Alyakbar Ahmed Sheikh RegNo. BCS-03-0003/2022

Assignment 2 Question

1. Using the housing prices data. Analyze it using random forests and decision trees. Make a report on which model is preferred for analyzing that data Importing data and aalso the csv file

In [ ]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import pandas as pd
df = pd.read_csv('Real estate.csv')
print(df)
```

```
      No  X1 transaction date  X2 house age  \
0      1             2012.917          32.0
1      2             2012.917          19.5
2      3             2013.583          13.3
3      4             2013.500          13.3
4      5             2012.833           5.0
..   ...                  ...           ...
409  410             2013.000          13.7
410  411             2012.667           5.6
411  412             2013.250          18.8
412  413             2013.000           8.1
413  414             2013.500           6.5

     X3 distance to the nearest MRT station  X4 number of convenience stores  \
0                                  84.87882                               10
1                                 306.59470                                9
2                                 561.98450                                5
3                                 561.98450                                5
4                                 390.56840                                5
..                                      ...                              ...
409                              4082.01500                                0
410                                90.45606                                9
411                               390.96960                                7
412                               104.81010                                5
413                                90.45606                                9

     X5 latitude  X6 longitude  Y house price of unit area
0       24.98298     121.54024                        37.9
1       24.98034     121.53951                        42.2
2       24.98746     121.54391                        47.3
3       24.98746     121.54391                        54.8
4       24.97937     121.54245                        43.1
..           ...           ...                         ...
409     24.94155     121.50381                        15.4
410     24.97433     121.54310                        50.0
411     24.97923     121.53986                        40.6
412     24.96674     121.54067                        52.5
413     24.97433     121.54310                        63.9

[414 rows x 8 columns]
```

In [ ]:
```
pip install seaborn
```

```
Collecting seaborn
  Obtaining dependency information for seaborn from https://files.pythonhosted.org/packages/83/11/00d3c3dfc25ad54e731d91449895a79e4bf2384dc3ac01809010ba88f6d5/seaborn-0.13.2-py3-none-any.whl.metadata
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\python312\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\python312\lib\site-packages (from seaborn) (2.2.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\python312\lib\site-packages (from seaborn) (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
   ---------------------------------------- 0.0/294.9 kB ? eta -:--:--
   ---------------------------------------- 0.0/294.9 kB ? eta -:--:--
   - -------------------------------------- 10.2/294.9 kB ? eta -:--:--
   ----- ---------------------------------- 41.0/294.9 kB 393.8 kB/s eta 0:00:01
   ---------- ----------------------------- 92.2/294.9 kB 581.0 kB/s eta 0:00:01
   ------------------------ ------------ 194.6/294.9 kB 980.4 kB/s eta 0:00:01
   ------------------------------------ -- 276.5/294.9 kB 1.1 MB/s eta 0:00:01
   ---------------------------------------- 294.9/294.9 kB 1.1 MB/s eta 0:00:00
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

