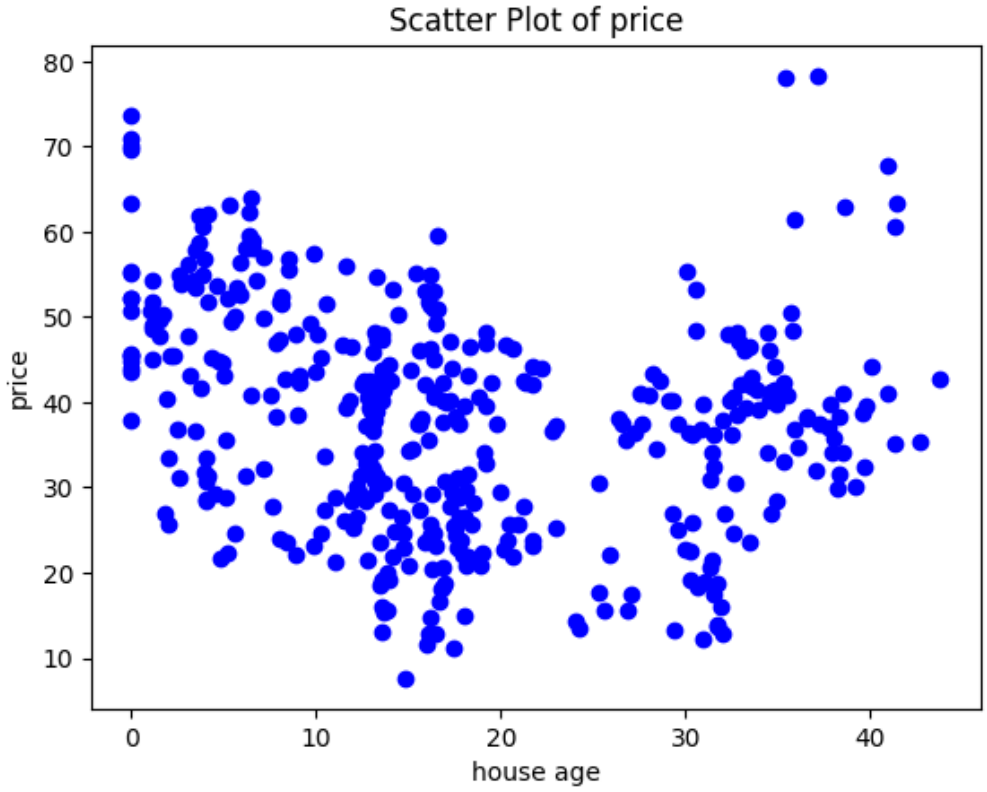
```
import matplotlib.pyplot as plt

plt.scatter(dataframe['X3 distance to the nearest MRT station'], dataframe['Y house price of unit area']
plt.xlabel('Nearest MRT')
plt.ylabel('price')
plt.title('Scatter Plot of Target Column')
plt.show()
```



