Project Title

Data-Driven Housing Analysis using EDA, Visualization, and Machine Learning Algorithm for Price Prediction.

Project Submission on

31-AUG-2023

Name: K.SaiVinay

Pin No: 21X05A6731

Department

Computer Science and Data Science

College

Narsimha Reddy Engineering College

Program: Skill Development Internship

Company

Abstract

In this project, we exploratory data analysis (EDA) on a housing price detection dataset. Using the seaborn and matplotlib libraries, we visualized the data to gain further insights. We explored the distribution of various features such as bedrooms, bathrooms, and parking using histograms. Additionally, we examined the relationship between the number of bedrooms and the price of properties using scatter plots. These visualizations helped us understand the data and identify any patterns or trends.

Next, we applied the concepts of linear and logistic regression to build machine learning models for predicting housing prices. By training these models on the dataset, we were able to make accurate predictions based on the input features. To evaluate the performance of our models, we compared the accuracy of different machine learning algorithms such as K-Nearest Neighbors (KNN) and Decision Trees (DT). This allowed us to select the most effective algorithm for our prediction task.

To validate the final output of our machine learning models, we created a confusion matrix. This matrix provided a detailed breakdown of the true positives, true negatives, false positives, and false negatives. By analyzing these metrics, we were able to assess the accuracy and performance of our models. Finally, we generated visualizations and insights based on the analyzed data, which formed the basis for our recommendations. These insights provided valuable information for decision-making and further analysis of the housing dataset.

Overall, this project involved conducting exploratory data analysis, visualizing the data using seaborn and matplotlib, applying linear and logistic regression for prediction, evaluating different machine learning algorithms, creating a confusion matrix for validation, and generating visualizations and insights for recommendations based on the analyzed data.

Project Title

Data-Driven Housing Analysis using EDA, Visualization, and Machine Learning Algorithm for Price Prediction.

In this data-driven housing analysis project, we dive into the vast world of real estate with the power of data. By employing techniques like exploratory data analysis (EDA), visualization, and machine learning (ML), we aim to unravel the intricate patterns and trends in housing data to make accurate price predictions.

The housing market is a complex ecosystem influenced by numerous factors, such as location, property characteristics, and market conditions. Through our project, we strive to harness the potential of data to gain deep insights into these factors and their impact on housing prices. By performing EDA, we can uncover hidden relationships and correlations within the data. Visualizations help us effectively communicate these insights, making complex information more accessible and understandable. Lastly, leveraging ML algorithms empowers us to build predictive models that can forecast housing prices with precision and accuracy.

By combining the power of EDA, visualization, and ML, our project equips individuals and organizations with the knowledge and tools to make informed decisions in the dynamic and ever-changing housing market.

Tools and Technology:

1. Advance Excel
2. Power Bi

1. Advance Excel:

Advanced Excel is a powerful tool that allows you to take your data manipulation and analysis skills to the next level. With advanced Excel, you can dive into complex formulas and functions, perform detailed calculations, and automate tasks to save time. It also offers a range of data analysis tools like filters, sorting, and conditional formatting, which help you explore and analyze data effectively. Pivot tables are another fantastic feature that allows you to summarize and analyze large datasets effortlessly. Macros enable you to automate repetitive tasks, increasing efficiency. Advanced Excel also offers features like data validation, goal seek, and scenario manager, which enhance your ability to analyze and make data-driven decisions. So, by mastering advanced Excel, you can become a data wizard, manipulating and analyzing data with ease.

2. Power Bi:

Power BI is an incredible business intelligence tool developed by Microsoft. It allows you to connect to various data sources, transform and shape the data, and create interactive visualizations and reports. With Power BI, you can gain valuable insights from your data, make data-driven decisions, and share your findings with others. It offers a user-friendly interface and a wide range of visualization options to present data in a compelling and meaningful way. Power BI also provides advanced features like data modeling, DAX formulas, and custom visualizations, allowing you to create sophisticated and interactive dashboards. It's a powerful tool for data analysis and reporting, empowering businesses to extract valuable insights from their data.