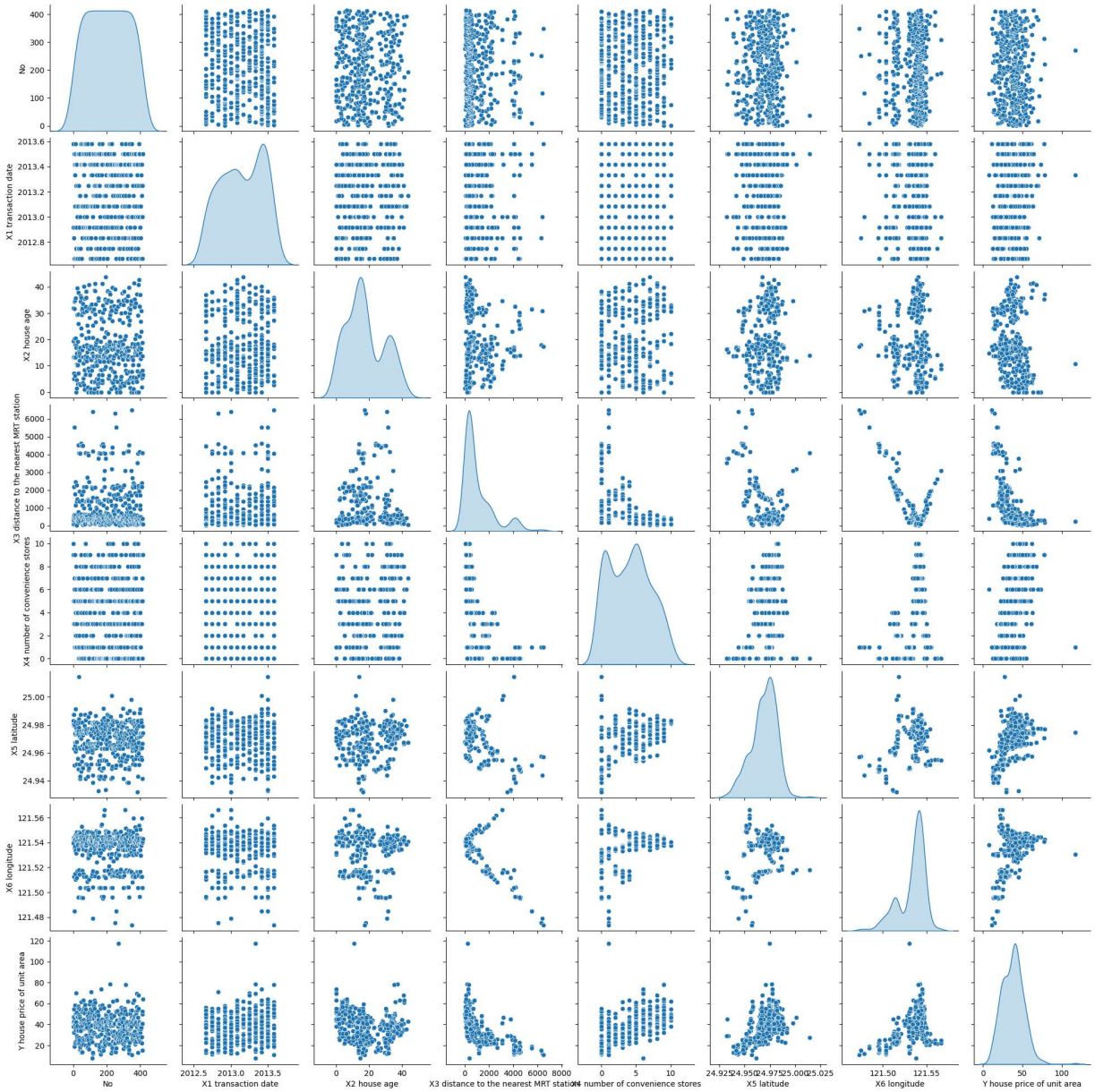
In [ ]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
print(df.head())
print(df.info())
print(df.describe())

sns.pairplot(df, diag_kind='kde')
plt.show()
```

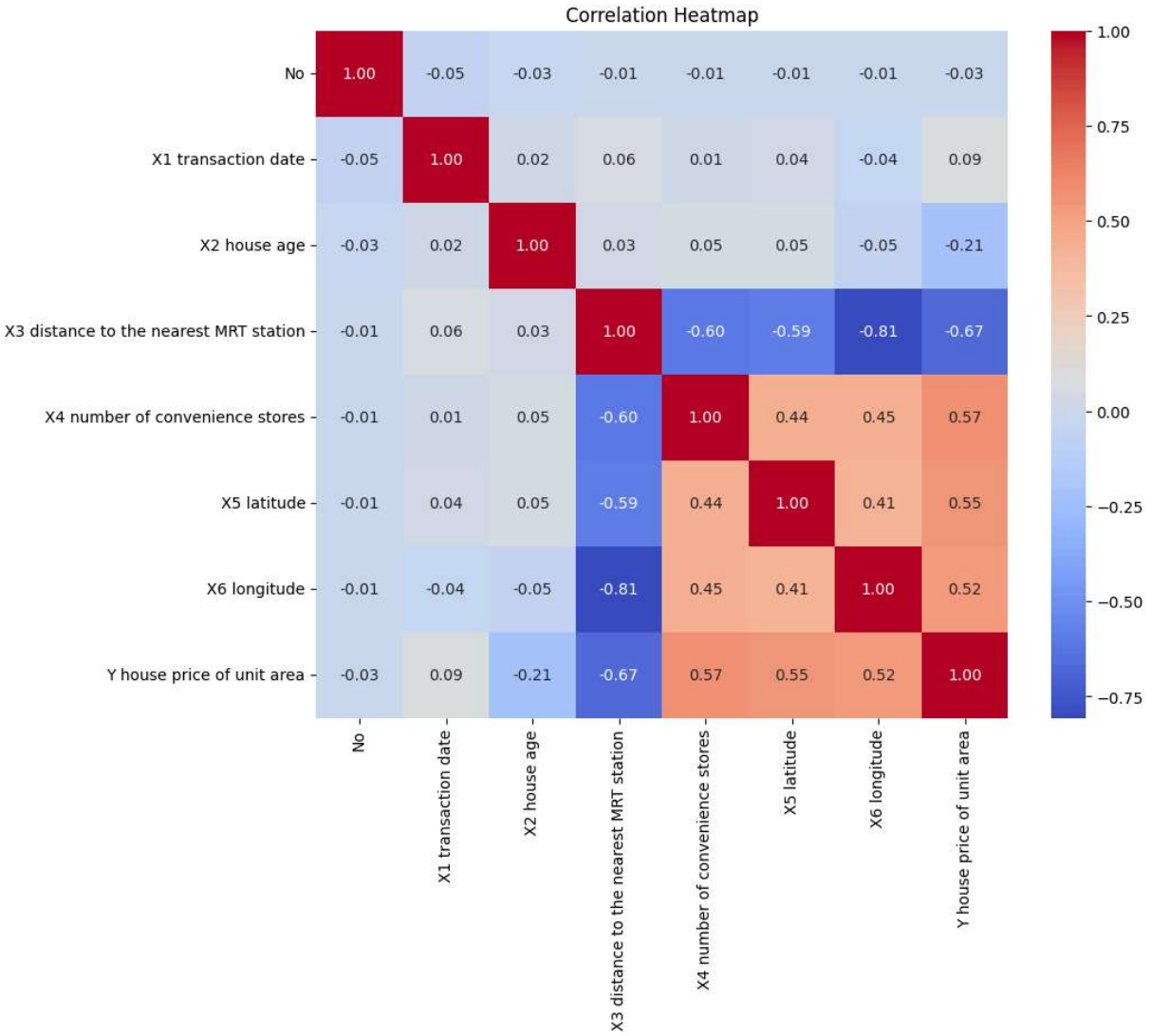
```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

```
   No  X1 transaction date  X2 house age  \
0   1             2012.917          32.0
1   2             2012.917          19.5
2   3             2013.583          13.3
3   4             2013.500          13.3
4   5             2012.833           5.0

   X3 distance to the nearest MRT station  X4 number of convenience stores  \
0                                84.87882                                10
1                               306.59470                                 9
2                               561.98450                                 5
3                               561.98450                                 5
4                               390.56840                                 5

   X5 latitude  X6 longitude  Y house price of unit area
0     24.98298     121.54024                        37.9
1     24.98034     121.53951                        42.2
2     24.98746     121.54391                        47.3
3     24.98746     121.54391                        54.8
4     24.97937     121.54245                        43.1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   No                                      414 non-null    int64
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64
 5   X5 latitude                             414 non-null    float64
 6   X6 longitude                            414 non-null    float64
 7   Y house price of unit area              414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
None
               No  X1 transaction date  X2 house age  \
count  414.000000           414.000000    414.000000
mean   207.500000          2013.148971     17.712560
std    119.655756             0.281967     11.392485
min      1.000000          2012.667000      0.000000
25%    104.250000          2012.917000      9.025000
50%    207.500000          2013.167000     16.100000
75%    310.750000          2013.417000     28.150000
max    414.000000          2013.583000     43.800000

       X3 distance to the nearest MRT station  \
count                              414.000000
mean                              1083.885689
std                               1262.109595
min                                 23.382840
25%                                289.324800
50%                                492.231300
75%                               1454.279000
max                               6488.021000
```

