VISVESVARAYA TECHNOLOGICAL UNIVERSITY

jnana sangama, belgaum – 590 018

INTERNSHIP REPORT ON

"HOUSE PRICE PREDICTION"

Submitted in partial fulfilment for the requirements of the VII Semester of degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

For the Academic year 2023-2024

By

ESHAN KUSHWAH [1DB20CS039]

Internship carried out at

SAMSUNG INNOVATION CAMPUS

Internal Guide

Dr. Sameena Bano

Professor, Dept. of CSE

External Guide

Mr. Prasad K

Samsung Innovation Campus

Together for Tomorrow! Enabling People

Education for Future Generations

DON BOSCO INSTITUTE OF TECHNOLOGY
BANGALORE-560074

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DON BOSCO INSTITUTE OF TECHNOLOGY

BANGALORE-560074

CERTIFICATE

This is to certify that the DBIT-Samsung Innovation Campus Internship entitled "**HOUSE PRICE PREDICTION"** is a bonafide report carried out by **ESHAN KUSHWAH (1DB20CS039),** student of **DON BOSCO INSTITUTE OF TECHNOLOGY** in partial fulfillment for the award of the degree of Bachelor of Engineering in Computer science and Engineering of the **Visvesvaraya Technological University, Belgaum** during the academic year **2023-24**. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The internship has been approved as it satisfies the academic requirements in respect of the internship prescribed for the Bachelor of Engineering Degree.

Signature of guide	**Signature of HOD**	**Signature of Principal**
……	…….....	……
Dr. Sameena Bano	**Dr. K ShivaKumar**	**Dr. B Nagabhushana**
Associate Professor,	H.O.D., Dept of CSE	Principal,
Dept of CSE,	D.B.I.T, Bengaluru	D.B.I.T, Bengaluru
D.B.I.T, Bengaluru		

External Viva

Name of Externals	**Signature with Date**
1. _____	_____
2. _____	_____

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DON BOSCO INSTITUTE OF TECHNOLOGY

BANGALORE-560074

DECLARATION

I, **ESHAN KUSHWAH** student of seventh semester B.E, Computer Science and Engineering, Don Bosco Institute of Technology, Bengaluru declare that the internship entitled "**HOUSE PRICE PREDICTION**" has been carried out by me and submitted in partial fulfilment of the course requirements for the seventh semester examination of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year **2023-24**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree.

Place: Bangalore **ESHAN KUSHWAH**

Date: **(1DB20CS039)**

ACKNOWLEDGEMEMT

The satisfaction and euphoria that successful completion of any internship is incomplete without the mention of people who made it possible, whose constant support and encouragement made my effort fruitful.

First and foremost, I ought to pay my due regards to this institute, which provided me a platform and gave an opportunity to display my skills through the medium of internship. I express heartfelt thanks to beloved principal **Dr. B S Nagbhushana** , Don Bosco Institute of Technology, Bangalore for his encouragement all through my graduation life and providing me with the infrastructure.

I express my deep sense of gratitude and thanks to **Dr. K B Shivakumar** & Head of the Department, computer Science and Engineering for extending his valuable insight and suggestions offered during the course of this internship.

It is my utmost pleasure to acknowledge the kind help extended by my guide **Dr.Sameen Bano,** Professor, Department of computer Science, and also **Prof. Rashmi B C**, Assistant Prof., Dept of CSE for excellent guidance and cooperation which consequently resulted in getting the internship completed successfully.

Last but not the least I would like to thank all my friends and family for their help and support in completing this internship.

ESHAN KUSHWAH

(1DB20CS039)

ABSTRACT

House price forecasting is an important topic of real estate. The literature attempts to derive useful knowledge from historical data of property markets. Machine learning techniques are applied to analyze historical property transactions in India to discover useful models for housebuyers and sellers. Accurate prediction of residential property prices is pivotal for various stakeholders in the real estate industry. The effectiveness of machine learning algorithms is investigated to forecasting house prices based on a comprehensive dataset comprising featureslike location, size, amenities, and historical sales data. Multiple models including linear regression, decision trees, random forests, support vector machines, and neural networks are deployed to construct predictive models.

Evaluation metrics, including mean squared error, R-squared, and root mean square error, are employed to assess model performance. The results demonstrate that ensemble methods, specifically random forests and gradient boosting, consistently outperform other models, showcasing superior prediction accuracy. Feature engineering techniques such as one-hot encoding and polynomial features play a significant role in enhancing predictive capabilities. Additionally, XGBoost provides a measure of feature importance, offering valuable insights into which factors weigh most heavily in determining property values. Its ability to handle large datasets, prevent overfitting, and deliver reliable predictions makes XGBoost a popular choicein real estate for accurately valuing residential properties. The findings provide valuable insights for stakeholders in the real estate industry, enabling them to make more accurate and informed decisions in a dynamic and competitive market.

TABLE OF CONTENT

LIST OF FIGURES

Chapter-1

INTRODUCTION

House price prediction is the practice of estimating the value of a residential property based on various factors. This process is crucial in real estate for buyers, sellers, and investors to make informed decisions. The real estate industry plays a pivotal role in economic development, making accurate house price prediction a crucial aspect for various stakeholders including buyers, sellers, and investors.

Predicting house prices presents a formidable challenge owing to the intricate interplay of numerous influential variables. Factors such as location, size, condition, and economic trends all converge to determine a property's market value. Traditionally, real estate professionals used methods like Comparative Market Analysis (CMA) to assess a property's value. CMA involves comparing the target property to similar properties that have recently sold in the area. While this approach is useful, it has limitations, such as subjectivity and a reliance on a limited set of features.

In recent years, the advent of advanced data analytics and machine learning has transformed the way house price prediction is done. Machine learning algorithms can analyze large datasets containing various features like location, size, number of bedrooms and bathrooms, proximity to amenities, and more. These algorithms learn patterns from historical data to make predictions about future property values.

The selection of features is crucial in house price prediction. Factors like location, the condition of the property, square footage, and recent renovations all play a significant role in determining a property's value. Different machine learning models can be applied to this problem. Linear regression, for example, is a straightforward model that assumes a linear relationship between features and price. More complex models like decision trees, random forests, and neural networks can capture non-linear relationships and interactions between features. House price prediction is a crucial aspect of real estate that has evolved with the advent of advanced data analytics and machine learning. By leveraging data-driven approaches, we can make more accurate and informed decisions in the dynamic real estate market.

1.1 Existing System

The existing system of house price prediction predominantly relies on traditional methods and expertise of real estate professionals**.**

Comparative Market Analysis (CMA): This method is widely used where real estate professionals assess a property's value by comparing it to recently sold properties in the same area. It takes into account factors like location, size, condition, and other relevant features.

Appraisal Methods: Certified appraisers play a critical role in estimating a property's value. They employ a combination of market data, property inspections, and industry knowledge. This can involve both the cost approach (evaluating the cost to replace the property) and the sales comparison approach (comparing it to similar properties).

Hedonic Pricing Model: This statistical method breaks down a property into its constituent features (e.g., number of bedrooms, location, size) and estimates the contribution of each feature to the property's overall value.