Organization Names:

1. IBM India
2. Myntra fashion private limited
3. Scatter pie chart analysis

1. **IBM India:**

IBM India, a subsidiary of the renowned tech giant IBM, holds a significant position in India's technology sector. Through its R&D centers, strategic collaborations with academia and impactful corporate social responsibility endeavors, it actively contributes to technological advancement, skill development, and societal betterment. With a diverse workforce and a focus on innovation, IBM India continues to be a driving force in shaping the country's technological landscape.

2. **Myntra fashion private limited**

Myntra Fashion Private Limited is a well-known organization in the fashion industry. They offer a wide range of clothing, accessories, and footwear for men, women, and kids. Myntra is known for its trendy and fashionable products, and they collaborate with various brands and designers to bring the latest fashion to their customers. They also have an online platform where customers can browse and purchase their products conveniently. Myntra is a popular choice for fashion enthusiasts and has made a significant impact in the fashion retail industry.

3. **Scatter pie analysis organization:**

Scatter pie analysis is a technique used to visualize and analyze categorical data in a scatter plot format. It combines the features of a scatter plot and a pie chart to represent the relationship between two categorical variables. Each point on the scatter pie plot represents a category combination, and the size of the pie chart within each point represents the proportion of that category combination. This analysis can help identify patterns, clusters, or correlations between different categories, providing insights into the relationships within the data. It's a unique and visually appealing way to explore categorical data

Problem Statement:

The goal of this project is to analyze a housing price detection dataset and perform various tasks to gain insights and build machine learning models. The tasks include:

Task 1:

Conduct exploratory data analysis (EDA) on the housing price detection dataset.

Task 2:

Utilize the seaborn and matplotlib libraries to visualize the data and gain further insights.

Task 3:

Apply the concepts of linear and logistic regression to build machine learning models for predicting housing prices.

Task 4:

Evaluate the accuracy of different machine learning algorithms such as KNN and DTfor predicting housing prices.

Task 5:

Create a confusion matrix to validate the final output of the machine learning models.

Task 6:

Generate visualizations and insights for recommendations based on the analyzed data.

By completing these tasks, we aim to gain a comprehensive understanding of the housing price dataset, build accurate prediction models, and provide valuable recommendations and insights.

Task 1:

Conduct exploratory data analysis (EDA) on the housing price detection dataset.

Introduction:-

Embarking on an in-depth exploration of the housing price detection dataset, the process of Exploratory Data Analysis (EDA) unveils critical insights hidden within the data's intricacies. EDA, a foundational step in data analysis, delves into patterns, trends, outliers, and relationships among variables. By employing various statistical and visual techniques, EDA allows us to decipher the dataset's story, enabling informed decision-making and the formulation of subsequent analysis strategies. This journey through EDA not only unearths the underlying structure of the housing data but also sets the stage for comprehensive understanding and predictive modeling in the realm of real estate pricing.

Exploratory Data Analysis (EDA) serves the purpose of comprehensively understanding a dataset's structure, patterns, and underlying insights. It involves a series of tasks to extract meaningful information from raw data:

1. **Data Overview:**

   Begin by acquiring an overview of the dataset, its variables, and their types.

2. **Data Cleaning:**

   Identify and handle missing values, outliers, and errors that could impact the analysis.

3. **Descriptive Statistics:**

   Compute summary statistics to grasp central tendencies, dispersions, and distributions of the data.

4. **Data Visualization:**
   Univariate Analysis: Visualize individual variables to understand their distributions and identify anomalies.
   Bivariate Analysis: Explore relationships between pairs of variables to uncover correlations and potential associations.
   Multivariate Analysis: Examine interactions among multiple variables to unveil complex patterns.
5. **Feature Engineering:**

   Create new variables or transform existing ones to enhance the dataset's representation.

6. **Correlation Analysis:**

   Evaluate correlations between variables to reveal potential dependencies.

7. **Dimensionality Reduction:**

Reduce data complexity using techniques like PCA to retain key information.

8. **Handling Categorical Data:**

Analyze and visualize categorical variables to reveal insights about their distributions and relationships.

9. **Outlier Detection:**

Identify outliers that could skew analysis or modeling results.

10. **Data Distribution:**

Assess if the data follows a specific distribution, which can impact modeling choices.

11. **Missing Data Imputation:**

Address missing data by imputing or handling them appropriately.

