**Innovation in House Price Prediction using Gradient Boosting:**

**Problem Statement:**

In today's dynamic real estate market, making informed decisions about buying or selling a house is paramount. Accurate house price prediction empowers individuals and organizations in this critical financial endeavor. Leveraging innovative machine learning techniques, we aim to revolutionize house price prediction by developing a highly accurate and reliable model.

**Design thinking process:**

To innovate in house price prediction using Gradient Boosting, follow a design thinking process. Start by empathizing with users' needs and challenges. Define the problem and set clear success metrics. Ideate various solutions, prioritize, and create a prototype with a user-friendly interface. Test the prototype with users, gather feedback, and iterate based on their input. Implement and launch the improved system, ensuring scalability and reliability. Continuously monitor performance and gather user feedback for further improvements, staying open to new ideas and technologies to enhance the prediction model. This user-centered approach ensures a more effective and satisfying solution.

**Phases of development:**

The development of an innovative house price prediction system using Gradient Boosting can be broken down into several phases:

1. Project Planning and Scoping:

- Define the project's objectives, scope, and success criteria.

- Identify stakeholders and create a project plan with timelines and resource allocation.

2. Data Collection and Preprocessing:

- Gather relevant real estate data, such as property features, historical prices, and market trends.

- Clean, preprocess, and transform the data to make it suitable for machine learning, addressing missing values and outliers.

3. Feature Engineering:

- Select and engineer relevant features that can improve prediction accuracy, such as property size, location, historical sales data, and economic indicators.

4. Model Selection:

- Choose the appropriate Gradient Boosting algorithm (e.g., XGBoost, LightGBM, or CatBoost) for the prediction task based on the dataset and project goals.

5. Model Training:

- Split the data into training and validation sets.

- Train the Gradient Boosting model using the training data, optimizing hyperparameters and reducing overfitting.

6. Evaluation and Validation:

- Assess the model's performance using validation data, measuring metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared.

- Validate the model's generalization capability to ensure it works well with unseen data.

7. Feature Importance Analysis:

- Analyze feature importance scores to understand which attributes have the most significant impact on price predictions.

8. User Interface Development:

- Create a user-friendly interface or application for users to input property details and receive price predictions.

9. Integration and Deployment:

- Integrate the trained model into the user interface, ensuring it can make real-time predictions.

- Deploy the system on a scalable and reliable platform, such as a web server or cloud infrastructure.

10. User Testing:

- Allow users to interact with the system and provide feedback on its usability and accuracy.

11. Fine-tuning and Optimization:

- Refine the model and user interface based on user feedback and further optimization, if needed.

12. Security and Privacy Considerations:

- Implement security measures to protect user data and privacy, especially if sensitive information is involved.

13. Documentation and Training:

- Provide documentation for users and support personnel on how to use and maintain the system.

- Offer training if necessary to ensure users can maximize the benefits.

14. Monitoring and Maintenance:

- Continuously monitor the system's performance, addressing any issues or drift in prediction accuracy.

- Regularly update the model with new data to adapt to changing market conditions.

15. Feedback and Iteration:

- Gather feedback from users and stakeholders to drive further improvements and innovations in the house price prediction system.

These development phases ensure a comprehensive and well-structured approach to building an innovative house price prediction system using Gradient Boosting, focusing on accuracy, usability, and adaptability.

**Dataset used = house\_price\_data.csv**

**Data preprocessing:**

Data preprocessing is the process of cleaning our data set. There might be missing values or outliers in the dataset. These can be handled by data cleaning. If there are many missing values in a variable we will drop those values or substitute it with the average value.

**Training the model:**

Since the data is broken down into two modules: a Training set and Testset, we must initially train the model. The training set includes the target variable. The decision tree regressor algorithm is applied to the training data set. The Decision tree builds a regression model in the form of a tree structure.

**Testing and Integrating with UI:**

The trained model is applied to test dataset and house prices are predicted. The trained model is then integrated with the frontend using Flask in python

**REGRESSION ALGORITHM:**

1. Import the python libraries that are required for house price prediction using linear regression. Example: numpy is used for convention of data to 2d or 3d array format which is required for linearregression model ,matplotlib for plotting the graph , pandas for readingthe data from source and manipulation that data, etc.

2. First Get the value from source and give it to a data frame and thenmanipulate this data to required form using head(),indexing, drop().

3. Next we have to train a model, its always best to spilt the data intotraining data and test data for modelling.

4. Its always good to use shape() to avoid null spaces which will cause error during modelling process.

5. Its good to normalize the value since the values are in very large quantity for house prices , for this we may use minmaxscaler to reducethe gap between prices so that its easy and less time consuming for comparing and values.range usually specified is between 0 to 1 using fittransform.