These methods, based on professional judgment and industry experience, form the backbone of property valuation. They are used by real estate agents, appraisers, and financial institutions to determine fair market values for residential properties. While they have proven effective over time, it's worth noting that the field is evolving, and there's growing interest in incorporating advanced data-driven techniques to enhance accuracy and efficiency in house price prediction.

1.2 Problem Statement

In the real estate market, accurately predicting house prices is a critical challenge. Traditional valuation methods often rely on subjective assessments and limited data, leading to discrepancies in property valuations. Moreover, factors like location, size, condition, and market trends interact in complex ways, making precise predictions difficult. As a result, there is a pressing need for advanced data-driven approaches, such as machine learning models, to provide more accurate and reliable house price predictions. This project aims to develop a robust and scalable house price prediction system that leverages modern data analytics techniques, considering a diverse range of features, to enhance the accuracy and efficiency of property valuations.

1.3 Objectives

Accurate Valuation: Develop a model that can accurately estimate the market value of residential properties based on a diverse set of features including location, size, amenities, and historical sales data.

Market Trends Analysis: Identify and incorporate market trends, both short-term and long-term, to provide up-to-date valuations that reflect the dynamic nature of the real estate market.

Risk Assessment for Financial Institutions: Provide a reliable basis for financial institutions to assess lending risks, enabling them to determine appropriate loan amounts and interest rates.

Informed Decision-Making: Empower buyers, sellers, and investors with precise property valuations, aiding them in making well-informed decisions regarding purchasing, selling, or investing in real estate.

Optimize Investment Strategies: Enable investors to identify emerging market trends and formulate effective investment strategies by leveraging accurate property valuations.

Benchmarking and Evaluation: Continuously assess the performance of the prediction model using appropriate metrics (e.g., Mean Absolute Error, R-squared) to ensure its effectiveness and reliability in providing accurate valuations.

Chapter-2

LITERATURE REVIEW

Anand G. Rawool , Dattatray V. Rogye , Sainath G. Rane , Dr. Vinayk A, “In proposed model of Machine Learning, the dataset is divided into two parts: Training and Testing. 80% of data is used for training purpose and 20% used for testing purpose. The training set include target variable. The model is trained by using various machine learning algorithms, out of which Random forest regressions predict better results. For implementing the Algorithms, they have used Python Libraries NumPy and Pandas” [1].

Pei-ying wang1, Chiao-ting chen , “In Deep Learning Model study, the authors have developed a mode based on using Heterogeneous Data Analysis Along with Joint Self-Attention Mechanism. The Heterogeneous Data is to supplement house information, and it also assigns the weights automatically depending different features or samples” [2].

Chenhao Zhou, “House price prediction using polynomial regression with Particle Swarm Optimization the authors have Washington DC house price prediction using polynomial regression and particle swarm optimization methods. They have also improved particle swarm optimization method with two methods. One is changing the topological structure of particle relations and second improvement is the introduction of new particle control mechanisms” [3].

Ankita Kamire , Nitin Chaphalkar , Sayali Sandbhor, “The present study uses data of sales transactions and the valuation of real estate properties from Pune city. For modeling the prediction process, the data is converted into the format of variables and the corresponding outcome in terms of the value of the property. The results are presented by using the performance matrices such as MAPE and R2, where Mean Absolute Percentage Error (MAPE) is most commonly used to forecast the error of any model” [4].

Anirudh Kaushal, Achyut Shankar “Real Property Value Prediction Capability Using Fuzzy Logic and ANFIS study uses data of sales transactions and the valuation of real estate properties from Pune city. For modeling the prediction process, the data is converted into the format of variables and the corresponding outcome in terms of the value of the property. The results are presented by using the performance matrices such as MAPE and R2, where Mean Absolute Percentage Error (MAPE) is most commonly used to forecast the error of any model” [5].

Gamze Tanak Coskun , Ayten Yılmaz Yalçıner, “In another paper based on Machine Learning has used the multivariate linear regression model to perform the prediction. Also, it is compared with other Machine Learning models like Lasso Regression, LassoCV, Ridge, RidgeCV and decision tree regressor. Multivariate linear regression and LassoCV performs the best with 84.5% accuracy” [6].

Jian Guan, Jozef Zurada, and Alan S. Levitan, “Determining the best price with linear performance pricing and checking with fuzzy logic paper aims to compare and verify findings with the LPP method and Fuzzy Logic results. This article explained linear performance pricing (LPP), backed up its accuracy with fuzzy logic, and showed how it can be used to efficiently provide the focus needed to achieve cost reduction. Besides, although it is widely used in the automotive industry in the USA and Europe, there is little discussion in the literature about its support with LPP and fuzzy logic” [7].

Andrzej Biłozor and Maurizo d’Amato, “ANFIS approach is first time applicable to the real estate property assessment. This study has shown that ANFIS can yield results that are comparable to those obtained using the traditional regression approach. The main contribution of this study is clear demonstration that ANFIS is a viable approach in real estate value assessment and is worthy of further exploration” [8].

Felipe Alonso Arias Arbelaez and Francisco Ivan Zuluaga Daz ,“In this study authors have aimed to developed four different methods in order to estimate the real market price of 380 properties owned by Midtown Realty Group in Miami, Florida. the FIS models that also explain satisfactorily nonlinear relationships of the variables with the dependent variable; however, the problem we have in these models is that they must be well defined number of inputs and relationships they have with each other, that is, the rules that explain the fuzzy inference model” [9].

Abdul G. Sarip and Muhammad Burhan Hafez, “In this paper a study has been shown to compare different methods like ANN, FL, and FLSR for house price prediction. The major focus was on FL and FLSR, and on Fuzzy Regression. Fuzzy Regression Model was explained through a graph of predicted and actual values. Also, a Result comparison is also done with MAE (Mean Absolute Error), which shows considerable reduction is achieved in FIS and FLSR where the error rate drops by more than 25000 compared to the MAE of ANN model” [10].

Chapter-3

PROPOSED SYSTEM

3.1 Requirements

➢ Hardware Requirements:

- Processor : Intel Core 3 and above
- RAM: 8GB
- Operating System: Windows 10

➢ Software Requirements:

- Programming language : Python 3.6
- IDE: Google Colab
- Front End Tool: Streamlit , CSS
- Libraries Used:
    1. Streamlit
    2. Pandas
    3. Seaborn
    4. Sklearn
    5. Matplotlib
    6. Numpy
    7. XGBoost

3.2 Library

- Pandas: Pandas is a powerful data manipulation and analysis library for Python. It Organizes and clean the dataset for use in training the machine learning model in house price prediction.
- Matplotlib: Matplotlib is a popular Python library for creating static, animated, or interactive visualizations. Generate plots to understand relationships between features and house prices.