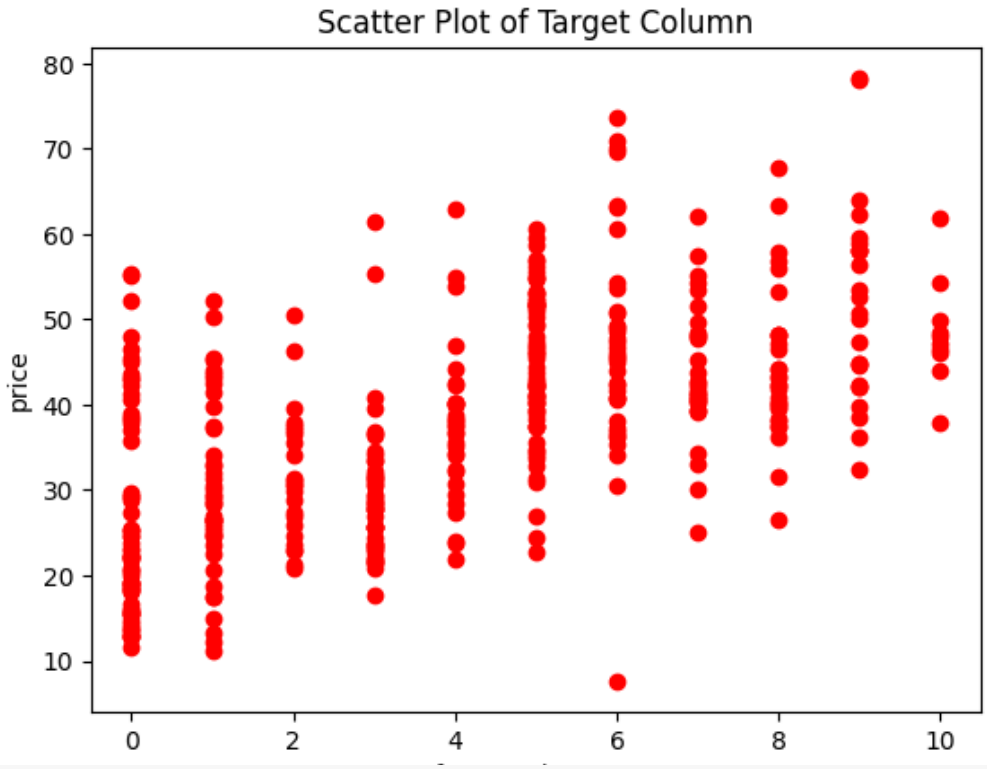

```
import matplotlib.pyplot as plt

plt.scatter(dataframe['X2 house age'], dataframe['Y house price of unit area'], color='blue')
plt.xlabel('house age')
plt.ylabel('price')
plt.title('Scatter Plot of price')
plt.show()
```




```
import matplotlib.pyplot as plt

plt.scatter(dataframe['X4 number of convenience stores'], dataframe['Y house price of unit area'], color
plt.xlabel('no of convenience stores')
plt.ylabel('price')
plt.title('Scatter Plot of Target Column')
plt.show()
```




```
dataframe.drop(columns = ["X1 transaction date","No"], inplace =True)
```

```
dataframe.drop(columns = ["X5 latitude","X6 longitude"], inplace =True)
```

```
dataframe
```

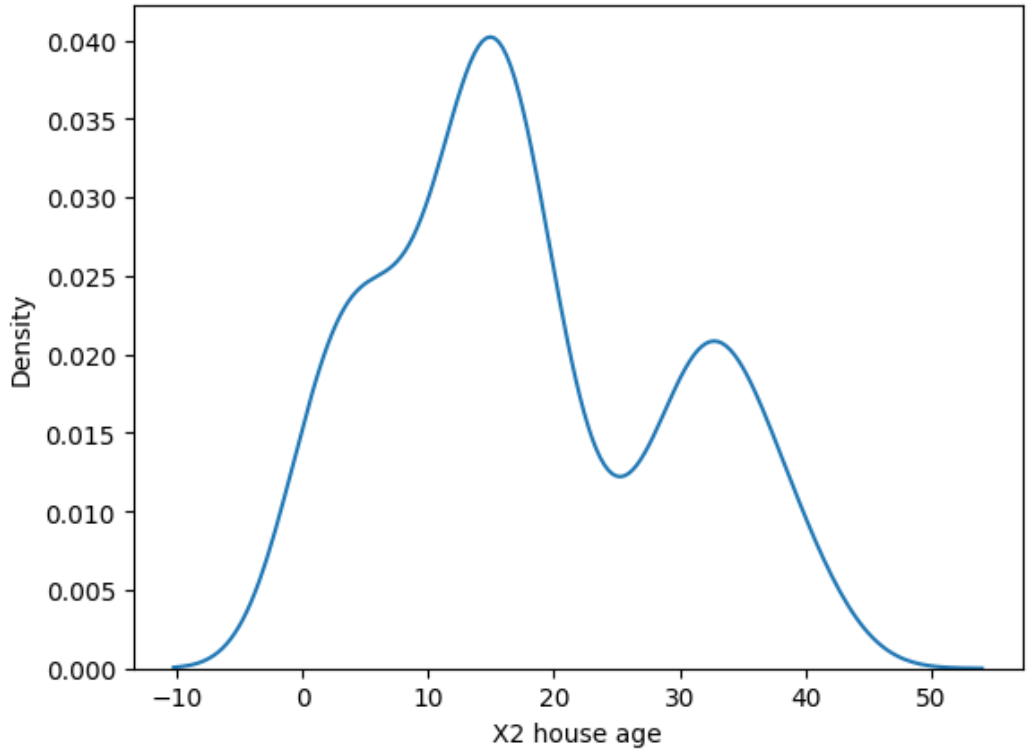
	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	Y house price of unit area
0	32.0	84.87882	10	37.9
1	19.5	306.59470	9	42.2
2	13.3	561.98450	5	47.3
3	13.3	561.98450	5	54.8
4	5.0	390.56840	5	43.1
...	...	...	...	...
409	13.7	4082.01500	0	15.4
410	5.6	90.45606	9	50.0
411	18.8	390.96960	7	40.6
412	8.1	104.81010	5	52.5
413	6.5	90.45606	9	63.9

