In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import classification_report,confusion_matrix,root_mean_squ
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
data=fetch_california_housing()
```

In [3]:
```
df=pd.DataFrame(data.data,columns=data.feature_names)
```

In [4]:
```
df['Prices']=data.target
```

In [5]:
```
df
```

Out[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

20640 rows × 9 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   Prices      20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Prices        0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

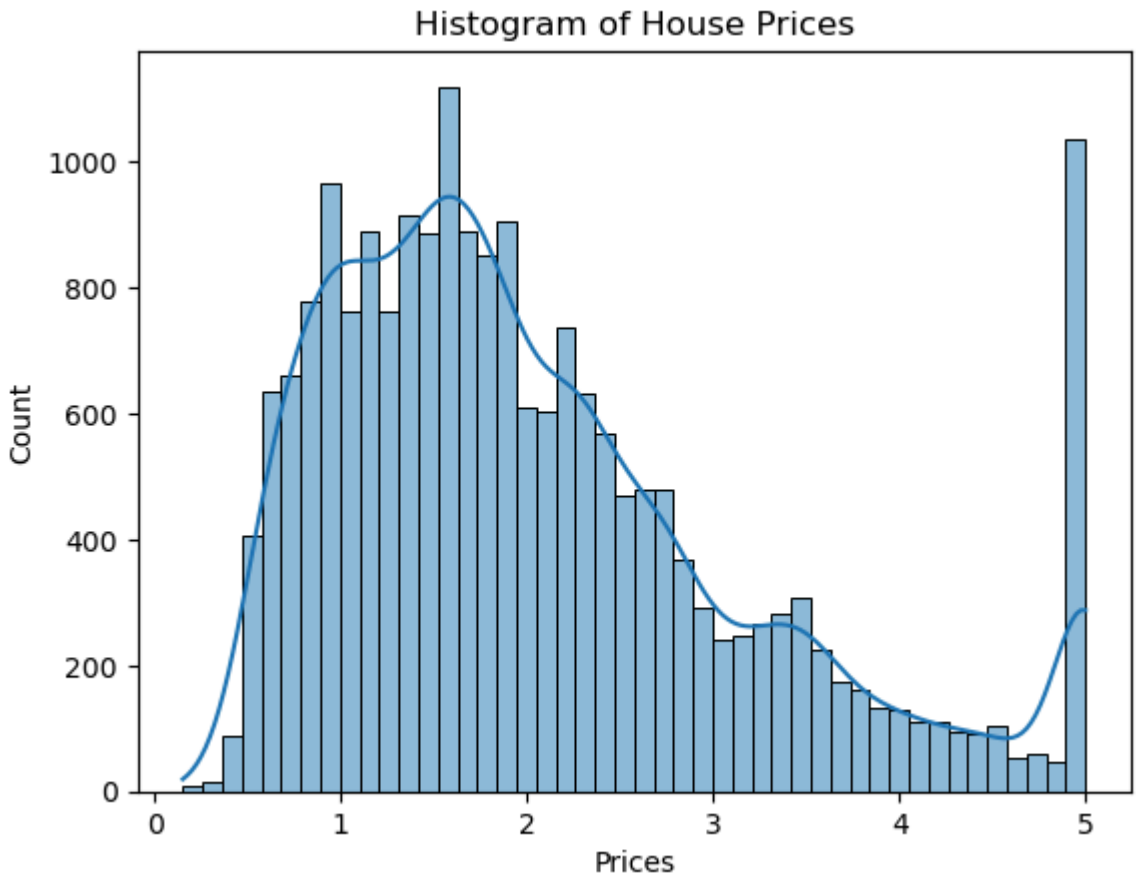
Out[8]:
```
0
```

In [9]:
```
df.describe()
```

Out[9]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOc
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333

In [10]:
```
sns.histplot(df,x=df['Prices'],kde=True)
plt.title('Histogram of House Prices')
plt.show()
```

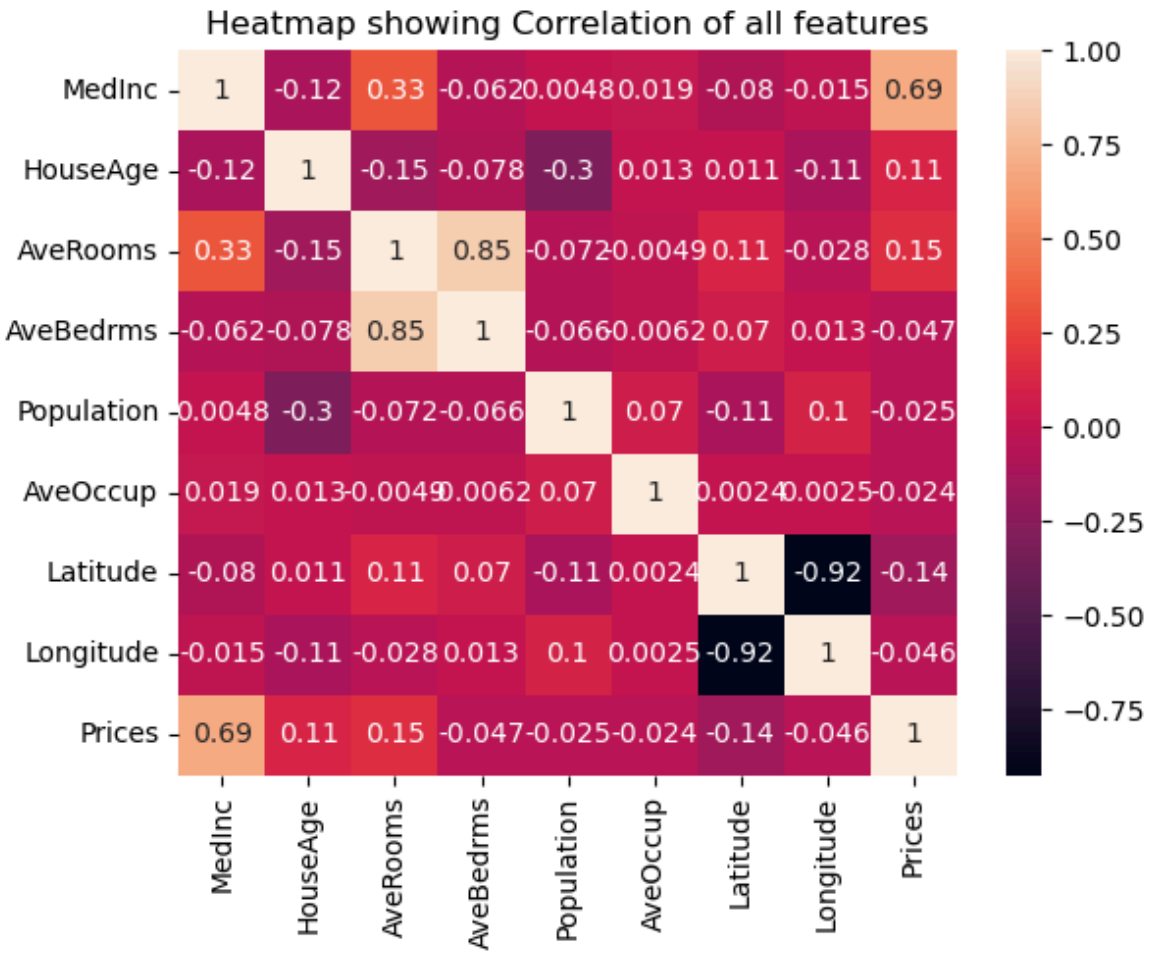


In [27]:
```
corr=df[['MedInc','HouseAge','AveRooms','AveBedrms','Population',
'AveOccup',
'Latitude',
'Longitude','Prices']].corr()
corr