- Numpy: Numpy is used for mathematical operations on large, multi-dimensional arrays and matrices. Performs numerical operations on features, such as scaling or normalizing data in house price prediction.
- Seaborn: Seaborn is based on matplotlib, used for plotting statistical graphics. It helps to Create more advanced statistical visualizations to explore relationships between variables to explore features of a house.
- Scikit-learn (sklearn): sklearn is a versatile and widely-used machine learning library in Python, providing a rich set of tools for data analysis, modelling, and evaluation. It Utilize various regression algorithms (e.g., Linear Regression, Decision Tree Regressor) to predict house prices.
- Xgboost: This library is an implementation of the XGBoost algorithm, an optimized and highly efficient gradient boosting framework that is widely used for both regression and classification tasks in machine learning. Trains an ensemble model like XGBoost for regression to potentially improve the predictive performance of house price.
- Streamlit: Streamlit is a Python library used to create web applications for data science and machine learning. Its simplicity and flexibility enabled the creation of an intuitive web-based platform where users could interact with the system where users can input features of a house, and the trained model can predict its price.

3.3 Dataset

The dataset is retrieved from kaggle which consist of trained set and test dataset. It contains 1460 training data points and 80 features that might help us predict the selling price of a house.

3.3.1 Dataset used in the project

Here's a brief data description of dataset.

- **SalePrice** - the property's sale price in dollars. This is the target variable that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification like agriculture, commercial, industrial
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access like gravel, paved

- **Alley**: Type of alley access like gravel, paved
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available like electricity, gas
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling like single family detached, duplex
- **HouseStyle**: Style of dwelling like one story,two story
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof like flat, hip,shed
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation like slab, stone, wood ,brick
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area

- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating like wall furnace, floor furnace
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **1stFlrSF**: First Floor square feet
- **2ndFlrSF**: Second floor square feet
- **LowQualFinSF**: Low quality finished square feet (all floors)
- **GrLivArea**: Above grade (ground) living area square feet
- **BsmtFullBath**: Basement full bathrooms
- **BsmtHalfBath**: Basement half bathrooms
- **FullBath**: Full bathrooms above grade
- **HalfBath**: Half baths above grade
- **Bedroom**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **WoodDeckSF**: Wood deck area in square feet
- **OpenPorchSF**: Open porch area in square feet
- **EnclosedPorch**: Enclosed porch area in square feet
- **3SsnPorch**: Three season porch area in square feet

- **ScreenPorch**: Screen porch area in square feet
- **PoolArea**: Pool area in square feet
- **PoolQC**: Pool quality
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

3.4 Algorithm

1. Data Retrieval and Source: The model utilizes real estate data from an online database or platform, such as publicly available real estate listings or a curated dataset.

2. Data Preprocessing: Various preprocessing techniques are applied to the dataset. This includes handling missing values, encoding categorical variables, scaling features, and potentially performing feature engineering (e.g., extracting additional information from property descriptions or images).

3. Linear Regression: Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features. The goal of the algorithm is to find the best linear equation that can predict the value of the dependent variable based on the independent variables.

4. Decision tree Regression: Decision Tree Regression is a method that uses a tree-like model to make predictions. Each "branch" of the tree represents a decision based on a feature, leading to a final prediction at the "leaves" of the tree. Features like square footage, number of bedrooms, location, etc., are evaluated to determine which ones are most influential in predicting house prices. The algorithm starts with the entire dataset and chooses the best feature to split it into subsets. For example, it may find that "square footage > 2000 sq. ft." is a good first split. This process continues, creating a tree structure. At each step, the algorithm chooses the best feature and value to split the data, aiming to minimize prediction errors. The final nodes of the tree (the

"leaves") represent specific conditions that lead to a predicted house price. For example, a leaf might represent "square footage > 2000 sq. ft. and located in a certain neighborhood."When you want to predict the price of a new house, you follow the path through the tree based on its features, and at the end, you get the predicted price.

5. XGBoost Regression: XGBoost is an ensemble learning technique that combines the predictions of multiple individual models (usually decision trees) to make more accurate predictions. It trains a sequence of models, where each subsequent model corrects the errors made by the previous ones. This is done by focusing on the data points that were predicted incorrectly. XGBoost uses gradient boosting, a technique that involves minimizing a loss function (e.g., mean squared error for regression) by iteratively fitting a new model to the residuals (the differences between actual and predicted values).

6. Selection of Machine Learning Algorithm: The choice of machine learning algorithm is crucial. In this case, the model uses the XGBoost regressor, which is known for its effectiveness in regression tasks and its ability to handle complex relationships in data.

7. Model Training and Validation: The dataset is split into training and validation sets. The model is trained on the training set and then validated on the validation set to ensure it generalizes well to new data.

8. Hyperparameter Tuning: Techniques like Randomized Search or Grid Search may be employed to fine-tune the hyperparameters of the XGBoost model, optimizing its performance.

6. Model Evaluation: The model's performance is evaluated using appropriate metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or R-squared (R2) to quantify its accuracy in predicting house prices.

9. Feature Importance Analysis: XGBoost provides a feature importance score, indicating which features have the most impact on the predicted house prices. This analysis helps identify the most influential factors.

10. Prediction and Deployment: The trained model is then used to make predictions on new or unseen data. It can be deployed in a web application or integrated into an existing system for real-time predictions.

11. Continuous Monitoring and Updating: The model's performance should be monitored over time, and it may need to be retrained or fine-tuned periodically to account for changing real estate market trends.

12. Error Handling and Robustness: The model should be equipped to handle unexpected scenarios, such as missing data in new predictions or outliers, to ensure its robustness in a real-world setting.

