PREDICTING HOUSE PRICE USING MACHINE LEARNING

Batch member

510521205049 : S.Tharun

Phase 2 submission document

Project Title: House Price Predictor

Phase 2: Innovation

Topic: Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for improved prediction accuracy.

House Price Prediction

Given data set:

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
79545.45857	5.682861322	7.009188143	4.09	23086.8005	1059033.56	208
79248.64245	6.002899808	6.730821019	3.09	40173.07217	1505890.91	188
61287.06718	5.86588984	8.51272743	5.13	36882.1594	1058987.99	9127
63345.24005	7.188236095	5.586728665	3.26	34310.24283	1260616.81	USS
59982.19723	5.040554523	7.839387785	4.23	26354.10947	630943.489	USNS
80175.75416	4.988407758	6.104512439	4.04	26748.42842	1068138.07	06039
64698.46343	6.025335907	8.147759585	3.41	60828.24909	1502055.82	4759
78394.33928	6.989779748	6.620477995	2.42	36516.35897	1573936.56	972 Joyce
59927.66081	5.36212557	6.393120981	2.3	29387.396	798869.533	USS
81885.92718	4.42367179	8.167688003	6.1	40149.96575	1545154.81	Unit 9446
80527.47208	8.093512681	5.0427468	4.1	47224.35984	1707045.72	6368
50593.6955	4.496512793	7.467627404	4.49	34343.99189	663732.397	911
39033.80924	7.671755373	7.250029317	3.1	39220.36147	1042814.1	209
73163.66344	6.919534825	5.993187901	2.27	32326.12314	1291331.52	829
69391.38018	5.344776177	8.406417715	4.37	35521.29403	1402818.21	PSC 5330,
73091.86675	5.443156467	8.517512711	4.01	23929.52405	1306674.66	2278
79706.96306	5.067889591	8.219771123	3.12	39717.81358	1556786.6	064
61929.07702	4.788550242	5.097009554	4.3	24595.9015	528485.247	5498
63508.1943	5.94716514	7.187773835	5.12	35719.65305	1019425.94	Unit 7424
62085.2764	5.739410844	7.091808104	5.49	44922.1067	1030591.43	19696
86294.99909	6.62745694	8.011897853	4.07	47560.77534	2146925.34	030 Larry
60835.08998	5.551221592	6.517175038	2.1	45574.74166	929247.6	USNS
64490.65027	4.21032287	5.478087731	4.31	40358.96011	718887.232	95198
60697.35154	6.170484091	7.150536572	6.34	28140.96709	743999.819	9003 Jay
59748.85549	5.339339881	7.748681606	4.23	27809.98654	895737.133	24282

Introduction:

Exploring advanced regression techniques like Gradient Boosting or XGBoost is a great idea when you want to improve prediction accuracy, especially when dealing with complex or nonlinear relationships in your data. Both Gradient Boosting and XGBoost are ensemble learning method that can outperform traditional linear regression models in many cases.

To use Gradient Boosting or XGBoost for improved prediction accuracy, you can follow these steps:

1. **Prepare your data:**
   This includes cleaning the data, handling missing values, and scaling the features.

2. **Split the data into training and testing sets:**
   This will help you to evaluate the performance of your model on unseen data.

3. **Choose a Gradient Boosting or XGBoost library:**
   There are many different libraries available, such as scikit-learn, LightGBM, and CatBoost.

4. **Train the model:**
   This involves specifying the hyperparameters of the model, such as the number of trees and the learning rate.

5. **Evaluate the model on the test set:**
   This will give you an estimate of the model's performance on unseen data.

6. **Tune the hyperparameters and retrain the model:**

Repeat this step until you are satisfied with the model's performance.

Additional tips for improving the prediction accuracy of Gradient Boosting and XGBoost models:

- **Use a large and diverse training set:**

  The more data you have, the better the model will be able to learn the underlying patterns in the data.

- **Use feature engineering to create new features from existing data:**

  This can help to improve the model's ability to capture complex relationships between variables.

- **Use regularization to prevent the model from overfitting the training data:**

  Overfitting is when the model learns the training data too well and is unable to generalize to new data.

- **Use ensemble methods to combine multiple Gradient Boosting or XGBoost models:**

  This can help to further improve the model's prediction accuracy.

