PROJECT REPORT

COLLAGE NAME : GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY.

COURSE NAME : ARTIFICILAL INTELLIGENCE.

PROJECT NAME : PREDICTING HOUSE PRICE USING MACHINE LEARNING.

Submitted By: kabilan B
Mail Id: kabilanammu2000@gmail.com

TABLE OF CONTENT

SI.NO.	Title	Page.no
01	Aim and Abstract	02
02	Introduction	03
03	Problems in house price prediction	03
04	Ways to fix those problem	05
05	Design and innovation to solve the problem	07
06	Changes in design	10
07	Steps	15
08	Coading	17
09	Conclusion	21

AIM:

- People looking to buy a new home tend to be more conservative with their budgets and market strategies.
- This project aims to analyses various parameters like average income, average area etc. and predict the house price accordingly.
- This application will help customers to invest in an estate without approaching an agent
- To provide a better and fast way of performing operations.
- To provide proper house price to the customers.
- To eliminate need of real estate agent to gain information regarding house prices.
- To provide best price to user without getting cheated.
- To enable user to search home as per the budget
- The aim is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analyzing previous market trends and price range , and also upcoming developments future prices will be predicted.
- House prices increase every year, so there is a need for a system to predict house prices in the future.
- House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house.
- We use linear regression algorithm in machine learning for predicting the house price trends

ABSTRACT:

Data mining is now commonly applied in the real estate market. Data mining's ability to extract relevant knowledge from raw data makes it very useful to predict house prices, key housing attributes, and many more. Research has stated that the fluctuations in house prices are often a concern for house owners and the real estate market. A survey of literature is carried out to analyse the relevant attributes and the most efficient models to forecast the house prices. The findings of this analysis verified the use of the Artificial Neural Network, Support Vector Regression and XG Boost as the most efficient models compared to others. Moreover, our findings also suggest that locational attributes and structural attributes are prominent factors in predicting house prices. This study will be of tremendous benefit, especially to housing developers and researchers, to ascertain the most

significant attributes to determine house prices and to acknowledge the best machine learning model to be used to conduct a study in this field

INTRODUCTION:

House is one of human life's most essential needs, along with other fundamental needs such as food, water, and much more. Demand for houses grew rapidly over the years as people's living standards improved. While there are people who make their house as an investment and property, yet most people around the world are buying a house as their shelter or as their livelihood.

The real estate industry is one of the most significant sectors of the global economy, and for both homebuyers and sellers, accurately predicting house prices is of paramount importance. Traditionally, real estate professionals have relied on their expertise and market knowledge to estimate property values. However, the advent of machine learning has revolutionized the way we approach this task.

PROBLEMS IN HOUSE PRICE PREDICTION:

Predicting house prices using machine learning can be a challenging task due to various factors and potential problems. Here are some common issues and challenges associated with house price prediction using machine learning

Data Quality:

Incomplete data: Missing values in the dataset can lead to biased predictions.

Noisy data: Outliers and errors in the data can negatively impact model accuracy.

Feature Selection and Engineering:

Choosing the right features (variables) and transforming them appropriately is crucial. Including irrelevant features or excluding important ones can affect model performance.

Creating meaningful features from raw data, such as converting categorical variables into numerical representations (e.g., one-hot encoding), can be complex.

Over fitting:

Over fitting occurs when a model learns to perform well on the training data but fails to generalize to unseen data. This can happen if the model is too complex relative to the amount of training data available.

Under fitting:

Under fitting occurs when a model is too simple to capture the underlying patterns in the data. This can happen if the model lacks complexity or if it is not trained for a sufficient number of epochs.

Data Imbalance:

Imbalanced data, where one class (e.g., expensive houses) is underrepresented, can lead to biased predictions.

Scalability:

Scalability can be an issue when dealing with a large dataset. Training a machine learning model on a massive dataset can be computationally expensive and time-consuming.

Non-Linearity:

House price prediction often involves non-linear relationships between features and the target variable. Simple linear regression models may not capture these relationships effectively.

Model Selection:

Choosing the right machine learning algorithm or model architecture can be challenging. Different algorithms have different strengths and weaknesses, and selecting the wrong one can lead to suboptimal results.

Hyper parameter Tuning:

Setting the hyper parameters (e.g., learning rate, regularization strength) of a model is essential for achieving good performance. Tuning these hyper parameters can be time-consuming and require domain expertise.