3.5 Coding

```
# 1. Import Libraries
# 'Pandas' is used for data manipulation and analysis
import pandas as pd
# 'Numpy' is used for mathematical operations on large, multi-dimensional arrays and matrices
import numpy as np
# 'Matplotlib' is a data visualization library for 2D and 3D plots, built on numpy
import matplotlib.pyplot as plt
# 'Seaborn' is based on matplotlib; used for plotting statistical graphics
import seaborn as sns
# import various functions to perform regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
import xgboost
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeRegressor
# 2. Set Options
# To show the all columns
pd.set_option("display.max_columns", 2000)
# To show all rows
pd.set_option("display.max_rows", 250)
# 3. Read Data
```

```
# read the csv data file

train_data=pd.read_csv('train.csv')

# display the top 5 rows of the dataframe

train_data.head()

#### Dimensions of the data

train_data.shape

# 4. Data Analysis and Preparation

## 4.1 Understand the Dataset

### 4.1.1 Data Type

#The main data types in Pandas dataframes are the object, float, int64, bool, and datetime64. To understand each attribute of our data, it is always good for us to know the data type of each column.

# 'dtypes' gives the data type for each column

train_data.dtypes

### 4.1.2 Check for missing Valuee

# get the count of missing values

missing_values =train_data.isnull().sum()

# print the count of missing values

print(missing_values)

### 4.1.3 Handling missing Value

train_data['LotFrontage']=train_data['LotFrontage'].fillna(train_data['LotFrontage'].mean())

train_data['Alley']=train_data['Alley'].fillna(train_data['Alley'].mode()[0])

train_data['BsmtCond']=train_data['BsmtCond'].fillna(train_data['BsmtCond'].mode()[0])

train_data['BsmtQual']=train_data['BsmtQual'].fillna(train_data['BsmtQual'].mode()[0])

train_data['FireplaceQu']=train_data['FireplaceQu'].fillna(train_data['FireplaceQu'].mode()[0])

train_data['GarageType']=train_data['GarageType'].fillna(train_data['GarageType'].mode()[0])

train_data.drop(['GarageYrBlt'],axis=1,inplace=True)

train_data['GarageFinish']=train_data['GarageFinish'].fillna(train_data['GarageFinish'].mode()[0]
)

train_data['GarageQual']=train_data['GarageQual'].fillna(train_data['GarageQual'].mode()[0])

train_data['GarageCond']=train_data['GarageCond'].fillna(train_data['GarageCond'].mode()[0])
```

```
train_data['PoolQC']=train_data['PoolQC'].fillna(train_data['PoolQC'].mode()[0])

train_data['Fence']=train_data['Fence'].fillna(train_data['Fence'].mode()[0])

train_data['MiscFeature']=train_data['MiscFeature'].fillna(train_data['MiscFeature'].mode()[0])

train_data.drop(['Id'],axis=1,inplace=True)

train_data['MasVnrType']=train_data['MasVnrType'].fillna(train_data['MasVnrType'].mode()[0])

train_data['MasVnrArea']=train_data['MasVnrArea'].fillna(train_data['MasVnrArea'].mode()[0])

train_data['BsmtExposure']=train_data['BsmtExposure'].fillna(train_data['BsmtExposure'].mode(
)[0])

train_data['BsmtFinType2']=train_data['BsmtFinType2'].fillna(train_data['BsmtFinType2'].mode
()[0])

train_data.isna().sum()

#remove the rows which has a null value

train_data.dropna(inplace=True)

#gives total number of null values in each columns

train_data.isna().sum()

train_data.shape
```

4.1.4 Statistical Summary

```
# data frame with numerical features

train_data.describe()
```

4.1.5 Cleaning of test data

```
test_data=pd.read_csv('test.csv')

test_data.shape

test_data.isnull().sum()

test_data['LotFrontage']=test_data['LotFrontage'].fillna(test_data['LotFrontage'].mean())

test_data['MSZoning']=test_data['MSZoning'].fillna(test_data['MSZoning'].mode()[0])

test_data['Alley']=test_data['Alley'].fillna(test_data['Alley'].mode()[0])

test_data['BsmtCond']=test_data['BsmtCond'].fillna(test_data['BsmtCond'].mode()[0])

test_data['BsmtQual']=test_data['BsmtQual'].fillna(test_data['BsmtQual'].mode()[0])

test_data['FireplaceQu']=test_data['FireplaceQu'].fillna(test_data['FireplaceQu'].mode()[0])

test_data['GarageType']=test_data['GarageType'].fillna(test_data['GarageType'].mode()[0])

test_data.drop(['GarageYrBlt'],axis=1,inplace=True)
```

```
test_data['GarageFinish']=test_data['GarageFinish'].fillna(test_data['GarageFinish'].mode()[0])
test_data['GarageQual']=test_data['GarageQual'].fillna(test_data['GarageQual'].mode()[0])
test_data['GarageCond']=test_data['GarageCond'].fillna(test_data['GarageCond'].mode()[0])
test_data['PoolQC']=test_data['PoolQC'].fillna(test_data['PoolQC'].mode()[0])
test_data['Fence']=test_data['Fence'].fillna(test_data['Fence'].mode()[0])
test_data['MiscFeature']=test_data['MiscFeature'].fillna(test_data['MiscFeature'].mode()[0])
test_data.drop(['Id'],axis=1,inplace=True)
test_data['MasVnrType']=test_data['MasVnrType'].fillna(test_data['MasVnrType'].mode()[0])
test_data['MasVnrArea']=test_data['MasVnrArea'].fillna(test_data['MasVnrArea'].mode()[0])
test_data['BsmtExposure']=test_data['BsmtExposure'].fillna(test_data['BsmtExposure'].mode()[0]
)