414 rows × 4 columns

Univariate analysis

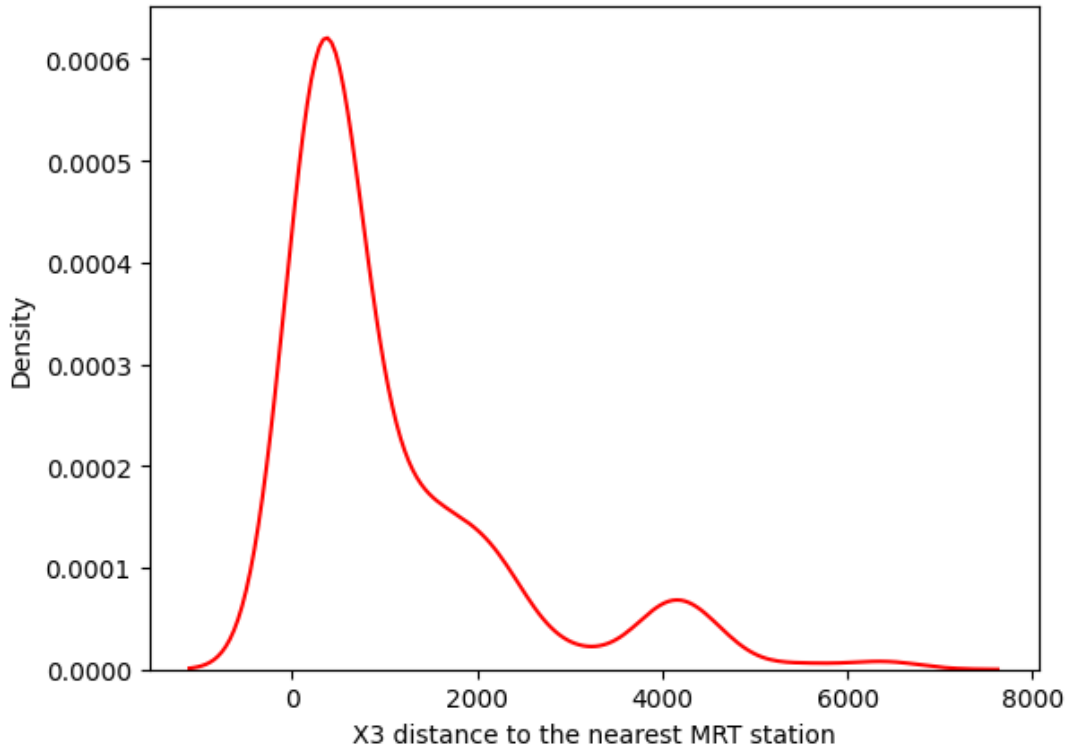
```
sns.kdeplot(x=dataframe["X2 house age"])
```

```
<Axes: xlabel='X2 house age', ylabel='Density'>
```