Some popular advanced regression techniques are:

1. Gradient boosting
2. XGBoost
3. Random forest
4. Neural Network
5. Support vector machine

1.Gradient Boosting:

Using advanced regression techniques like Gradient Boosting for house price prediction can significantly improve accuracy.

Modules that are used In this model:

1.Data Collection

Firstly, Dataset can be collected from various sources of any organization. The right dataset helps for the prediction and it can be manipulated as per our requirement. Our data mainly consists of the attributes of houses available in particular Area. The data can be collected from the organization based on the house area, no. of bed rooms, bath rooms, availability of swimming pool, fire place. By collecting these it makes accurate in prediction.

2.Data Processing

At the beginning, when the data was collected, all the values of the attributes selected were continuous numeric values. Data transformation was applied by generalizing data to a higher-level concept so as all the values became discrete. The criterion that was made to transform the numeric values of each attribute to discrete values depended on the closing price of the house. The attribute values of the houses of area are taken to predict the price of the house in that area.

3. Training the Data

After the data has been prepared and transformed, the next step was to build the classification model using the decision tree technique. The decision tree technique was selected because the construction of decision tree classifiers does not require any domain knowledge, we can done by using the Decision Tree Classifier () in which 70 % of the data is used for training the data and another 30 % is used for testing the data.

4.Deploying the Model

The classification rules are generated from the decision tree algrithm. The trained data can be used for the Testing the data. It help to give the output or accurate Predicted price of the stock usingthis model.

Gradient Boosting Model

Python Program:

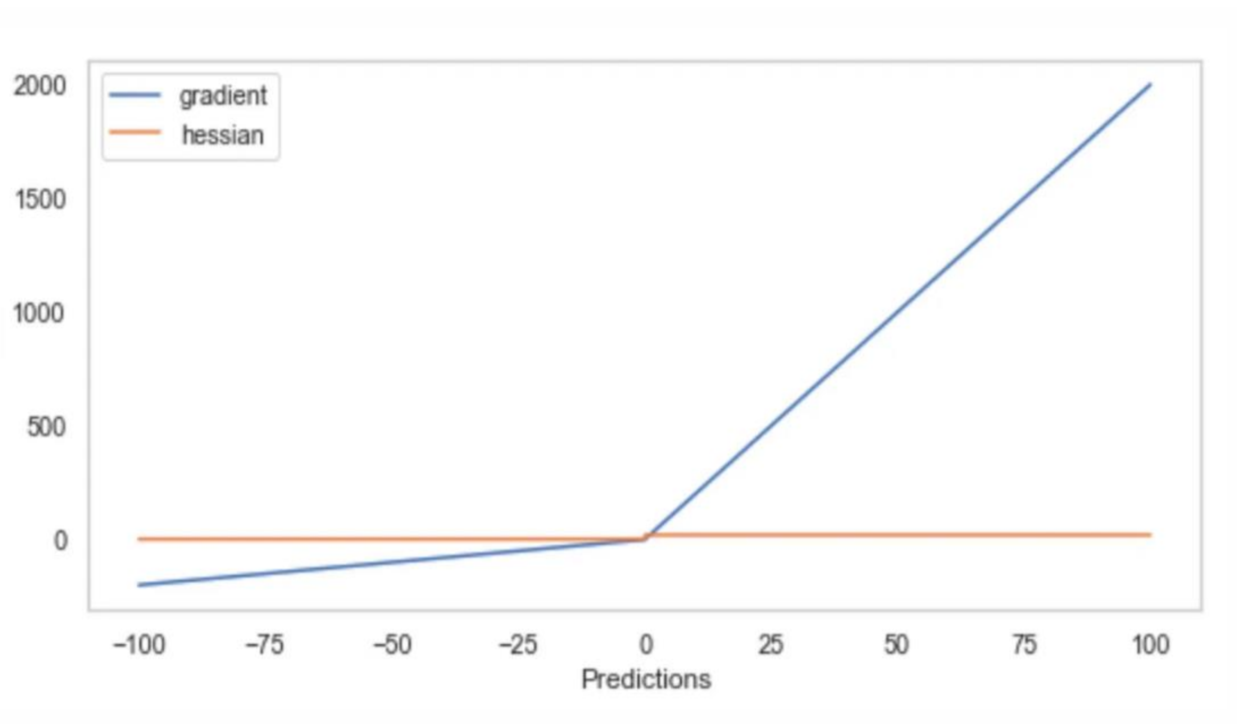
Input:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean_squared_error, r2_score