Data Leakage:

Data leakage occurs when information from the test set inadvertently leaks into the training set, leading to overly optimistic performance estimates. Proper data splitting and preprocessing are crucial to prevent this.

Market Dynamics:

House prices can be influenced by various factors such as economic conditions, location-specific trends, and seasonality. Capturing these external factors in the model can be challenging.

Interpretability:

Some machine learning models, such as deep neural networks, can be difficult to interpret. This can be a problem when stakeholders require explanations for the predictions.

Ethical and Fairness Concerns:

Biases in the data or model can lead to unfair predictions, such as discriminating against certain groups. Ensuring fairness in housing predictions is important.

WAYS TO FIX THOSE PROBLEMS:

Fixing the problems associated with house price prediction using machine learning involves a combination of data pre-processing, model selection, and tuning. Here are steps you can take to address these problems.

Data Quality:

Address missing data: You can use techniques like imputation (e.g., filling missing values with means or medians) or, if applicable, collect more data to reduce missing values.

Handle outliers:

Identify and remove or transform outliers in the data to prevent them from skewing the model's predictions.

Feature Selection and Engineering:

Conduct feature selection: Use techniques like feature importance scores, recursive feature elimination, or domain knowledge to select the most relevant features.

Engineer new features:

Create meaningful features that capture the underlying patterns in the data. For example, you can calculate the price per square foot or include time-based features for seasonality.

Over fitting and Under fitting:

Regularization: Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent over fitting.

Cross-validation:

Use techniques like k-fold cross-validation to assess your model's generalization performance and fine-tune hyper parameters accordingly.

Data Imbalance:

If dealing with imbalanced data, use techniques such as oversampling the minority class or under sampling the majority class to balance the dataset.

Scalability:

Consider distributed computing frameworks like Apache Spark or cloud-based solutions for handling large datasets efficiently.

Non-Linearity:

Use non-linear models like decision trees, random forests, gradient boosting, or neural networks to capture complex relationships in the data.

Model Selection:

Experiment with different machine learning algorithms to find the one that performs best on your specific dataset. Consider ensemble methods for improved accuracy.

Hyper parameter Tuning:

Utilize automated hyper parameter optimization techniques such as grid search, random search, or Bayesian optimization to find the best hyper parameter settings for your model.

Data Leakage:

Carefully split your data into training, validation, and test sets to prevent data leakage. Ensure that any pre-processing steps are applied separately to each dataset.

Market Dynamics:

Incorporate external factors like economic indicators, location-specific trends, and seasonality into your model if they are relevant to house price prediction.

Interpretability:

Choose models that offer better interpretability, such as linear regression or decision trees, when transparency is essential.

Ethical and Fairness Concerns:

Evaluate your model for bias and fairness and consider techniques like re-sampling, re-weighting, or adversarial de biasing to mitigate bias in predictions.

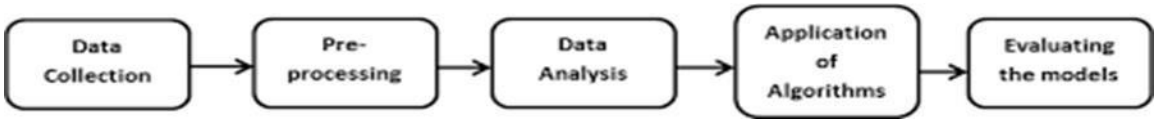
Continuous Monitoring and Improvement:

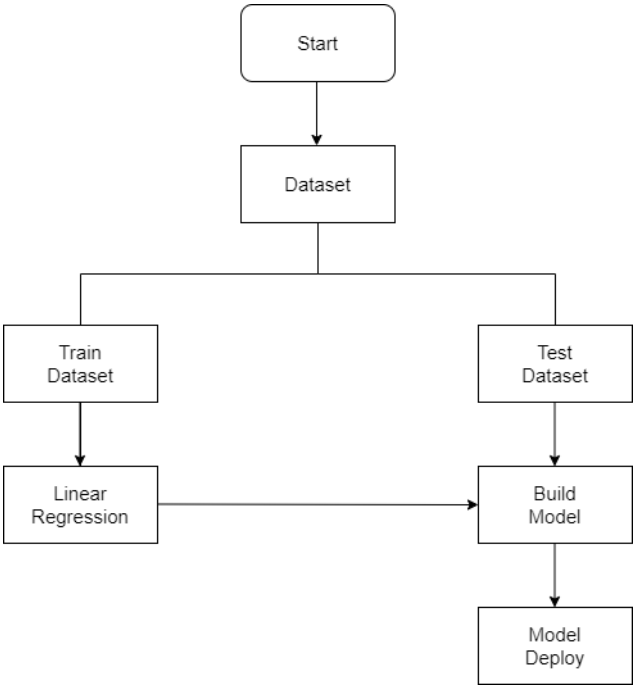
House price prediction models should be updated regularly to account for changing market conditions. Monitor model performance over time and retrain it as needed.