6. Then we have to make few imports from keras: like sequential for initializing the network,lstm to add lstm layer, dropout to prevent overfitting of lstm layers, dense to add a densely connected networklayer for output unit.

7. In lstm layer declaration its best to declare the unit, activiation,returnsequence.

8. To compile this model its always best to use adam optimizer and set the loss as required for the specific data.

9. We can fit the model to run for a number of epochs. Epochs are the number of times the learning algorithm will work through the entire training set. 24

10. Then we convert the values back to normal form by using inverse minimal scale by scale factor. 11. Then we give a test data(present data)to the trained model to get the predicted value(future data).

12. Then we can use matplotlib to plot a graph comparing the test andpredicted value to see the increase/decrease rate of values in each time of the year in a particular place. Based on this people will know when its best time to sell or buy a place in a given location.

**Evaluation metrics:**

When evaluating the performance of an alternative regressor like XGBoost Regressor for tasks such as house price prediction, you can use various evaluation metrics to assess how well the model performs. Some commonly used evaluation metrics for regression tasks include:

1. Mean Absolute Error (MAE):

- MAE calculates the average absolute difference between the actual and predicted values. It is easy to interpret, with lower values indicating better performance.

2. Mean Squared Error (MSE):

- MSE calculates the average squared difference between actual and predicted values. It penalizes larger errors more than MAE, making it sensitive to outliers. A lower MSE indicates better performance.

3. Root Mean Squared Error (RMSE):

- RMSE is the square root of MSE, providing an interpretable metric in the same units as the target variable. Smaller RMSE values are better.

4. R-squared (R^2):

- R-squared measures the proportion of the variance in the dependent variable that is explained by the model. A higher R-squared value (closer to 1) indicates a better fit of the model to the data.

5. Mean Absolute Percentage Error (MAPE):

- MAPE expresses the prediction error as a percentage of the actual values. It is useful when you want to understand the relative size of errors. Lower MAPE values indicate better accuracy.

6. Coefficient of Determination (adjusted R-squared):

- Adjusted R-squared is a modified version of R-squared that accounts for the number of predictors in the model. It helps prevent overfitting by penalizing excessive complexity in the model.

7. Median Absolute Error (MedAE):

- MedAE calculates the median of the absolute errors. It is less sensitive to outliers compared to MAE.

8. Explained Variance Score:

- This metric indicates the proportion of variance in the target variable explained by the model. It ranges from 0 to 1, with higher values indicating better explanatory power.

9. Poisson Deviance:

- Poisson Deviance is useful when dealing with count data or non-negative values. It measures the goodness of fit for Poisson regression models.

10. Quantile Loss (e.g., Pinball Loss):

- This metric is used to assess how well the model performs at specific quantiles of the target variable distribution. It's valuable when predicting conditional quantiles.

The choice of evaluation metric depends on the specific objectives and characteristics of your regression problem. Some metrics may be more appropriate than others, and it's common to use a combination of metrics to get a comprehensive view of model performance.

**SOURCE CODE:**

import pandas as pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt

%matplotlib inline

HouseDF = pd.read\_csv('USA\_Housing.csv')

HouseDF.head()

HouseDF=HouseDF.reset\_index()

HouseDF.head()

HouseDF.info()

HouseDF.describe() HouseDF.columns sns.pairplot(HouseDF) sns.distplot(HouseDF['Price’]) sns.heatmap(HouseDF.corr(), annot=True)

X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']] y = HouseDF['Price’] from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=101)

from sklearn.linear\_model import minmaxscaler lm = minmaxscaler(feature\_range=(0,1)) lm.fit\_transform(X\_train,y\_train) print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient’]) coeff\_df

from keras.layers import Dense,Dropout,LSTM from keras.models import Sequential model = Sequential() model.add(LSTM(units = 50,activation = 'relu',return\_sequences = True,input\_shape =

(x\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units = 60,activation = 'relu',return\_sequences = True)) model.add(Dropout(0.3))

model.add(LSTM(units = 80,activation = 'relu',return\_sequences = True)) model.add(Dropout(0.4))

model.add(LSTM(units = 120,activation = 'relu')) model.add(Dropout(0.5))

model.add(Dense(units = 1))

model.compile(optimizer='adam', loss = 'mean\_squared\_error’) model.fit(x\_train, y\_train,epochs=50)

print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient’]) coeff\_df

predictions = lm.predict(X\_test)

scale\_factor = 1/0.02099517 y\_predicted = y\_predicted \* scale\_factor y y\_test = y\_test \* scale\_factor

plt.scatter(y\_test,predictions)

sns.distplot((y\_test-predictions),bins=50);

