Lab 7 (12-03-2024)

Boosted Trees

XGBoost

dataset : Housing price dataset

In [ ]:
```
Registration_Number = "22011103010"
Name = "Deepthi I"

# Python Program to Get IP Address
import socket
hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)

print("My name is " + Name + " and my roll no : " + Registration_Number)
print("Computer IP Address is: " + IPAddr)
```

```
My name is Deepthi I and my roll no : 22011103010
Computer IP Address is: 10.18.90.96
```

In [ ]:
```
## Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import cross_val_score
```

In [ ]:
```
df = pd.read_csv("real_estate.csv")
df.head()
```

Out[ ]:

	Address	Rooms	Price	Bedroom2	Bathroom	Landsize	YearBuilt	Regionnam
0	85 Turner St	2	1480000	2	1	202	NaN	Northern Metropolitar
1	25 Bloomburg St	2	1035000	2	1	156	1900.0	Northern Metropolitar
2	5 Charles St	3	1465000	3	2	134	1900.0	Northern Metropolitar
3	40 Federation La	3	850000	3	2	94	NaN	Northern Metropolitar
4	55a Park St	4	1600000	3	1	120	2014.0	Northern Metropolitar

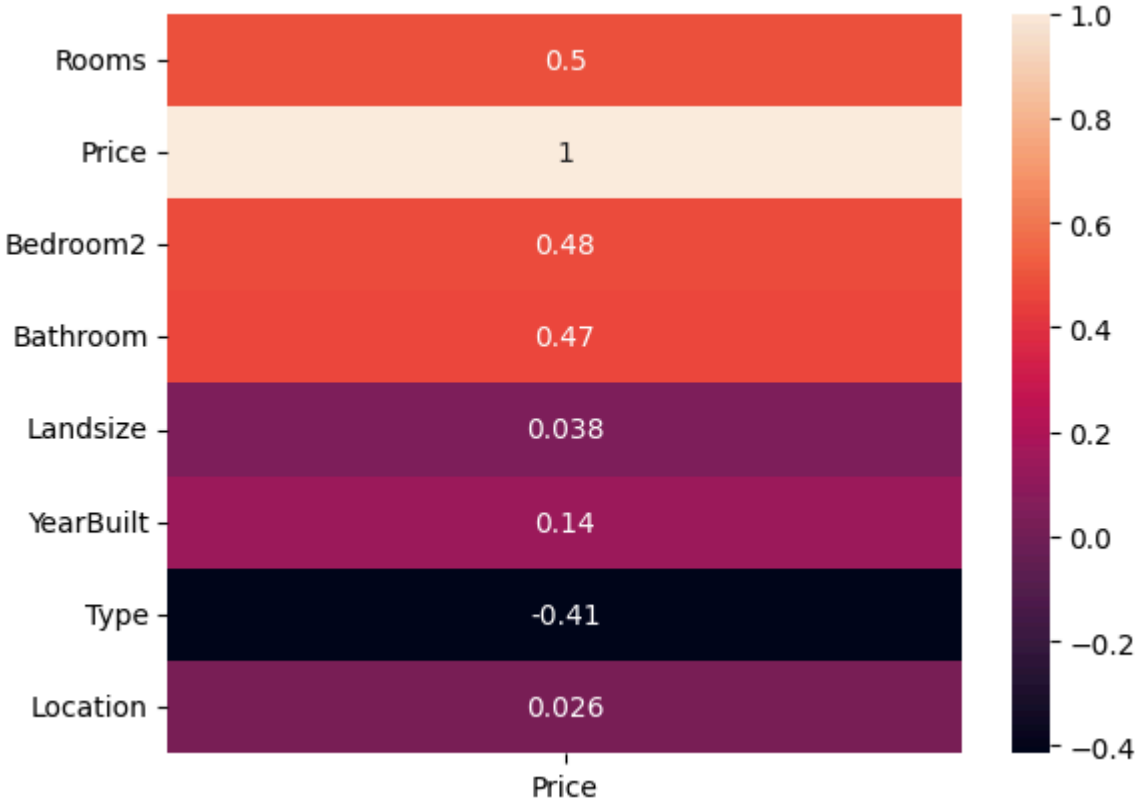
In [ ]:
```
le = LabelEncoder()
# We combine the Regionname and Suburb columns for approx. estimation of locatio
# Then we use LabelEncoder to convert the Location column to numerical data
df["Location"] = df["Regionname"] + " " + df["Suburb"]
df["Location"] = le.fit_transform(df["Location"])

# We also encode the house type column
df["Type"] = le.fit_transform(df["Type"])

# We drop the Address, Regionname and Suburb columns
df.drop(["Address","Regionname", "Suburb"], axis=1, inplace=True)


## To rank the year built column,as new house as top (expensive) (0-8)
def year_mapping(x):
    if x < 1800:
        return 8
    elif  1800 <= x <= 1850:
        return 7
    elif  1851 <= x <= 1900:
        return 6
    elif 1901 <= x <= 1920:
        return 5
    elif 1921 <= x <= 1950:
        return 4
    elif 1951 <= x <= 1980:
        return 3
    elif 1981 <= x <= 2000:
        return 2
    elif 2001 <= x <= 2023:
        return 1
    else :
        return 0

# To change Year built column to categorical data
df["YearBuilt"] = df["YearBuilt"].apply(year_mapping).fillna(0)


# Landsize : Used MinMax Scaler, Landsize is scaled from 0 to 1
min_max_scaler = MinMaxScaler()
df["Landsize"] = min_max_scaler.fit_transform(df[["Landsize"]])
df.head()
```

Out[ ]:

	Rooms	Price	Bedroom2	Bathroom	Landsize	YearBuilt	Type	Location
0	2	1480000	2	1	0.000466	0	0	70
1	2	1035000	2	1	0.000360	6	0	70
2	3	1465000	3	2	0.000309	6	0	70
3	3	850000	3	2	0.000217	0	0	70
4	4	1600000	3	1	0.000277	1	0	70

In [ ]:
```
correlation_matrix = df.corr()['Price']
sns.heatmap(correlation_matrix.to_frame(), annot=True)
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
sns.pairplot(df)
plt.show()
```

In [ ]:
```
# Split Dataset

x = df.drop(columns=['Price'])  # Features
y = df['Price']  # Target variable

# Split data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
```

XGBoost Trees

In [ ]:
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error

# Train XGBoost model
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.3,
                max_depth = 10, alpha = 10, n_estimators = 100)
xg_reg.fit(x_train, y_train)

# Predict on test set
y_pred = xg_reg.predict(x_test)

# Evaluate model
```

```
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("RMSE:", rmse)

# Feature importance analysis
xg_feature_importance = pd.DataFrame({'Feature': x.columns, 'Importance': xg_reg
print(xg_feature_importance)
```

```
RMSE: 328457.1624896741
     Feature  Importance
0      Rooms    0.362688
1   Bedroom2    0.064113
2   Bathroom    0.150244
3   Landsize    0.039193
4  YearBuilt    0.108463
5       Type    0.172203
6   Location    0.103097
```