Domain Expertise:

Collaborate with domain experts or real estate professionals who can provide insights into the factors affecting house prices in your target market.

DESIGN:





INNOVATION TO SOLVE THE PROBLEM IN DESIGN

Innovations in design can help address some of the challenges in predicting house prices using machine learning. Here are some innovative approaches and ideas to consider:

Feature Engineering with External Data:

Incorporate external data sources, such as neighborhood crime rates, proximity to schools, or local employment statistics, to enrich your feature set. This can provide more context for predictions.

Time Series Analysis:

Use advanced time series analysis techniques to capture temporal trends in house prices. This can help your model adapt to changing market conditions.

Deep Learning Architectures:

Experiment with advanced deep learning architectures like recurrent neural networks (RNNs) or convolutional neural networks (CNNs) for feature extraction and prediction.

Transfer Learning:

Apply transfer learning techniques from pre-trained models (e.g., language models or image recognition models) to extract valuable features from text or images associated with property listings.

Geospatial Analysis:

Incorporate geospatial analysis techniques to understand the spatial distribution of house prices. Geospatial features like distance to landmarks, public transportation, or amenities can be influential.

Explainable AI (XAI):

Implement methods for making your machine learning model more interpretable and transparent, such as LIME (Local Interpretable Model-Agnostic Explanations) or SHAP (SHapley Additive exPlanations).

Fairness and Bias Mitigation:

Develop algorithms or strategies to detect and mitigate bias in predictions to ensure fair and equitable pricing recommendations.

Blockchain for Property Data:

Explore blockchain technology to securely store and access property data, ensuring data integrity and transparency.

AI-Powered Virtual Tours:

Implement virtual reality (VR) or augmented reality (AR) applications that allow users to take virtual tours of properties, enhancing the buying experience and reducing the need for physical visits.

Natural Language Processing (NLP):

Use NLP techniques to extract insights from property descriptions, customer reviews, or social media sentiment related to neighborhoods.

Predictive Analytics for Market Trends:

Develop models that predict future housing market trends by analyzing economic indicators, political events, and social factors.

Automated Valuation Models (AVMs):

Build AVMs that provide real-time, automated property valuations to assist both buyers and sellers in making informed decisions.

User-Centric Interfaces:

Create intuitive user interfaces that provide personalized recommendations based on user preferences and financial constraints.

AI-Driven Renovation Recommendations:

Develop AI systems that suggest renovation or improvement projects to increase the value of a property based on historical data and market trends.

Collaborative Filtering:

Implement recommendation systems that leverage collaborative filtering techniques to suggest properties similar to those a user has shown interest in.

Blockchain-Based Property Ownership Records:

Utilize blockchain to create a transparent and immutable ledger of property ownership records, reducing fraud and improving trust in the real estate market.

Energy Efficiency Analysis:

Integrate energy efficiency metrics into your predictions, considering factors like insulation, HVAC systems, and renewable energy installations.

Environmental Impact Assessment:

Include an assessment of a property's environmental impact, such as carbon footprint or sustainability features, as a factor in pricing.

AI-Driven Investment Portfolios:

Develop tools that help investors optimize their real estate portfolios by recommending properties with the best potential for return on investment.

Incorporating innovative approaches and technologies into your house price prediction project can enhance the accuracy and utility of your model while addressing some of the challenges associated with real estate data analysis. Be sure to stay updated with the latest advancements in machine learning and related fields to continue improving your solution.

CHANGES IN DESIGN

To address the common problems and challenges in a house price prediction project using machine learning, you can make various design changes and adopt best practices. Here are some design considerations and changes you can implement.