test_data['BsmtFinType2']=test_data['BsmtFinType2'].fillna(test_data['BsmtFinType2'].mode()[0
])
test_data['Utilities']=test_data['Utilities'].fillna(test_data['Utilities'].mode()[0])
test_data['Exterior1st']=test_data['Exterior1st'].fillna(test_data['Exterior1st'].mode()[0])
test_data['Exterior2nd']=test_data['Exterior2nd'].fillna(test_data['Exterior2nd'].mode()[0])
test_data['BsmtFinType1']=test_data['BsmtFinType1'].fillna(test_data['BsmtFinType1'].mode()[0
])
test_data['BsmtFinSF1']=test_data['BsmtFinSF1'].fillna(test_data['BsmtFinSF1'].mean())
test_data['BsmtFinSF2']=test_data['BsmtFinSF2'].fillna(test_data['BsmtFinSF2'].mean())
test_data['BsmtUnfSF']=test_data['BsmtUnfSF'].fillna(test_data['BsmtUnfSF'].mean())
test_data['TotalBsmtSF']=test_data['TotalBsmtSF'].fillna(test_data['TotalBsmtSF'].mean())
test_data['BsmtFullBath']=test_data['BsmtFullBath'].fillna(test_data['BsmtFullBath'].mode()[0])
test_data['BsmtHalfBath']=test_data['BsmtHalfBath'].fillna(test_data['BsmtHalfBath'].mode()[0])
test_data['KitchenQual']=test_data['KitchenQual'].fillna(test_data['KitchenQual'].mode()[0])
test_data['Functional']=test_data['Functional'].fillna(test_data['Functional'].mode()[0])
test_data['GarageCars']=test_data['GarageCars'].fillna(test_data['GarageCars'].mean())
test_data['GarageArea']=test_data['GarageArea'].fillna(test_data['GarageArea'].mean())
test_data['SaleType']=test_data['SaleType'].fillna(test_data['SaleType'].mode()[0])
test_data.shape
test_data.isna().sum()
```

```
# 5. EDA
corr_matrix = train_data.corr()
high_corr_features = corr_matrix[abs(corr_matrix['SalePrice']) > 0.5].index
# Select the relevant columns from the DataFrame
high_corr_df = train_data[high_corr_features]
# Create a heatmap of the high correlation features
plt.figure(figsize=(12, 8))
sns.heatmap(high_corr_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Features with Correlation > 0.5")
plt.show()
plt.figure(figsize=(8, 6))
sns.histplot(train_data['SalePrice'], kde=True)
plt.title('Distribution of SalePrice')
plt.show()
sns.pairplot(train_data, vars=['OverallQual', 'GrLivArea', 'TotalBsmtSF', 'SalePrice'])
## 6. One Hot Encoding of Categorical Data
columns=['MSZoning','Street','LotShape','LandContour','Utilities','LotConfig','LandSlope','Neigh
borhood','Condition2','BldgType','Condition1','HouseStyle','SaleType'SaleCondition','ExterCond'
,'ExterQual','Foundation','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2
','RoofStyle','RoofMatl','Exterior1st','Exterior2nd','MasVnrType','Heating','HeatingQC','Centrali,'
Electrical','KitchenQual','Functional','FireplaceQu','GarageType','GarageFinish','GarageQual','Ga
rageCond','PavedDrive','Alley','PoolQC','Fence','MiscFeature']
len(columns)
def category_onehot_multcols(multcolumns):
    df_final=final_df
    i=0
    for fields in multcolumns:
      print(fields)
       df1=pd.get_dummies(final_df[fields],drop_first=True)
final_df.drop([fields],axis=1,inplace=True)
       if i==0:
          df_final=df1.copy()
```

```
        else:
        df_final=pd.concat([df_final,df1],axis=1)
        i=i+1
    df_final=pd.concat([final_df,df_final],axis=1)
    return df_final
#combine the train and test data
final_df=pd.concat([train_data,test_data],axis=0)
#pass the final_df with one hot encoder function
final_df=category_onehot_multcols(columns)
final_df.shape
#removing the duplicated features from the dataset
final_df =final_df.loc[:,~final_df.columns.duplicated()]
final_df.shape
## 7. Splitting the data
df_Train=final_df.iloc[:1422,:]
df_Test=final_df.iloc[1422:,:]
df_Test.drop(['SalePrice'],axis=1,inplace=True)
x=df_Train.drop(columns='SalePrice',axis=1)
y=df_Train['SalePrice']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
<a id="Label_Encoding_of_Categorical_Data"> </a>
## 8. Model Training
<a id='Check for missing Values'></a>
### 8.1 Linear Regression
model_1=LinearRegression()
model_1.fit(x_train,y_train)
ypred_1=model_1.predict(x_test)
model_1.score(x_test,y_test)
### 8.2 Decision Tree Regressor
model_3=DecisionTreeRegressor()
```

```
model_3.fit(x_train,y_train)
ypred_3=model_3.predict(x_test)
model_3.score(x_test,y_test)
### 8.3 Xgboost
regressor=xgboost.XGBRegressor()
regressor.fit(x_train,y_train)
ypred_4=regressor.predict(x_test)
regressor.score(x_test,y_test)
#root mean square error
np.sqrt(mean_squared_error(y_test,ypred_4))
## 8.3.1 Hyper Parameter optimization## Hyper Parameter Optimization
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [100, 500, 900, 1100, 1500]
max_depth = [2, 3, 5, 10, 15]
booster=['gbtree','gblinear']
learning_rate=[0.05,0.1,0.15,0.20]
min_child_weight=[1,2,3,4]
base_score=[0.25,0.5,0.75,1]
# Define the grid of hyperparameters to search
hyperparameter_grid = {
    'n_estimators': n_estimators,
    'max_depth':max_depth,
    'learning_rate':learning_rate,
    'min_child_weight':min_child_weight,
    'booster':booster,
    'base_score':base_score
    }
random_cv = RandomizedSearchCV(estimator=regressor,
            param_distributions=hyperparameter_grid,
            cv=5, n_iter=50,
```