```

Out[27]:

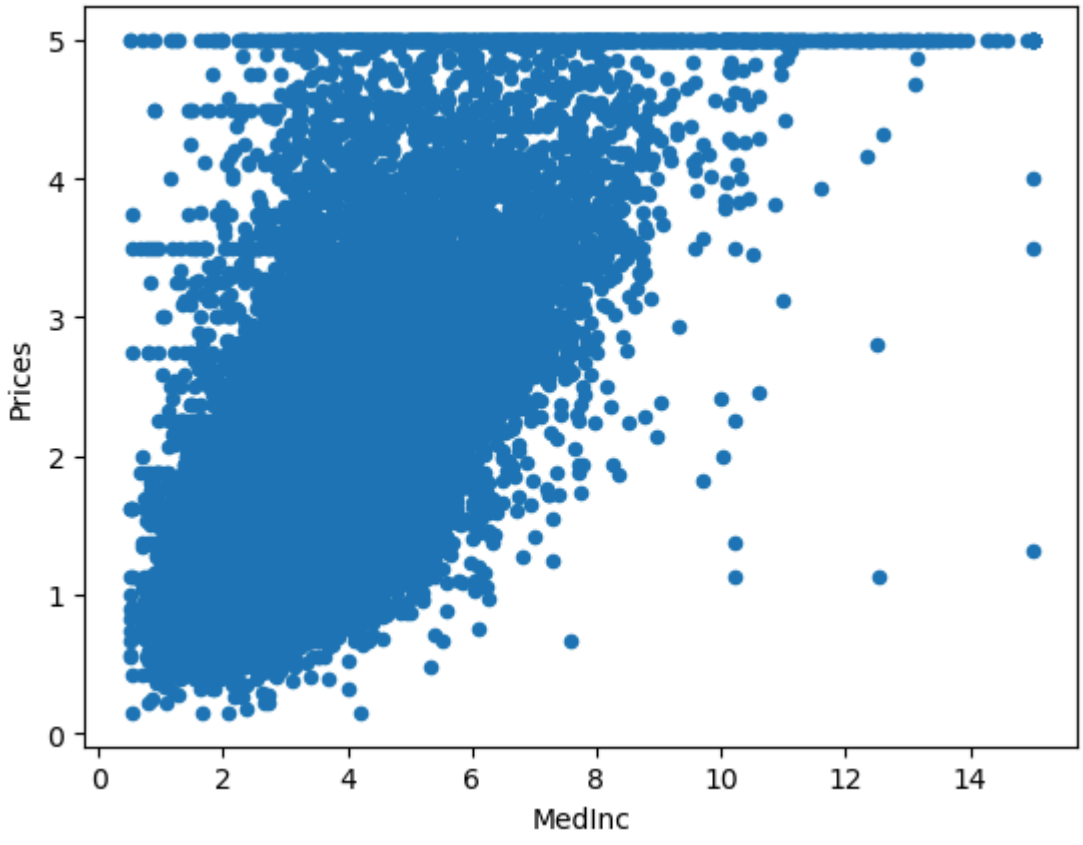
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Lati
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.07
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.01
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.10
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	0.06
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	-0.10
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	0.00
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	1.00
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	-0.92
Prices	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	-0.14

In [51]:
```
sns.heatmap(corr,annot=True)
plt.title("Heatmap showing Correlation of all features")
plt.show()
```

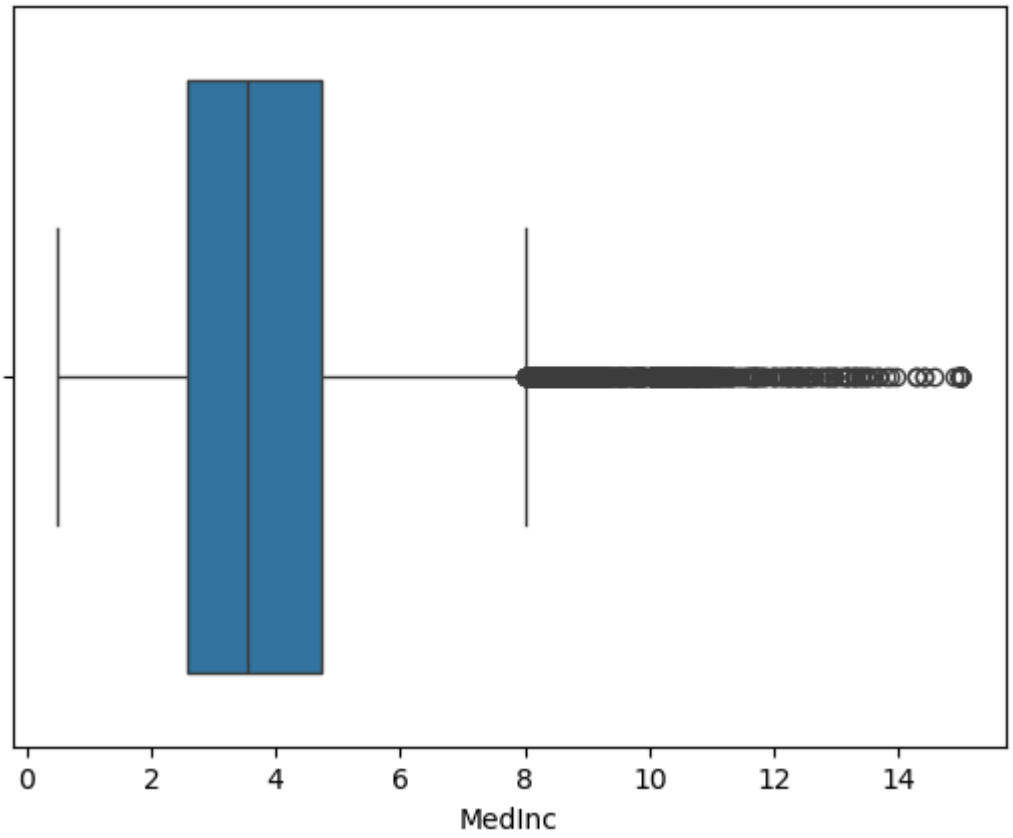


Feature of utmost importance:- 1.MedInc

In [16]:
```
df.plot(kind="scatter",x='MedInc',y='Prices')
plt.show()
```



In [45]:
```
sns.boxplot(data=df,x='MedInc')
plt.show()
```



In [47]:
```
X = df.drop(columns=['Prices'])
y = np.log1p(df['Prices'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