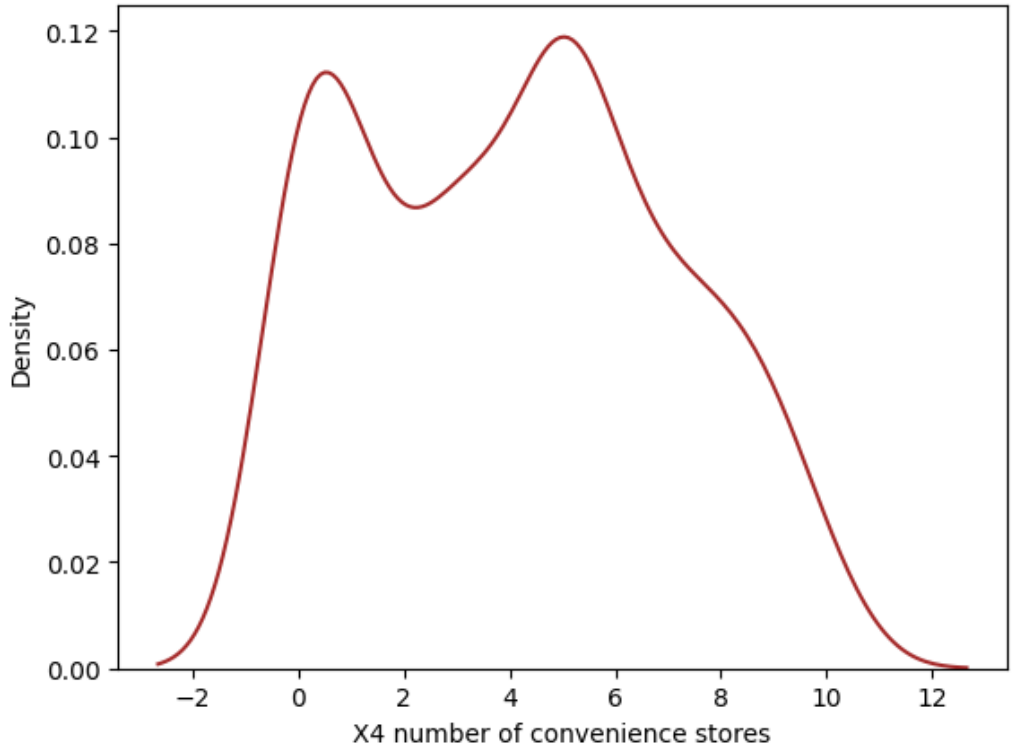

```
sns.kdeplot(x=dataframe["X3 distance to the nearest MRT station"],color ='red')
```

```
<Axes: xlabel='X3 distance to the nearest MRT station', ylabel='Density'>
```