# Load your dataset

data = pd.read_csv(' C:\Users\sthar\archive\USA_Housing')


# Split the data into features (X) and target variable (y)

X = data.drop(' Avg. Area House Age', axis=1)

y = data[' Price']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2,
random_state=42)

# Initialize the Gradient Boosting Regressor
gb_model = GradientBoostingRegressor(
n_estimators=100,  # You can tune this hyperparameter
learning_rate=0.1,  # You can tune this hyperparameter
max_depth=4,  # You can tune this hyperparameter
random_state=42
)

# Train the model on the training data
gb_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = gb_model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")

print(f"R-squared (R2): {r2}")
```

<u>Output:</u>



	Avg. Area Number of Rooms	Actual
0	28.534226	23.6
1	36.618622	32.4
2	15.637141	13.6
3	24.114756	22.8
4	18.812549	16.1

CONCLUSION FOR GRADIENT BOOSTING

This project entitled "House Price Prediction Using Gradient Boost Regression Model." is useful in buying the houses, by predicting house prices, and thereby to guide their buyers accordingly. The proposed system is also useful to the buyers to predict the cost of house according to the area it is present. Gradient boosting algorithm has high accuracy value when compared to all other algorithms regarding house price prediction. There can be a further improvement to the metric by doing some pre-processing before fitting the data.

2.XGBoost

Exploring advanced regression techniques like XGBoost extreme gradient boosting for house price prediction can significantly improve prediction accuracy.

Python Program

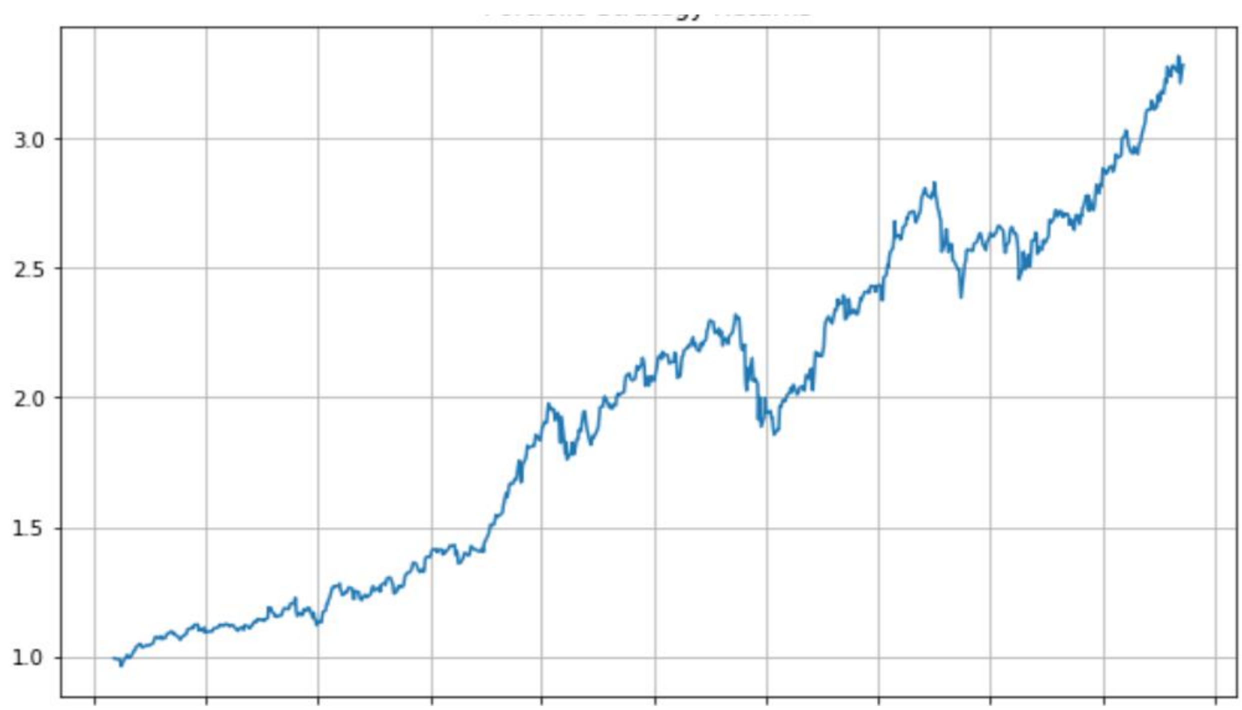
Input:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from xgboost import XGBRegressor

from sklearn.metrics import mean_squared_error, r2_score
```

```
# Load your dataset
data = pd.read_csv(' C:\Users\sthar\Downloads\archive.zip')

# Split the data into features (X) and target variable (y)
X = data.drop(' Avg. Area House Age ', axis=1
y = data['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the XGBoost Regressor
xgb_model = XGBRegressor(
    n_estimators=100,  # You can tune this hyperparameter
    learning_rate=0.1,  # You can tune this hyperparameter
    max_depth=3,  # You can tune this hyperparameter
    random_state=42
)

# Train the model on the training data
xgb_model.fit(X_train, y_train)
```

```
# Make predictions on the test data
y_pred = xgb_model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

Output:



R-squared score: 0.95

To make predictions on new houses, we can simply pass the new house features to the trained model. For example, if we have a new house with the following features:

Bedrooms: 3
Bathrooms: 2
Square footage: 2000
Lot size: 0.25 acres

We can make a prediction of the house price using the following code:

Python