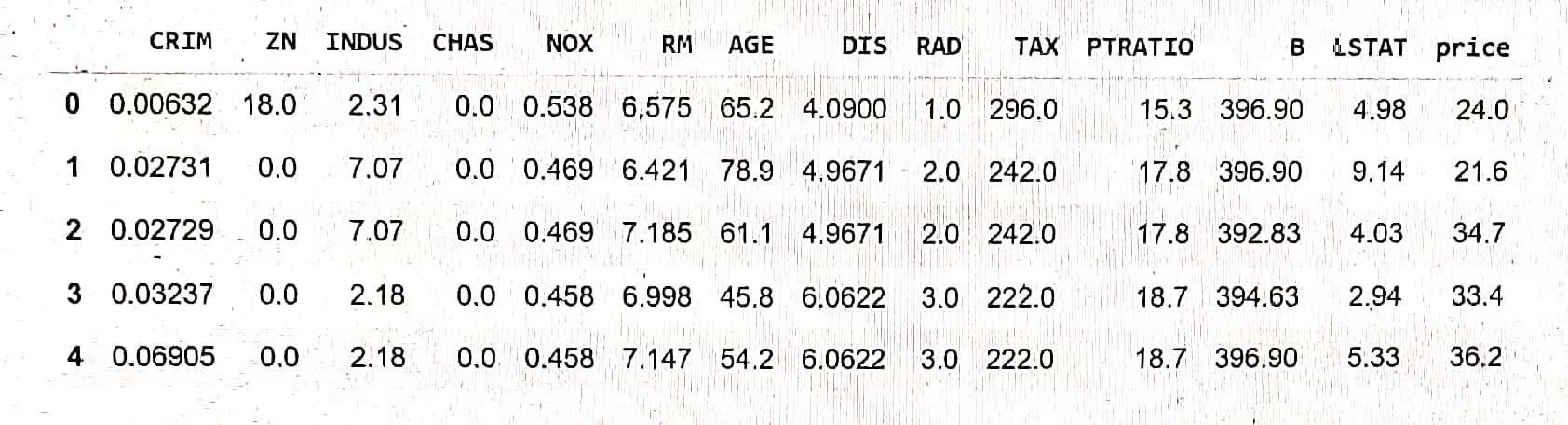
plt.figure(figsize=(12,6)) plt.plot(y\_test,'b',label = 'Original Price') plt.plot(y\_predicted,'r',label = 'Predicted Price') plt.xlabel('Time') plt.ylabel('Price') plt.legend()

plt.show()

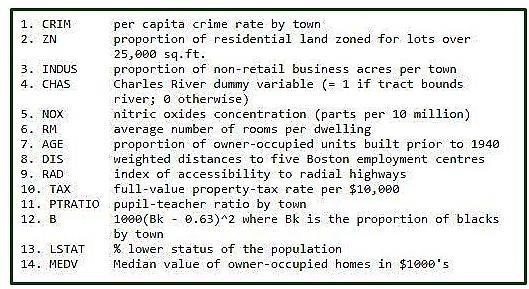
from sklearn import metrics

print('MAE:', metrics.mean\_absolute\_error(y\_test, predictions)) print('MSE:', metrics.mean\_squared\_error(y\_test, predictions)) print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, predictions)))

**EXPLANATION OF THE OUTPUT RESULTS AND THE DATASET:**

 First we import a sample data from sklearn library , you can get different types of sample data from Kaggle. The data taken here is the data of various parameters and the house prices in a given city called boston in the year between 1970 to 2020.

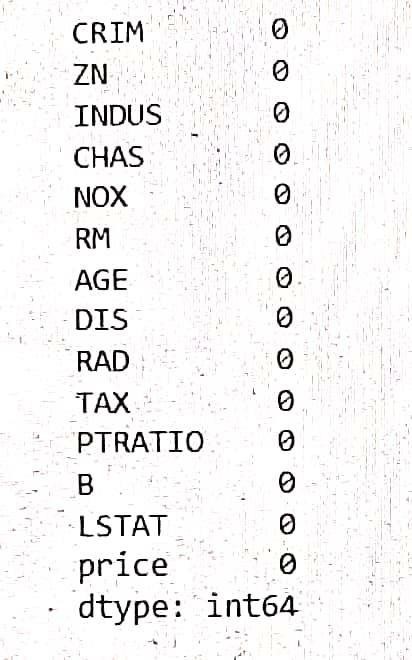
Here the data parameters are explained as follows:



Here for understanding purpose we have taken first 5 index/instance of data and printed them. In total there are 506 rows of data from the dataset , of which we have printed first 5 rows using head() function. There are 14 columns in total, i.e, 13 colums containing data of the place, and the 14th column is the target column which contains the house prices.