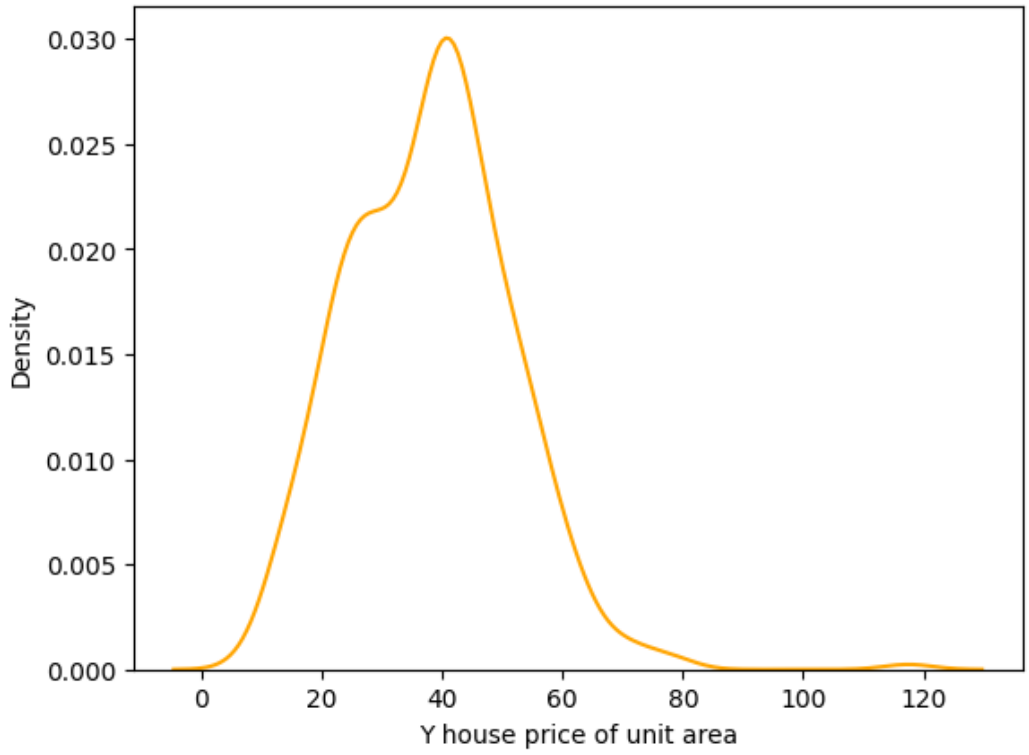
```
sns.kdeplot(x=dataframe["X4 number of convenience stores"],color ='brown')
```

```
<Axes: xlabel='X4 number of convenience stores', ylabel='Density'>
```



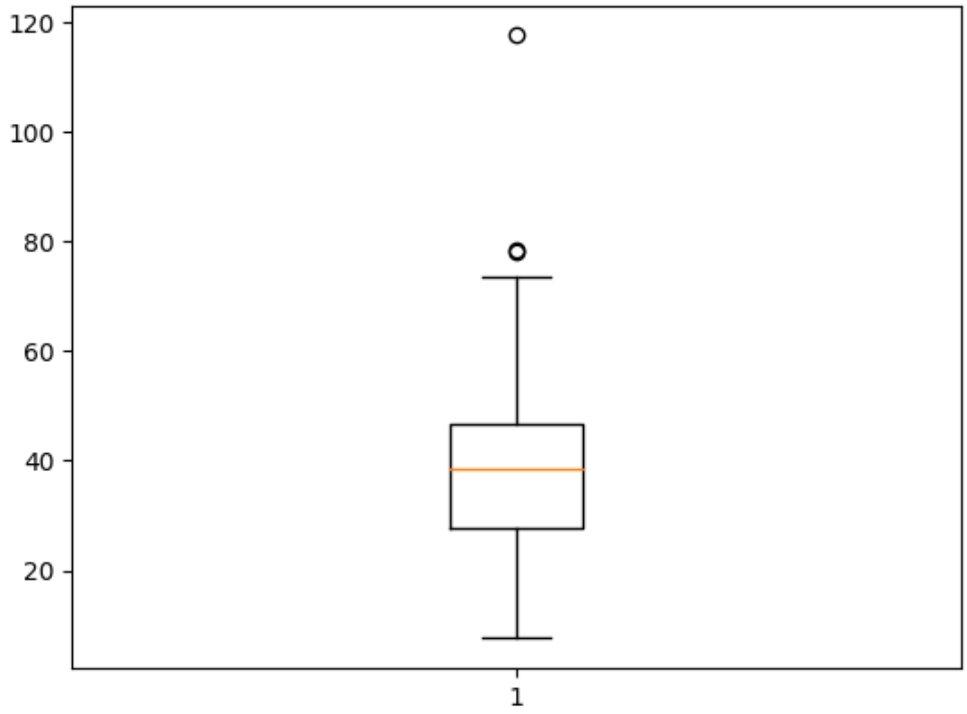

```
sns.kdeplot(x=dataframe["Y house price of unit area"],color ='orange')
```

```
<Axes: xlabel='Y house price of unit area', ylabel='Density'>
```




```
plt.boxplot(x=dataframe["Y house price of unit area"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7c633f7fab30>,
  <matplotlib.lines.Line2D at 0x7c633f7f9a20>],
 'caps': [<matplotlib.lines.Line2D at 0x7c633f7faa10>,
  <matplotlib.lines.Line2D at 0x7c633f7f8730>],
 'boxes': [<matplotlib.lines.Line2D at 0x7c633f7fbcd0>],
 'medians': [<matplotlib.lines.Line2D at 0x7c633f7fa8f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7c633f7fb8b0>],
 'means': []}
```




```
print(dataframe['Y house price of unit area'].nlargest(5))
```

```
270    117.5
220     78.3
312     78.0
166     73.6
105     71.0
Name: Y house price of unit area, dtype: float64
```

```
max_value_index = dataframe['Y house price of unit area'].idxmax()
```

```
dataframe.loc[max_value_index]
```

```
X2 house age                             10.8000
X3 distance to the nearest MRT station  252.5822
X4 number of convenience stores           1.0000
Y house price of unit area              117.5000
Name: 270, dtype: float64
```