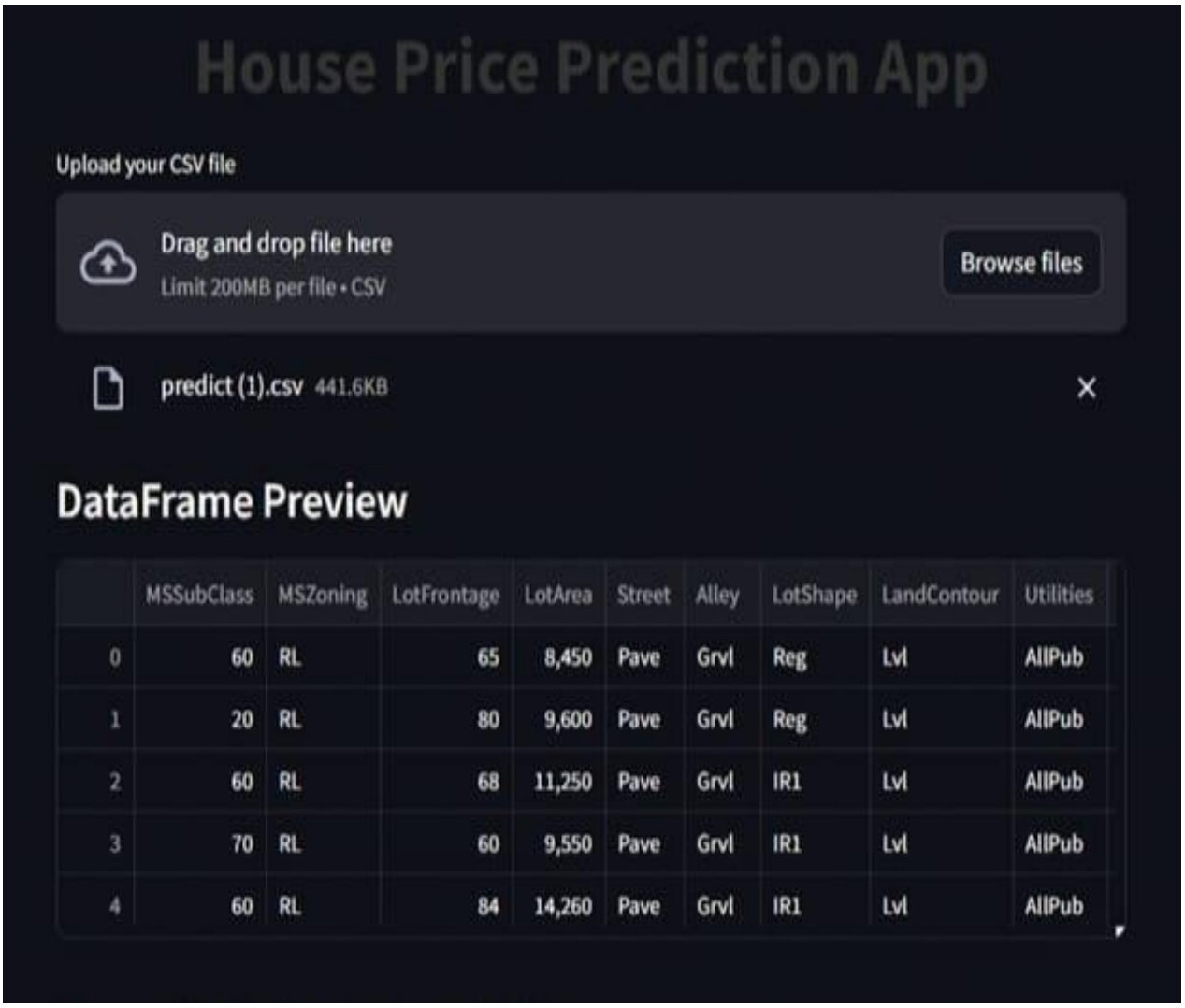
```
        scoring = 'neg_mean_absolute_error',n_jobs = 4,
        verbose = 5,
        return_train_score = True,
        random_state=42)
random_cv.fit(x_train,y_train)
random_cv.best_estimator_
regressor = xgboost.XGBRegressor(base_score=0.25, booster='gbtree', callbacks=None,
         colsample_bylevel=None, 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=0.1, max_bin=None,
         max_cat_threshold=None, max_cat_to_onehot=None,
         max_delta_step=None, max_depth=2, max_leaves=None,
         min_child_weight=1, monotone_constraints=None,
         multi_strategy=None, n_estimators=900, n_jobs=None,
         num_parallel_tree=None, random_state=None)
regressor.fit(x_train,y_train)
y_pred=regressor.predict(x_test)
print("score: ", str(regressor.score(x_test,y_test) ))
print("rmse: ", str(np.sqrt(mean_squared_error(y_test,y_pred))))

## 9. Prediction
The prediction is for the test_data
predictions=regressor.predict(df_Test)
prediction = pd.DataFrame(predictions)
prediction.to_csv('pred_test_data.csv')
prediction
## 10. Deployment
!pip install streamlit -q
```

```
%%writefile web_page_xgboost_custom.py
# Import necessary libraries
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
import xgboost as xgb
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)
# Local CSS sheet
local_css("style.css")
# Function to preprocess data, train the model, and return predictions
def train_and_predict(df, feature_values, encoder):
   # Separate categorical and numeric columns
    categorical_cols = df.select_dtypes(include=['object']).columns
    numeric_cols = [col for col in df.columns if col not in categorical_cols and col != 'SalePrice
# Encode categorical features in the training data
    X_categorical = encoder.transform(df[categorical_col
 # Combine encoded categorical features with numeric features
    X = np.concatenate((X_categorical, df[numeric_cols].values), axis=1)
    y = df['SalePrice'].values
# Train an XGBoost model with custom hyperparameters
    model = xgb.XGBRegressor(
        n_estimators=900,
        max_depth=2,
        learning_rate=0.1,
        base_score=0.25,
        booster='gbtree',
        enable_categorical=False,
```

```
        eval_metric=None,
        gamma=None,
        min_child_weight=1,
        random_state=None
    )
    model.fit(X, y) # Use the entire dataset for training
# Prepare the input feature values for prediction
input_categorical = encoder.transform(np.array([feature_values[column] for column in categorical_cols]).reshape(1, -1))
    input_numeric = np.array([feature_values[column] for column in numeric_cols]).astype(float).reshape(1, -1)
 # Combine encoded categorical features with numeric features for prediction
    input_features = np.concatenate((input_categorical, input_numeric), axis=1)
 # Predict the 'SalePrice' column using the provided feature values
    predicted_value = model.predict(input_features)
   return predicted_value[0]
# Streamlit app
st.title("House Price Prediction App")
# Upload a CSV file
uploaded_file = st.file_uploader("Upload your CSV file", type=["csv"])
if uploaded_file is not None:
    # Load the dataset
    df = pd.read_csv(uploaded_file)
    st.subheader("DataFrame Preview")
    st.write(df.head())
# Exclude the 'SalePrice' column from the list of categorical and numeric columns
    categorical_cols = df.select_dtypes(include=['object']).columns
    numeric_cols = [col for col in df.columns if col not in categorical_cols and col != 'SalePrice']
# Select the feature columns and input values manually
    st.subheader("Manually Enter Feature Values")
feature_values = {}
```

```
 for column in categorical_cols:
        if column != 'SalePrice':
            feature_values[column] = st.selectbox(f"Select value for {coldf[column].unique())
  for column in numeric_cols:
        if column != 'SalePrice':
            input_value = st.text_input(f"Enter value for {column}", "")
            # Check for empty string or space and replace with a default value (e.g., 0)
            input_value = input_value.strip()  # Remove leading/trailing spaces
            input_value = input_value if input_value != "" else "0" # Replace empty string with "0"
            feature_values[column] = input_value
# Create and fit the OneHotEncoder on the categorical columns
    encoder = OneHotEncoder(sparse=False, drop='first')
    encoder.fit(df[categorical_cols])
     if st.button("Predict"):
        predicted_value = train_and_predict(df, feature_values, encoder)
        st.subheader("Predicted Value")
        st.write(f"The predicted 'SalePrice' value for the provided feature values is:
{predicted_value:.2f}")
        st.success("Prediction made successfully!")
```