	X4 number of convenience stores	X5 latitude	X6 longitude	\
count	414.000000	414.000000	414.000000	
mean	4.094203	24.969030	121.533361	
std	2.945562	0.012410	0.015347	
min	0.000000	24.932070	121.473530	
25%	1.000000	24.963000	121.528085	
50%	4.000000	24.971100	121.538630	
75%	6.000000	24.977455	121.543305	
max	10.000000	25.014590	121.566270	

	Y house price of unit area
count	414.000000
mean	37.980193
std	13.606488
min	7.600000
25%	27.700000
50%	38.450000
75%	46.600000
max	117.500000





preparing the data

In [ ]:
```
X = df.drop(['No', 'Y house price of unit area'], axis=1)
y = df['Y house price of unit area']
```

Spliting the data

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Train Decision Tree model

In [ ]:
```
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeRegressor(random_state=42)
```

Train Random Forest model

In [ ]:
```
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
```

Out[ ]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Evaluate models

In [ ]:
```
dt_predictions = dt_model.predict(X_test)
rf_predictions = rf_model.predict(X_test)

dt_rmse = np.sqrt(mean_squared_error(y_test, dt_predictions))
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_predictions))

print("Decision Tree RMSE:", dt_rmse)
print("Random Forest RMSE:", rf_rmse)
```

```
Decision Tree RMSE: 8.153002255803228
Random Forest RMSE: 5.693072342983686
```

Performance Metrics:

1. **Root Mean Squared Error (RMSE):** This metric measures the average deviation of the predictions made by the model from the actual values in the testing set. Lower RMSE indicates better performance.

Results:

- **Decision Tree RMSE:** 8.153
- **Random Forest RMSE:** 5.693

Analysis:

- The Decision Tree model achieved an RMSE of 8.153, while the Random Forest model achieved an RMSE of 5.693.
- Based solely on the RMSE values, we can observe that the Random Forest model outperforms the Decision Tree model, indicating that it provides more accurate predictions on average.
- The Random Forest model generally outperforms the Decision Tree model due to its ensemble nature, which reduces overfitting and captures more complex relationships in the data.