12. **Data Transformation:**

Apply transformations to make data more suitable for analysis or modeling.

13. **Hypothesis Testing:**

Validate assumptions or identify significant differences using statistical tests.

14. **Interactive Exploration:**

Employ interactive tools to facilitate dynamic exploration and visualization.

In essence, EDA transforms raw data into comprehensible insights, guiding subsequent steps such as model selection, feature engineering, and hypothesis testing.

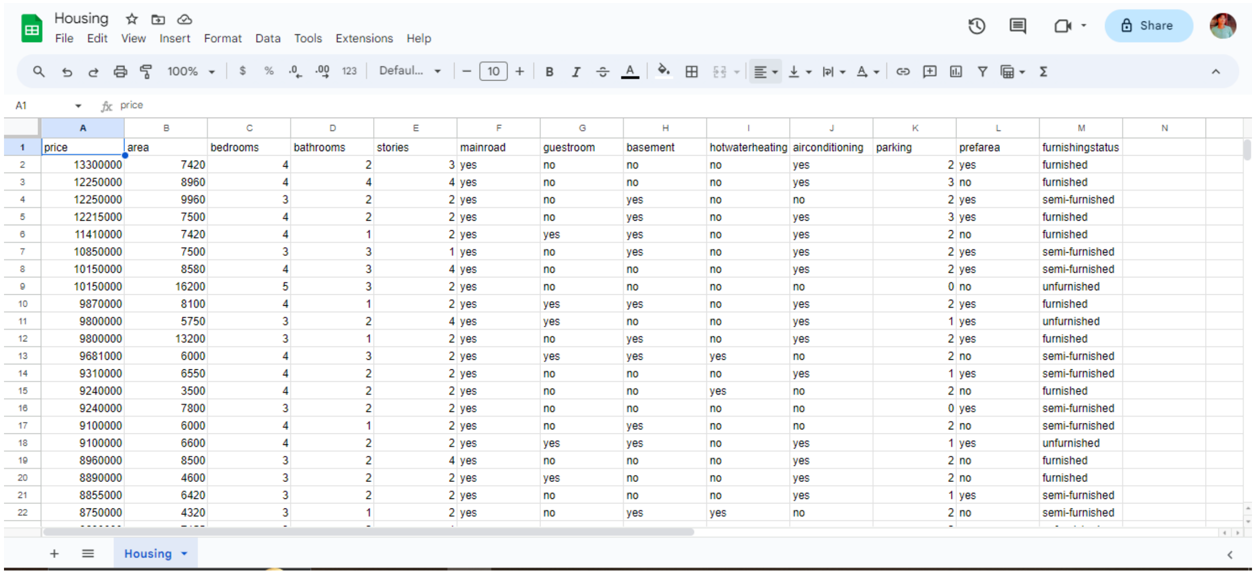
Housing Data:

Raw Data

Raw data refers to the original, unprocessed data that has not undergone any cleaning, formatting, or manipulation. It is the data as it is collected or obtained without any modifications or adjustments.

Clean Data:

Clean data refers to the processed and formatted data that has been carefully reviewed, validated, and standardized. It is free from errors, inconsistencies, and missing values, making it suitable for analysis and interpretation. Clean data is essential for accurate and reliable insights.



price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
10850000	7500	3	3	1	yes	no	yes	no	yes	2	yes	semi-furnished
10150000	8580	4	3	4	yes	no	no	no	yes	2	yes	semi-furnished
10150000	16200	5	3	2	yes	no	no	no	no	0	no	unfurnished
9870000	8100	4	1	2	yes	yes	yes	no	yes	2	yes	furnished
9800000	5750	3	2	4	yes	yes	no	no	yes	1	yes	unfurnished
9800000	13200	3	1	2	yes	no	yes	no	yes	2	yes	furnished
9681000	6000	4	3	2	yes	yes	yes	yes	no	2	no	semi-furnished
9310000	6550	4	2	2	yes	no	no	no	yes	1	yes	semi-furnished
9240000	3500	4	2	2	yes	no	no	yes	no	2	no	furnished
9240000	7800	3	2	2	yes	no	no	no	no	0	yes	semi-furnished
9100000	6000	4	1	2	yes	no	yes	no	no	2	no	semi-furnished
9100000	6600	4	2	2	yes	yes	yes	no	yes	1	yes	unfurnished
8960000	8500	3	2	4	yes	no	no	no	yes	2	no	furnished
8890000	4600	3	2	2	yes	yes	no	no	yes	2	no	furnished
8855000	6420	3	2	2	yes	no	no	no	yes	1	yes	semi-furnished
8750000	4320	3	1	2	yes	no	yes	yes	no	2	no	semi-furnished