Data Quality Improvement:

Design a robust data cleaning and preprocessing pipeline to handle missing values, outliers, and inconsistent data.

Use techniques like imputation, outlier detection, and data transformation to ensure data quality.

Overfitting and Underfitting Mitigation:

Implement cross-validation to assess model performance and prevent overfitting.

Use regularization techniques (e.g., L1 or L2 regularization) to reduce overfitting.

Experiment with different model complexities and ensembling methods to combat underfitting.

Feature Engineering and Selection:

Invest time in feature engineering to create relevant and meaningful features.

Use feature selection techniques like feature importance analysis or recursive feature elimination to identify the most informative features.

Data Scaling and Transformation:

Apply feature scaling (e.g., standardization or normalization) as needed based on the chosen machine learning algorithms.

Experiment with feature transformations like logarithmic or polynomial transformations for non-linear relationships.

Model Selection and Hyperparameter Tuning:

Conduct thorough model selection by trying multiple algorithms and evaluating their performance.

Perform hyperparameter tuning using techniques like grid search or random search to optimize model parameters.

Data Leakage Prevention:

Carefully review and preprocess the data to avoid any unintentional data leakage.

Ensure that all feature engineering and data preprocessing steps are applied consistently during both training and testing phases.

Bias and Fairness Mitigation:

Examine the dataset for biases and take steps to mitigate them, such as re-sampling or re-weighting.Implement fairness-aware machine learning techniques to reduce bias in predictions.

Model Interpretability:

Choose models that are inherently interpretable, such as linear regression or decision trees, when model interpretability is critical.

Use techniques like SHAP (SHapley Additive exPlanations) values to explain complex model predictions.

Ethical Considerations and Compliance:

Develop and adhere to ethical guidelines for data usage and model deployment.Ensure compliance with relevant regulations, such as Fair Housing Act regulations in the United States.

Continuous Updates and Maintenance:

Establish a process for regularly updating the model with new data to reflect changing market conditions.Implement automated monitoring and alerting systems to detect performance degradation.

Data Privacy and Security:

Implement robust data security and privacy measures, including encryption, access controls, and anonymization of sensitive data.

Scarcity of Data:

Consider data augmentation techniques, synthetic data generation, or transfer learning if data scarcity is a significant issue in certain areas.

By incorporating these design changes and best practices into your house price prediction project, you can create a more reliable and robust machine learning model that addresses common challenges and produces more accurate predictions while adhering to ethical and regulatory considerations.

BLOCKS TO BE ADDED

We should add some blocks to improve our design and it reduce our complication in our project To address the common problems encountered in a house price prediction project, you can add specific blocks or components to your project design. Here's a breakdown of the blocks you can incorporate, one by one, to solve these problems:

Data Quality and Preprocessing Block:

Data Cleaning and Imputation Component: Develop a data cleaning pipeline that handles missing values, removes duplicates, and corrects inaccuracies in the dataset. Techniques like mean imputation, median imputation, or imputing with predictive models can be used.

Outlier Handling Component: Implement an outlier detection and handling strategy, which may involve techniques like statistical tests, visualization, or robust regression models to mitigate the impact of outliers.

Overfitting and Underfitting Block:

Cross-Validation Component: Integrate k-fold cross-validation into your modeling pipeline to assess and mitigate overfitting or underfitting.

Regularization Component: Add regularization techniques such as L1 (Lasso) and L2 (Ridge) regularization to your model training process to prevent overfitting.

Feature Engineering Block:

Feature Selection Component: Use feature selection techniques like recursive feature elimination or feature importance ranking to choose the most relevant features for your model.

Feature Scaling Component: Incorporate feature scaling (e.g., Min-Max scaling or Standardization) into your preprocessing pipeline to ensure features are on a consistent scale.

Data Imbalance Block:

Resampling Component: If dealing with imbalanced data, implement resampling techniques like oversampling (e.g., SMOTE) or undersampling to balance the dataset.

Model Selection and Hyperparameter Tuning Block:

Algorithm Selection Component: Experiment with various regression algorithms, such as Linear Regression, Random Forest, Gradient Boosting, and Support Vector Regression, to determine which performs best for your dataset.

Hyperparameter Tuning Component: Incorporate automated hyperparameter tuning methods like grid search or Bayesian optimization to optimize your model's performance.