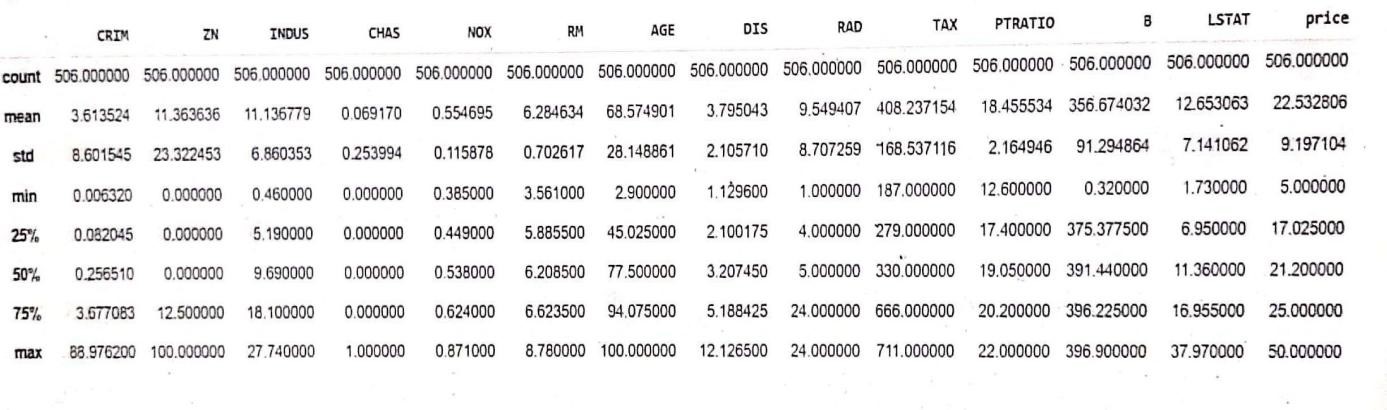
18

Then we check if our data has some null values i.e missing values. Since if the data is incomplete , then there will be error during processing state which may lead to loss of accuracy in predicting model. Here in our given data , there is no missing value as we can see.



Since our data contains no missing value, the program will skip the dropping phase in data processing, where data is dropped to increase accuracy and fit missing values in a way so that it is suitable for modelling.

Next we try to describe the data in such a way so that both people and machine find it easy to understand the given data . In order to do thiswe use the describe() function.



19

Counts refers to the number of instances of data in each column i.e 506 since there are 506 rows of data for each column Mean refers to mean value of data in given colum.

Std means the standard value i.e the most common value in given set of data for a particular column.

Min refers the least data value in each column.

Max refers to the maximum data value in each column.

25% refers that 25 percentile of the data in that column is equal to or below that value.

Next we try to understand the correlation between the different values, in order to do that, the best way is by using heat map. Heat map is a representation of data in the form of a map or diagram in which data values are represented as colours.

**Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate)**

There are two types of correlation, they are:

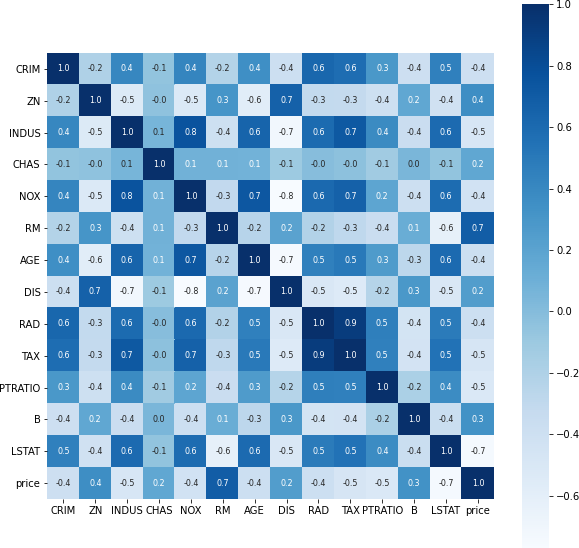
1. Positive correlation: A positive correlation is a relationship between two variables that move in tandem—that is, in the same direction. A positive correlation exists when one variable decreases as the other variable decreases, or one variable increases while the other increases.