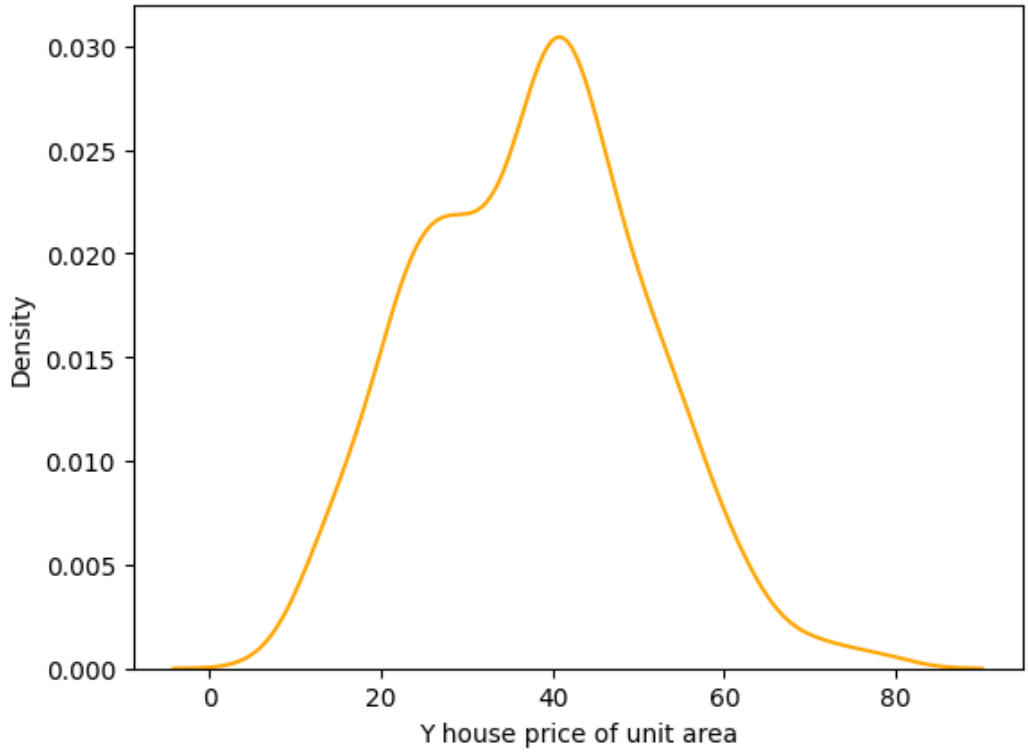
```
dataframe.drop(270, inplace=True)
```

```
dataframe["Y house price of unit area"].nlargest(5)
```

```
220    78.3
312    78.0
166    73.6
105    71.0
16     70.1
Name: Y house price of unit area, dtype: float64
```

```
sns.kdeplot(dataframe["Y house price of unit area"], color ="orange")
```

```
<Axes: xlabel='Y house price of unit area', ylabel='Density'>
```




```
dataframe.describe()
```