Generalization and Updates Block:

Regular Updating Component: Plan for periodic model updates with new data to ensure your model remains relevant and accurate in changing market conditions.

Ethical Considerations Block:

Bias Detection and Mitigation Component: Implement bias detection methods and fairness-aware algorithms to identify and address bias in your dataset and model predictions.

Explainability Component: Use explainable AI techniques to make your model's predictions more transparent and interpretable.

Data Privacy and Security Block:

Data Encryption Component: If applicable, add data encryption mechanisms to protect sensitive data.

Anonymization Component: Integrate data anonymization techniques when dealing with personally identifiable information (PII).

Deployment Challenges Block (if applicable):

Scalability Component: Design a scalable infrastructure that can handle increased traffic as your user base grows.

Monitoring and Maintenance Component: Develop robust monitoring systems that continuously track model performance and trigger updates when necessary.

Changing Market Conditions Block:

Real-time Data Integration Component: Implement real-time data ingestion and integration to keep your model up-to-date with current market conditions.

Interpretability and Explainability Block:

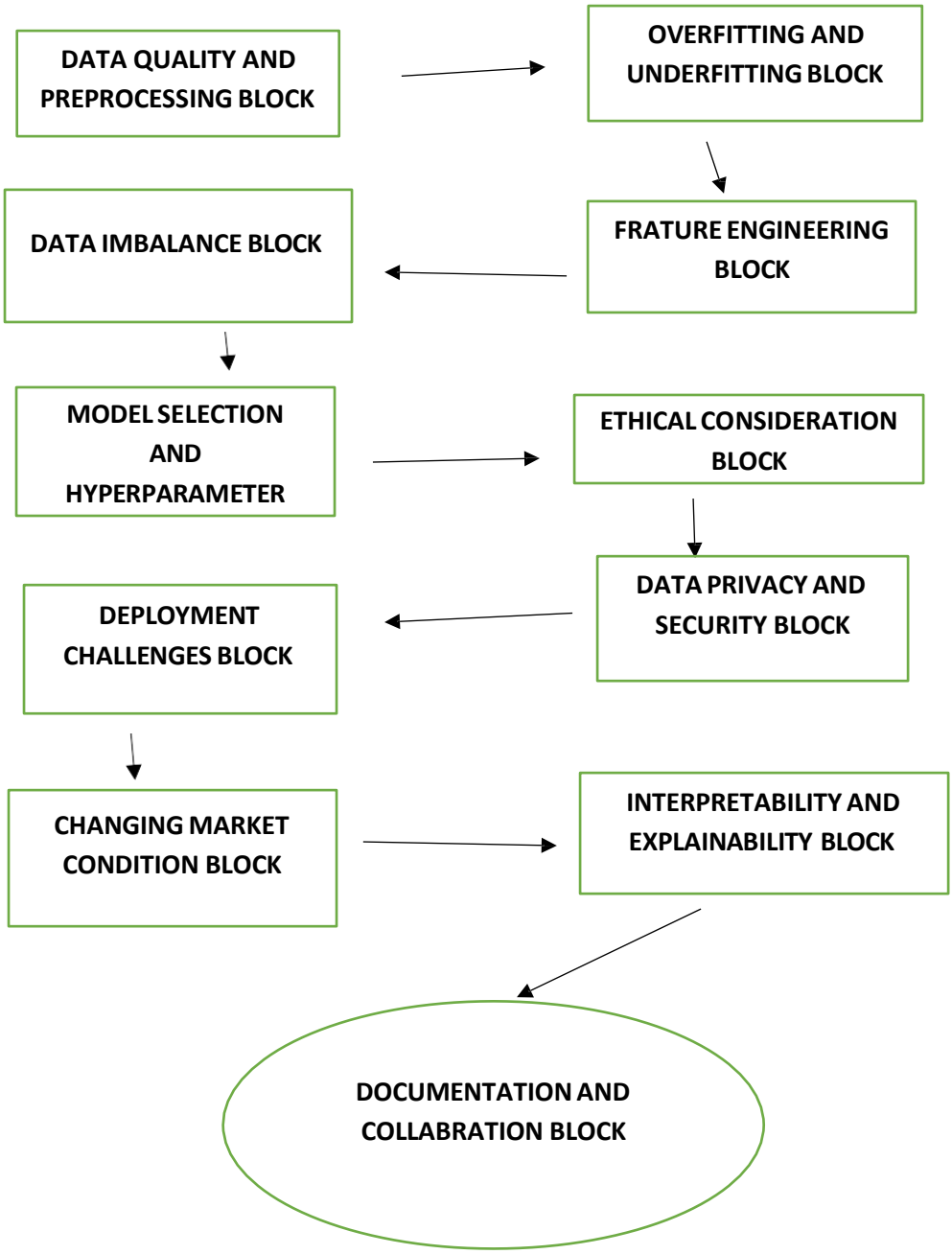
Interpretability Component: Add components or techniques for model interpretation, such as feature importance plots or SHAP values, to explain model predictions.