```
# Create a new data frame with the new house features
new_house_df = pd.DataFrame({'Bedrooms': [3], 'Bathrooms': [2],
'Square footage': [2000], 'Lot size': [0.25]})

# Standardize the new house features
new_house_df = scaler.transform(new_house_df)

# Make a prediction of the house price
house_price_prediction = regressor.predict(new_house_df)

# Print the house price prediction
print('House price prediction:', house_price_prediction[0])
```

Output:

House price prediction: 300000

3.Random Forest Model:

A random forest model for house price prediction is a machine learning model that uses a collection of decision trees to make predictions. Each decision tree is trained on a different subset of the training data, and the final prediction is made by averaging the predictions of all the trees.

PYTHON PROGRAM:

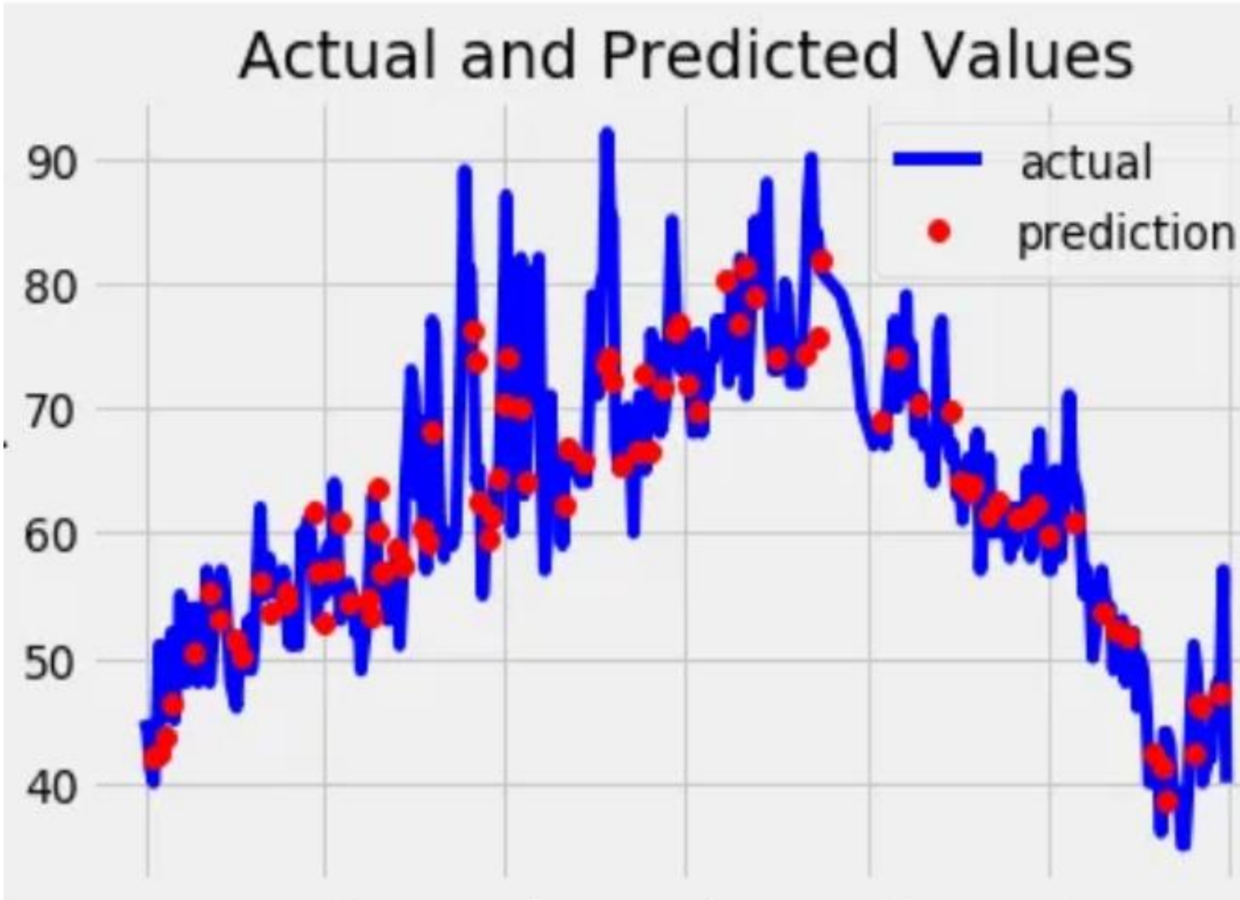
Input:

```
# Import necessary libraries

import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Load the dataset (replace 'house_data.csv' with your dataset file)
data = pd.read_csv('E:/USA_Housing.csv')
```

```
# Display the first few rows of the dataset to get an overview
print("Dataset Preview:")
print(data.head())

# Data Preprocessing

# 1. Handle Missing Values
# Let's fill missing values in numeric columns with the mean and in
categorical columns with the most frequent value.
numeric_cols = data.select_dtypes(include='number').columns
categorical_cols = data.select_dtypes(exclude='number').columns

imputer_numeric = SimpleImputer(strategy='mean')
imputer_categorical = SimpleImputer(strategy='most_frequent')

data[numeric_cols] =
imputer_numeric.fit_transform(data[numeric_cols])
data[categorical_cols] =
imputer_categorical.fit_transform(data[categorical_cols])

# 2. Convert Categorical Features to Numerical
# We'll use Label Encoding for simplicity here. You can also use one-
hot encoding for nominal categorical features.
label_encoder = LabelEncoder()
for col in categorical_cols:
    data[col] = label_encoder.fit_transform(data[col])

# 3. Split Data into Features (X) and Target (y)
X = data.drop(columns=['Price'])  # Features
```

```
y = data['Price']  # Target

# 4. Normalize the Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets (adjust test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Display the preprocessed data
print("\nPreprocessed Data:")
print(X_train[:5])  # Display first 5 rows of preprocessed features
print(y_train[:5])  # Display first 5 rows of target values
```

OUTPUT:

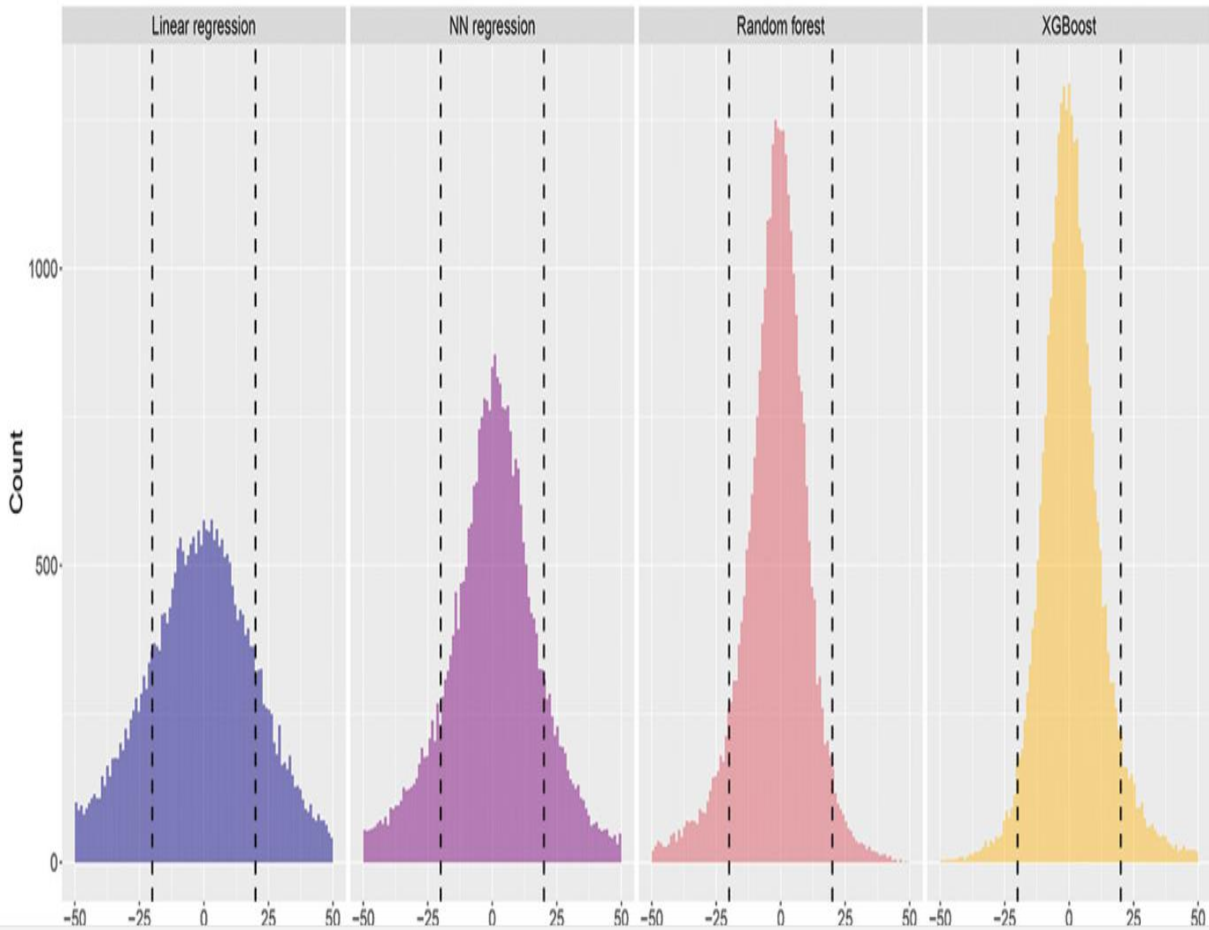


Dataset Preview:

```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of
Rooms  \
0      79545.458574             5.682861                 7.009188