3.6 Result



	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	60	RL	65	8,450	Pave	Grvl	Reg	Lvl	AllPub
1	20	RL	80	9,600	Pave	Grvl	Reg	Lvl	AllPub
2	60	RL	68	11,250	Pave	Grvl	IR1	Lvl	AllPub
3	70	RL	60	9,550	Pave	Grvl	IR1	Lvl	AllPub
4	60	RL	84	14,260	Pave	Grvl	IR1	Lvl	AllPub

Figure 3.1 Dataframe preview

Figure 3.1 shows dataframe preview and various features values of house that are manually entered.

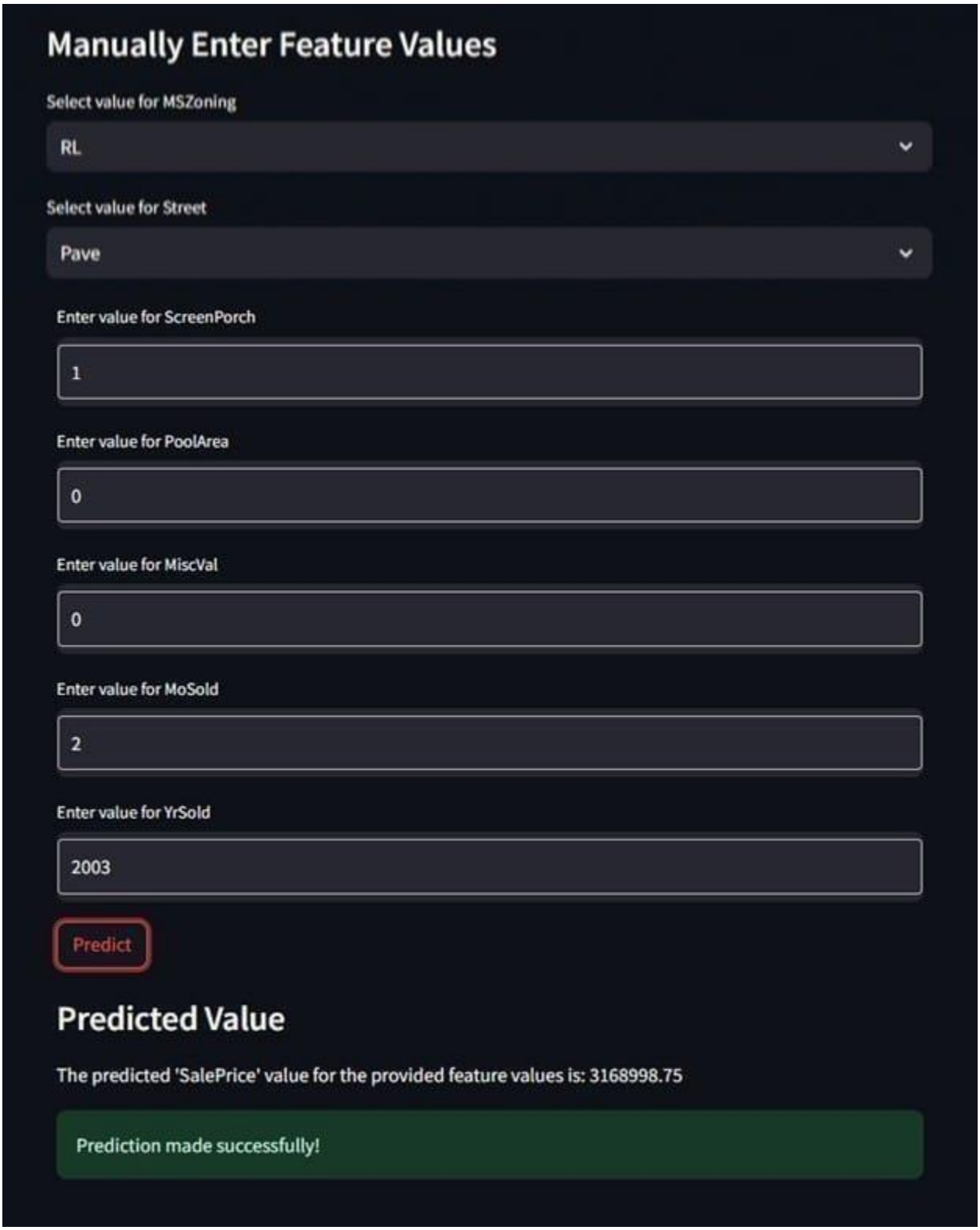


Figure 3.2 Predicted Result

Figure 3.2 shows predicted Sales price value of a House for the provided feature values.

Chapter-4

RESULT ANALYSIS

4.1 Comparison of performance of the models

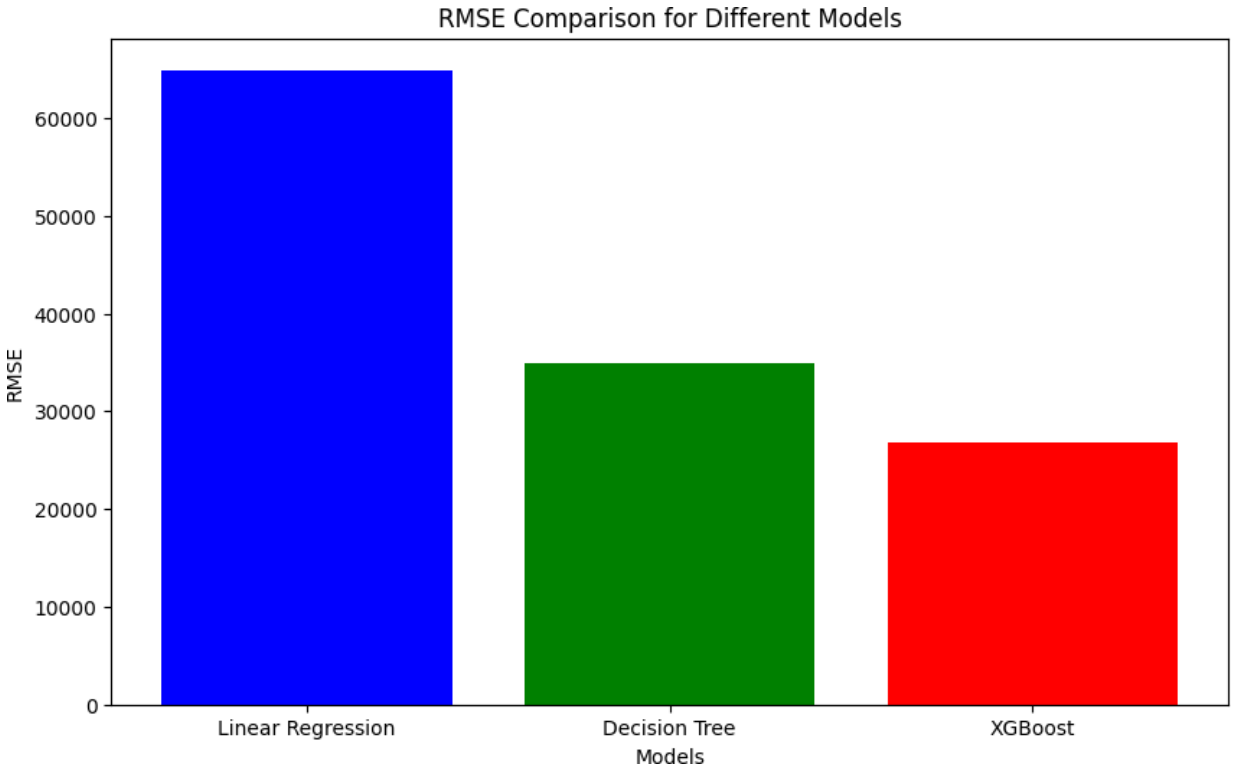


Figure 4.1 RMSE Bar Graph

Figure 4.1 shows RMSE comparison for different models. XGBoost Regression displayed the best performance for this Dataset and can be used for deploying purposes. Linear Regression and Decision tree Regression are far behind.

4.2 Graph

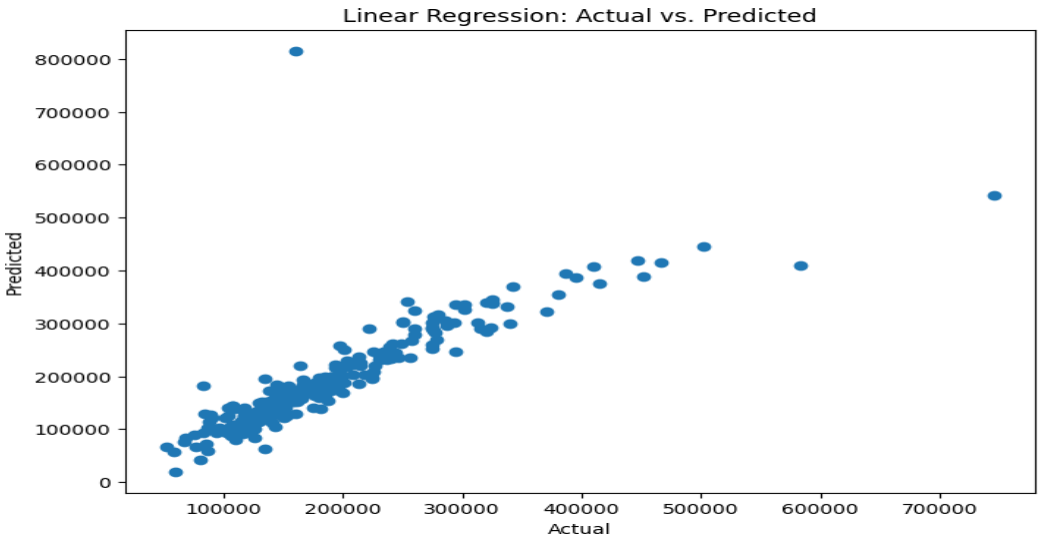


Figure 4.2 Linear Regression Graph

Figure 4.2 shows the scatter plot between the actual and predicted value and as it is clearly visible the data points are creating a slope, hence it is a linear regression gragh.