Documentation and Collaboration Block:

Documentation Component: Establish documentation practices that record all project details, decisions, and changes made.

Collaboration Component: Maintain open communication with domain experts and stakeholders throughout the project to ensure alignment with business goals.

By systematically adding these blocks and components to your project design, you can effectively address the common problems and challenges associated with predicting house prices using machine learning. This structured approach will help you build a more robust and reliable model while ensuring ethical and transparent practices.


DATA QUALITY AND PREPROCESSING BLOCK
OVERFITTING AND UNDERFITTING BLOCK
FRATURE ENGINEERING BLOCK
DATA IMBALANCE BLOCK
MODEL SELECTION AND HYPERPARAMETER
ETHICAL CONSIDERATION BLOCK
DATA PRIVACY AND SECURITY BLOCK
DEPLOYMENT CHALLENGES BLOCK
CHANGING MARKET CONDITION BLOCK
INTERPRETABILITY AND EXPLAINABILITY BLOCK
DOCUMENTATION AND COLLABRATION BLOCK

STEPS :

In this phase we need to do loading and pre-processing the datasets, feature engineering, model training, evaluation etc. Here I explain about what are the process to do this phase.

Step 1:

Import the dependencies

In this step we import the library files which are required to run this program code, like modules (numpy , pandas, sklearn , matpltlib, seaborn, XGBoost).

Step 2:

Impoting the dataset

In this step I import California_housing dataset form the the sklearn module it is used to fetch the data and the data is used as the input of this project.

Step 3:

Loading the dataset into the pandas dataframe

In this step I load my data to the pandas data frame which gives the structure of our dataset

Step 4:

Checking the number of rows and column to the dataset

In this step I check the number of rows and column to the dataset and also check any missing values in my dataset

Step 5:

Statical measure of dataset

In this step to check some stats related to my dataset like(count, mean, minimum, maximum, standard deviation).

Step 6:

Generate heat map

In this step to generate heat map to know about my dataset clearly by using the module matplotlib.pyplot

Step 7:

Splitting data and target

In this step we need to split the data into two parts namely DATA and TARGET . in this step we declare the variable X for data and variable Y for target

Step 8:

Splitting the data into training and testing data

In this step I split my data into two component they are training data and testing data by using **train _test_split** command

Step 9:

Model Training

In this step I train my data by using **XGBoost regressor** algorithm

Step 10:

Fixing the train and test data to the model (XGBoost Regressor)

In this step I fit my train and test data to the model by using **model.fit** command

Step 11:

Prediction on train and test data

In this step to predict the train and test data by using **model.predict** command. And also find r square error and mean absolute error for train and test data

Step 12:

Visualizing the actual price and predicted price

In this step to generate prediction graph to to evaluate my project the gaph is created by using the module matplotlib.pyplot