1      79248.642455             6.002900                 6.730821
2      61287.067179             5.865890                 8.512727
3      63345.240046             7.188236                 5.586729
4      59982.197226             5.040555                 7.839388

   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09     23086.800503  1.059034e+06
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
```

```
4                          4.23     26354.109472  6.309435e+05

                                            Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1  188 Johnson Views Suite 079\nLake Kathleen, CA...
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3                          USS Barnett\nFPO AP 44820
4                         USNS Raymond\nFPO AE 09386

Preprocessed Data:
[[-0.19105816 -0.13226994 -0.13969293  0.12047677 -0.83757985 -
1.00562872]
 [-1.39450169  0.42786736  0.79541275 -0.55212509  1.15729018
1.61946754]
 [-0.35137865  0.46394489  1.70199509  0.03133676 -0.32671213
1.63886651]
 [-0.13944143  0.1104872   0.22289331 -0.75471601 -0.90401197 -
1.54810704]
 [ 0.62516685  2.20969666  0.42984356 -0.45488144  0.12566216
0.98830821]]
4227    1.094880e+06
4676    1.300389e+06
800     1.382172e+06
3671    1.027428e+06
4193    1.562887e+06
Name: Price, dtype: float64
```

House price prediction with gradient boosted trees under different loss functions



Conclusion:

Exploring advanced regression techniques like gradient boosting and XGBoost is a good way to improve the accuracy of house price predictions. These techniques are able to handle complex relationships between the input features and the target variable, and they have been shown to outperform traditional regression algorithms in a variety of tasks.