Real estate price predictor

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
data=pd.read_csv('datasets/Real_estate_prediction.csv')
```

In [ ]:
```
data.head()
```

Data Cleaning

changing column headings

In [ ]:
```
new_headings=['No.','date','age','dist_mrt','num_stores','latitude','longitude
data=data.set_axis(new_headings,axis=1)
```

In [ ]:
```
data.set_index('No.',inplace=True)
```

changing data type

In [ ]:
```
data['date']=data['date'].astype(int)
```

checking for duplicates

In [ ]:
```
duplicates=data.duplicated()
duplicates.any()
```

checking for null values

In [ ]:
```
data.isnull().sum()
```

dealing with outliers

setting graph attributes

In [ ]:
```
sns.set(style="whitegrid", palette="deep", rc={"figure.figsize": (6, 4)})
```

- age

In [ ]:
```
sns.boxplot(x=data['age'])
plt.title('Box Plot of Age')
```

- distance to metro station

In [ ]:
```
sns.boxplot(x=data['dist_mrt'])
```

- num_stores

In [ ]:
```
sns.boxplot(x=data['num_stores'])
```

- latitude and longitude

In [ ]:
```
sns.boxplot(x=data['latitude'])
```

In [ ]:
```
sns.boxplot(x=data['longitude'])
```

- house price

In [ ]:
```
sns.boxplot(x=data['price'])
```

there are ouliers in:

- dist_mrt
- latitude and longitude
- price

removing outliers

In [ ]:
```
data = data[data['price']<80]
data = data[data['dist_mrt']<3000]
data = data[(data['longitude']>121.50) & (data['longitude']<121.56)]
data=data[(data['latitude'] >24.92) & (data['latitude'] < 25.00)]
```

exploring data

In [ ]:
```
data.hist(bins=50, figsize=(20,15))
plt.show()
```

In [ ]:
```
sns.pairplot(data,x_vars=['date', 'age', 'dist_mrt', 'num_stores','latitude',
```

checking correlation

In [ ]:
```

plt.figure(figsize=(6, 4))
sns.heatmap(data.corr(),cmap='Blues',linewidths=0.5,annot=True)
```

Splitting Data

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

- x1 - the predictor variables
- y1 - target variable (what we have to predict)

In [ ]:
```
x1 = data.drop( ['price'], axis=1)
y1 = data.price
```

- The dataset is split into training (80%) and testing (20%) sets.
- random_state=100 ensures the split is reproducible.

In [ ]:
```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1,y1, test_size = 0.2
```

Linear Regression Model

fitting the model

In [ ]:
```
reg = LinearRegression()
reg.fit (x1_train, y1_train)
np.set_printoptions(suppress=True) #to remove scientifc notation
reg.coef_
```

the coeffecients in the array show that by increase in one unit of the predictor varibale the target variable changes by that amount

evaluating model

In [ ]:
```
y1_pred = reg.predict(x1_test)
print('r2 Score : ', r2_score(y1_test, y1_pred))
```

- An R^2 score of 0.598 suggests that the model has a moderate to strong fit to the data.
- 59.825% of the variance in the target variable can be explained by the features in the model.

visualising errors

In [ ]:
```
df=pd.DataFrame({'actual':y1_test,'predictions':y1_pred})
df['predictions']=round(df['predictions'])
df.head()
```

In [ ]:
```
sns.regplot(x='actual',y='predictions',data=df)
```

creating function to predict diffrent entries

In [ ]:
```
def predict_price(input_data):
    predicted_price= reg.predict(input_data)
    return predicted_price
```

In [ ]:
```
a=np.array([[2012,32,84.8,10,24.982,121.540]])
predict_price(a)
```

Ridge and Lasso Regression

- Ridge Regression: It smooths out our predictions by putting a limit on how much the factors can influence the outcome.
- Lasso Regression: It helps us pick out the most important factors for prediction and ignores the less important ones.

In [ ]:
```
from sklearn.linear_model import Ridge, Lasso
```

In [ ]:
```
ridge_reg = Ridge(alpha=1, solver="cholesky")
ridge_reg.fit (x1_train, y1_train)
y1_pred_ridge = ridge_reg.predict(x1_test)
print('r^2 Score : ', r2_score(y1_test, y1_pred_ridge))
```

In [ ]:
```
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit (x1_train, y1_train)
y1_pred_lasso = lasso_reg.predict(x1_test)
print('r^2 Score : ', r2_score(y1_test, y1_pred_lasso))
```

In [ ]:
```
```