xg_model=XGBRegressor(n_estimators=150,learning_rate=0.4,max_depth=5)
xg_model.fit(X_train_scaled,y_train)
price_pred_xg=xg_model.predict(X_test_scaled)
```

In [33]:
```
print("R2 value:",r2_score(y_test,price_pred_xg))
print('Root Mean squared error:',root_mean_squared_error(y_test,price_pred_xg))
```

```
R2 value: 0.8430133277994334
Root Mean squared error: 0.13980306039411947
```

In [35]:
```
Mean_hosue_price=df['Prices'].mean()
print("Mean_hosue_price (millions):",Mean_hosue_price)
```

```
Mean_hosue_price (millions): 2.068558169089147
```

In [37]:
```
error=root_mean_squared_error(y_test,price_pred_xg)/Mean_hosue_price
print("Relative Error:",round(error*100,2),"%")
```

```
Relative Error: 6.76 %
```

Insights and Conclusion: Dataset Overview The dataset used in this project is the California Housing dataset, sourced from the sklearn.datasets module. It contains 20,640 observations with the following 8 numerical features: Feature Description MedInc Median income in block HouseAge Median age of houses AveRooms Average number of rooms per household AveBedrms Average number of bedrooms per household Population Block population AveOccup Average household occupancy Latitude

Latitude coordinate Longitude Longitude coordinate The target variable is the median house price in each block, which right-skewed when visualised by histogram thereby making it a strong candidate for log transformation. Project Summary Applied a log transformation (log_prices = np.log1p(prices)) to stabilize variance and reduce skew in the target variable. Scaled features using StandardScaler after train-test split to avoid data leakage. Trained an XGBRegressor with optimized hyperparameters: max_depth=5, learning_rate=0.4, n_estimators=150 Evaluated the model on both log-transformed and original price scale using np.expm1(). Model Performance: R2 value: 0.8430133277994334 Root Mean squared error: 0.13980306039411947 Relative Error: 6.76 % A 6.76% prediction error makes the model highly reliable for automated home valuation, real estate pricing tools, and investment analysis platforms. Demonstrates how proper preprocessing and thoughtful feature engineering can significantly boost model performance.