1. Negative correlation: Negative correlation is a relationship between two variables in which one variable increases as the other decreases, and vice versa.

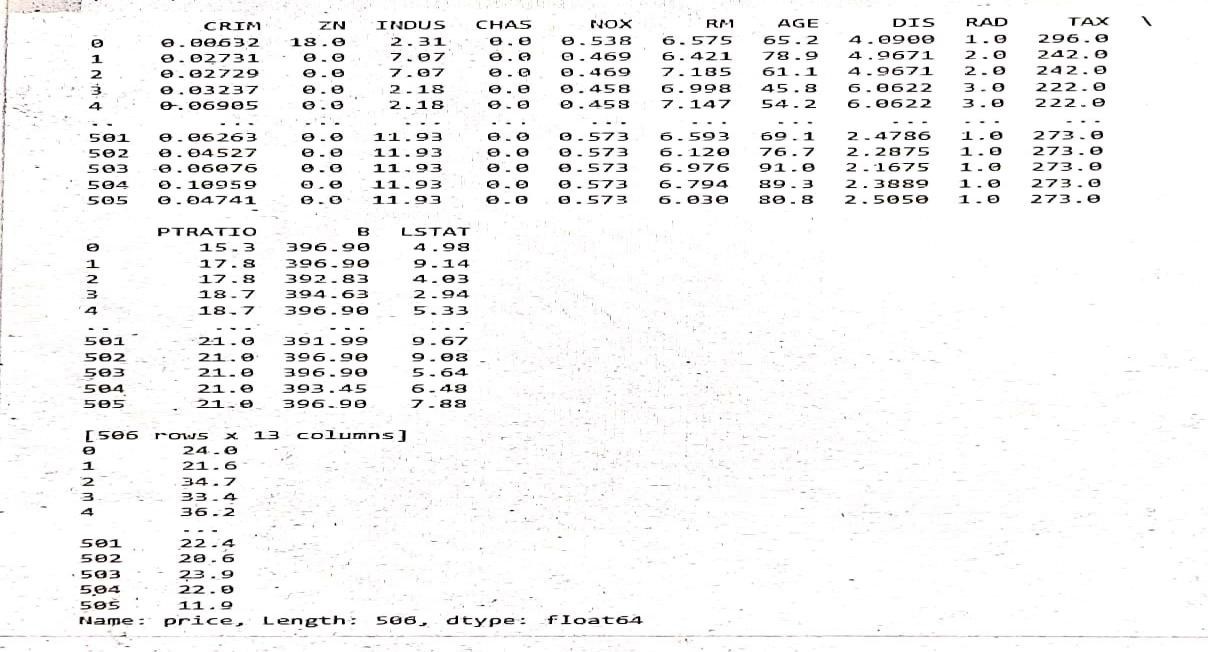
In statistics, a perfect negative correlation is represented by the value -1.0, while a 0 indicates no correlation, and +1.0 indicates a perfect positive correlation. A perfect negative correlation means the relationship that exists between two variables is exactly opposite all of the time. These are two types of correlation are represented numerically and as well as by shade of colour in the heat map.

20

HEATMAP – for better understanding of which place is best suited for individual personal preference based on given dataset. This uses correlation concept



Next we split our data into variables x and y , in order to train our model to predict data.



21

Here the varible x contains the value of the first 13 columns i.e the parameters that are required for calculating and predicting the house prices. The varible y contains the 14th column values which are the house prices.

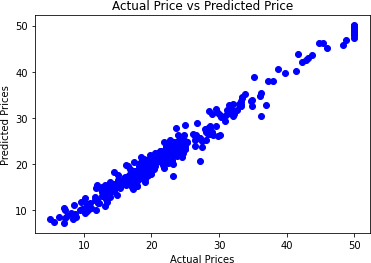
First we predict the values in y using the values in x . Then we compare the actual prices and predicted prices by using scatter plot. Then we find the r square error and mean square error between them . If the errors is less enough then we proceed for testing of the model since the training phase is over. If the error is large , then we use optimizers like adam, and repeat drop and fitting process for a set number of epochs to reduce the error.

The r square error or mean square error for good accuracy of the model in predicting the data is indicated numerically also.

A model is good if these error values are less then 5.

Then during testing process we predict the future house prices using present and past data parameters of houses in an location. Then we plot this graphically as a house price over time graph.

For training the model , the error needs to be minimum for greater accuracy of model. The error between the actual and predicted price is plotted graphically using scatter plot. Here we can see that error is minimum since the data points of actual and predicted value are close to each other



22

## PREDICTED VALUE OF HOUSE PRICE BASED ON TEST SAMPLE DATA