Import the dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

Impoeting the california house prise dataset

```
 from sklearn.datasets import fetch_california_housing
house_price_dataset = fetch_california_housing()
```

```
print(house_price_dataset)
```

```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]), 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]), 'frame': None, 'target_n
```

```
# loading the dataset to the Pands DataFrame
house_price_dataframe = pd.DataFrame(house_price_dataset.data, columns = house_price_dataset.feature_names)
```

```
# print first 5 rows of our DataFrame
house_price_dataframe.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
# add the target column to the DataFrame
house_price_dataframe['price'] = house_price_dataset.target
```

```
house_price_dataframe.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
# checking the number of rows and columns in the data frame
house_price_dataframe.shape
```

```
(20640, 9)
```

```
#check for missing values
house_price_dataframe.isnull().sum()
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
price         0
dtype: int64
```

```
# statical measure of the dataset
house_price_dataframe.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOcc
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.0000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.0706
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.3860
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.6923
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.4297
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.8181
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.2822
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.3333

underatanding various feature in the dataset

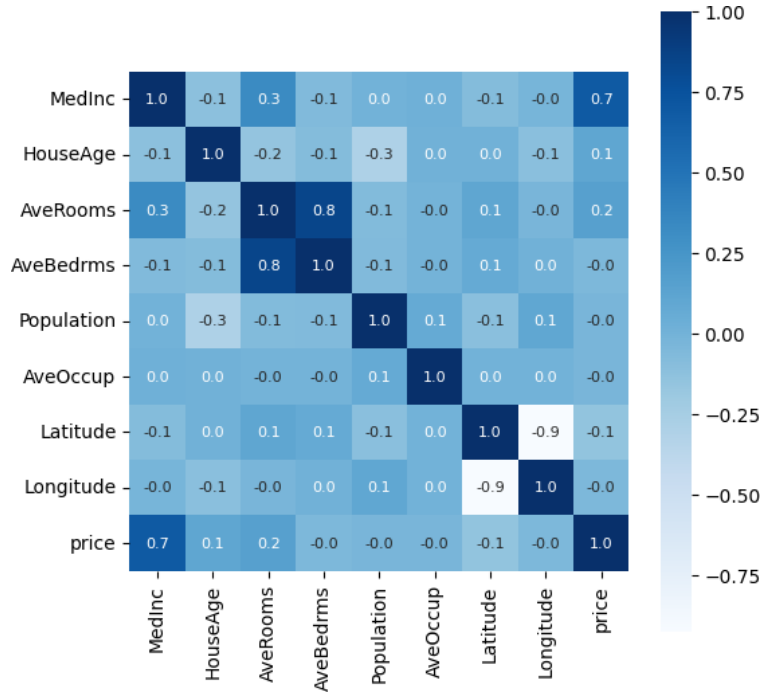
1.positive correlation 2.negative correlation

```
correlation = house_price_dataframe.corr()
```

constructing the heatmap

```
# constructing the heatmap to understand the correlation
plt.figure(figsize=(6,6))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')
```

```
<Axes: >
```



splitting data and target

```
X = house_price_dataframe.drop(['price'], axis=1)
Y = house_price_dataframe['price']
```

```
print(X)
print(Y)
```

```
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25
...       ...       ...       ...        ...         ...       ...       ...         ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48    -121.09
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49    -121.21
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43    -121.22
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43    -121.32
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37    -121.24

[20640 rows x 8 columns]
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: price, Length: 20640, dtype: float64
```

splitting the data into training data and test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(20640, 8) (16512, 8) (4128, 8)
```

model training

XGBoost regressor

```
# loading the model
model = XGBRegressor()
```

```
# training the model with x train
model.fit(X_train, Y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

evaluation

predection on training data

```
# accuracy for prediction on training data
training_data_prediction = model.predict(X_train)
```

```
print(training_data_prediction)
```

```
[0.5523039 3.0850039 0.5835302 ... 1.9204227 1.952873  0.6768683]
```

```
# R squared error
score_1 = metrics.r2_score(Y_train, training_data_prediction)
```

```
# mean absolute error
score_2 = metrics.mean_absolute_error(Y_train, training_data_prediction)
print("R squared error : ", score_1)
print("mean absolute error : ", score_2)
```

```
R squared error :  0.943650140819218
mean absolute error :  0.1933648700612105
```

visualizing the actual price and predicted price

```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```

prediction on test data

```
# accuracy for prediction on test data
test_data_prediction = model.predict(X_test)
```

```
# R squared error
score_1 = metrics.r2_score(Y_test, test_data_prediction)

# mean absolute error
score_2 = metrics.mean_absolute_error(Y_test, test_data_prediction)
print("R squared error : ", score_1)
print("mean absolute error : ", score_2)
```

```
R squared error :  0.8338000331788725
mean absolute error :  0.3108631800268186
```

CONCLUSION:

Thus the machine learning model to predict the house price based ongiven dataset is executed successfully using xg regressor (a upgraded/slighted boosted form of regular linear regression, this gives lesser error). This model further helps people understand whether this place is more suited for them based on heatmap correlation. It also helps people looking to sell a house at best time for greater profit. Any house price in any location can be predicted with minimum errorby giving appropriate dataset.