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	Y house price of unit area
count	413.000000	413.000000	413.000000	413.000000
mean	17.729298	1085.898530	4.101695	37.787651
std	11.401205	1262.974876	2.945182	13.046097
min	0.000000	23.382840	0.000000	7.600000
25%	9.000000	289.324800	1.000000	27.700000
50%	16.100000	492.231300	4.000000	38.400000
75%	28.200000	1455.798000	6.000000	46.600000
max	43.800000	6488.021000	10.000000	78.300000

```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 413
Data columns (total 4 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   X2 house age                            413 non-null    float64
 1   X3 distance to the nearest MRT station  413 non-null    float64
 2   X4 number of convenience stores         413 non-null    int64  
 3   Y house price of unit area              413 non-null    float64
dtypes: float64(3), int64(1)
memory usage: 32.3 KB
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
```

```
X_train, X_test ,Y_train, Y_test = train_test_split(dataframe.drop(columns =["Y house price of unit area
```

```
norm_transformer = ColumnTransformer(
    transformers = [ ('scaling', MinMaxScaler(),['X3 distance to the nearest MRT station'])]
, remainder = "passthrough")
```

```
X_train['X3 distance to the nearest MRT station']=norm_transformer.fit_transform(X_train)
```

```
X_test['X3 distance to the nearest MRT station']=norm_transformer.transform(X_test)
```

```
X_train.describe()
```

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores
count	330.000000	330.000000	330.000000
mean	17.530909	0.166309	4.084848
std	11.336350	0.199201	2.934764
min	0.000000	0.000000	0.000000
25%	9.025000	0.041138	1.000000
50%	15.950000	0.072635	4.000000
75%	27.400000	0.221577	6.000000
max	43.800000	1.000000	10.000000

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

```
lr.fit(X_train,Y_train)
```

▾ LinearRegression
LinearRegression()

```
Y_pred =lr.predict(X_test)
```

```
X_test
```

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores
377	3.9	0.004065	8
170	24.0	0.696760	0
230	4.0	0.328556	3
331	25.6	0.695523	0
337	31.3	0.089329	5
...	...	...	...
308	16.4	0.041138	5
100	17.5	0.145618	4
7	20.3	0.040872	6
22	14.7	0.206780	1
68	30.4	0.068193	6

83 rows × 3 columns

```
Y_test
```

```
from sklearn.metrics import mean_squared_error as mse, r2_score
```

```
mse = mse(Y_test, Y_pred)
```

```
rmse =np.sqrt(mse)
```

```
print(rmse)
```

```
6.52805867931348
```

```
print(mse)
```

```
42.61555012056005
```

```
print(r2_score(Y_test,Y_pred))
```

```
0.7118820479919712
```

```
from sklearn.linear_model import Ridge
```

```
ri = Ridge(alpha =2)
```

```
ri.fit(X_train,Y_train)
```

```
Ridge
Ridge(alpha=2)
```

```
ri_pred = ri.predict(X_test)
```

```
print(r2_score(Y_test,ri_pred))
```

```
0.6985773713549839
```

```
Y_train_pred = ri.predict(X_train)
```

```
print(r2_score(Y_train,Y_train_pred))
```

```
0.5520697253523801
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline


degree = 2
poly_features = PolynomialFeatures(degree=degree)

linear_regression = LinearRegression()

poly_regression_model = make_pipeline(poly_features, linear_regression)
```

```
poly_regression_model.fit(X_train,Y_train)
```

```
Pipeline
PolynomialFeatures
LinearRegression
```

```
poly_pred = poly_regression_model.predict(X_test)
```

```
print(r2_score(Y_test,poly_pred))
```

```
0.7852937177172077
```