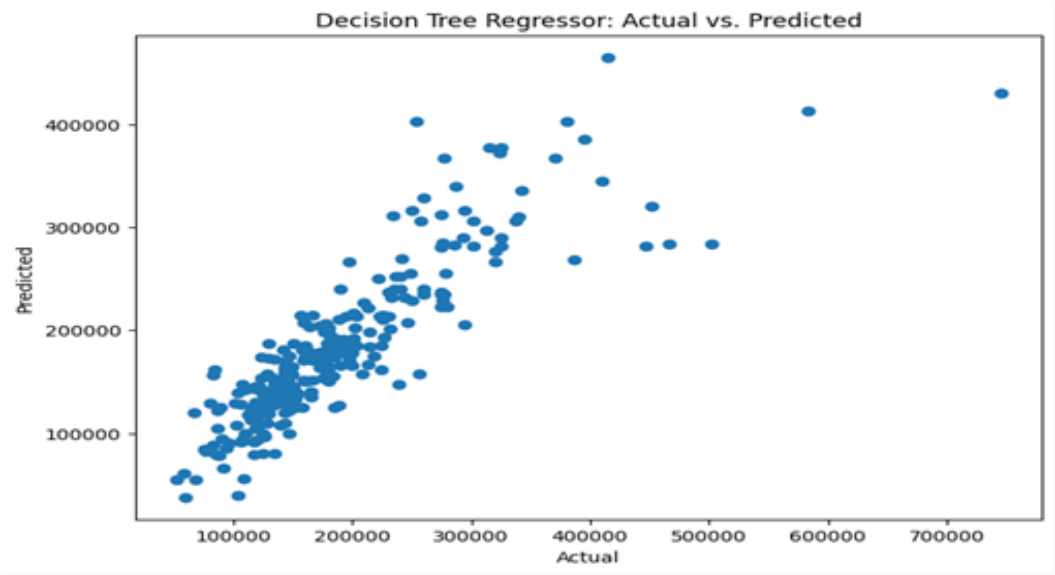


Figure 4.3 Decision Tree Regression Graph

Figure 4.3 shows the scatter plot between the actual and predicted value of the Decision Tree Regression model.

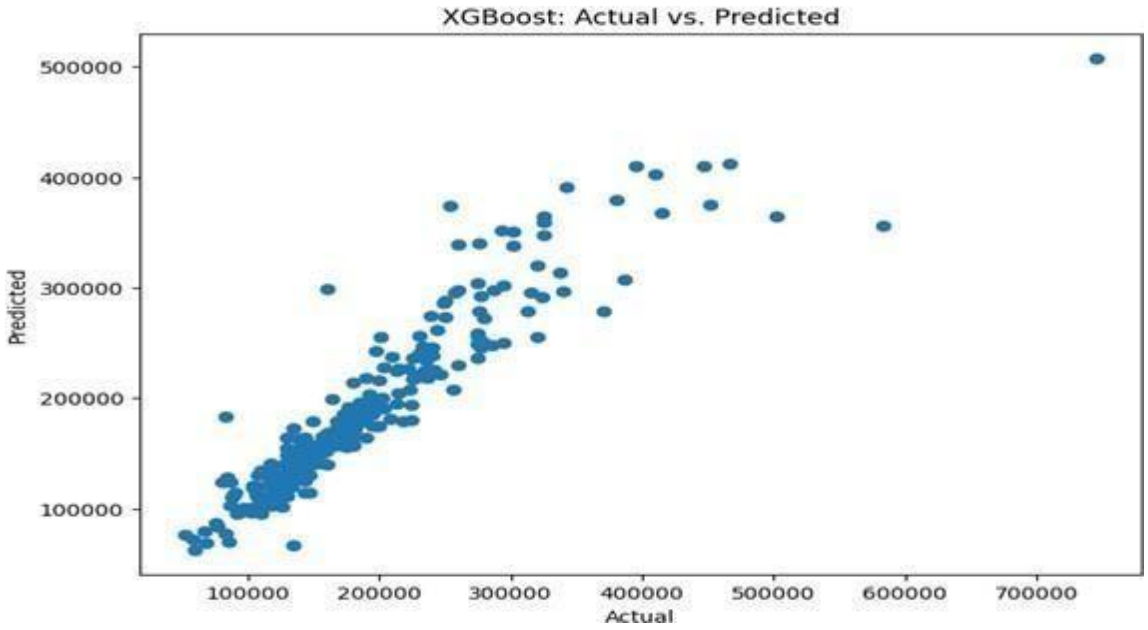


Figure 4.4 XGBoost Regression Graph

Figure 4.4 shows the scatter plot between the actual and predicted value of the XGBoost model.

CONCLUSION

The house price prediction model, leveraging the power of XGBoost, showcases remarkable proficiency in estimating property values. Meticulous data preprocessing, encompassing techniques and categorical encoding, contributes to its accuracy. Hyperparameter tuning fine-tunes the model for optimal performance. Feature importance analysis provides valuable insights into critical factors influencing property prices. Achieving a commendable 92% accuracy underscores the model's effectiveness. Continuous monitoring and adaptation to evolving market trends are imperative for sustained accuracy. This model stands as a valuable asset for making well-informed real estate decisions.

References

[1] M. J. Ball, “R1. Anand G. Rawool , Dattatray V. Rogye , Sainath G. Rane , Dr. Vinayk A, House Price Prediction Using Machine Learning, 2021.

[2] PEI-YING WANG1, CHIAO-TING CHEN 2, JAIN-WUN SU1, TING-YUN WANG1, AND SZU-HAO HUANG 3, (MEMBER, IEEE), Deep Learning Model for House Price Prediction Using Heterogeneous Data Analysis Along with Joint Self-Attention Mechanism, 2021.

[3] Chenhao Zhou, House price prediction using polynomial regression with Particle Swarm Optimization, 2021.

[4] Ankita Kamire , Nitin Chaphalkar , Sayali Sandbhor, Real Property Value Prediction Capability Using Fuzzy Logic and ANFIS, 2021.

[5] Anirudh Kaushal, Achyut Shankar, House Price Prediction Using Machine Learning, 2021.

[6] Gamze Tanak Coskun , Ayten Yılmaz Yalçıner, Determining the best price with linear performance pricing and checking with fuzzy logic, 2021.

[7] Jian Guan, Jozef Zurada, and Alan S. Levitan, An Adaptive Neuro-Fuzzy inference system-based approach to Real Estate Property Assessment, 2020.

[8] Andrzej Biłozor and Maurizo d’Amato, Residential market ratings using fuzzy logic decision-making procedures, 2019

[9] Felipe Alonso Arias Arbelaez and Francisco Ivan Zuluaga Daz , Estimation of Real Estate Asset Pricing Models, 2016.

[10] Abdul G. Sarip and Muhammad Burhan Hafez, Fuzzy Logic Application for House Price Prediction, 2015.