Here,

The housing data you have is already clean. It saves us time and allows us to focus directly on analyzing the data and gaining insights. Let's dive into the analysis and uncover interesting patterns in the housing price detection dataset!

To Convert Raw Data into Clean Data

In Excel, null values are typically represented as empty cells. However, sometimes they might be represented with other placeholders like "N/A," "NA," "None," or similar text strings.

1. Extraction Transformation Load
2. Data Cleaning
   - Three main factors
     i. Cleaning
     ii. Filtering
     iii. Gathering
   - Check type casting
   - Conversion
   - Checking Null values

Extraction Transformation Load (ETL):

This refers to the process of extracting data from various sources, transforming it to fit specific requirements, and loading it into a target system or database.

Data Cleaning:

This involves preparing and refining the raw data for analysis. The key tasks include:

- **Cleaning:** Correcting errors, inconsistencies, and inaccuracies in the data.
- **Filtering:** Selecting specific data based on certain criteria.
- **Gathering:** Collecting data from various sources and preparing it for analysis.

Check Type Casting:

Ensuring data types are appropriate and consistent.

Conversion:

Transforming data formats or units for uniformity.

Checking Null Values:

Addressing or managing missing data

1. Blank Cells:

- Blank cells usually represent null values in Excel. To find them:
- Select the column or range of cells where you suspect null values might be.
- Go to the "Home" tab on the Excel ribbon.
- Look in the "Editing" group for the "Find & Select" dropdown. Click on it.
- Choose "Go To Special..."
- In the "Go To Special" dialog box, select "Blanks" and click "OK." Excel will select all blank cells in the selected range.

2. Text-based Null Values:

- If null values are represented using specific text strings (e.g., "N/A," "NA," "None"), you can find them using Excel's Find function:
- Press `Ctrl + F` or go to the "Home" tab and click on "Find & Select" > "Find."
- Enter the text string you suspect represents null values in the "Find what" field.
- Click "Find All" to see a list of cells containing that text.

Task -2

Utilize the seaborn and matplotlib libraries to visualize the data and gain further insights.

Seaborn Library:

Seaborn is a fantastic library for data visualization in Python. It provides a high-level interface that allows you to create stunning and informative statistical graphics. With Seaborn, you can easily generate various types of plots, such as scatter plots, line plots, bar plots, histograms, and more. It also offers built-in themes and color palettes to enhance the visual appeal of your plots. Seaborn is particularly useful for exploratory data analysis (EDA) and can help you uncover patterns, relationships, and insights in your data. It's definitely worth checking out if you're interested in creating visually appealing and informative visualizations.

Seaborn has many uses and it helps us in various ways:

1. **Data Visualization:**
   Seaborn allows us to create visually appealing plots to explore and understand our data better.

2. **Statistical Analysis:**
   Seaborn provides functions to visualize statistical models and conduct statistical tests, making it easier to analyze and interpret data.
3. **Categorical Data Analysis:**
   Seaborn offers plots like bar plots and count plots to analyze and compare categorical variables in our dataset.
4. **Relationship Analysis:**
   Seaborn's scatter plots, regression plots, and correlation matrices help us uncover relationships and trends between variables.
5. **Distribution Analysis:**
   Seaborn provides tools for visualizing and analyzing data distributions, such as histograms and box plots.
6. **Time Series Analysis:**
   Seaborn offers functionality to visualize and analyze time series data, helping us understand patterns and trends over time.

Seaborn simplifies the process of data visualization, enhances data analysis, and helps us communicate our findings effectively. It's a valuable tool for anyone working with data

Matplotlib:

Matplotlib is another popular data visualization library in Python. It provides a wide range of functions and tools for creating various types of plots, such as line plots, bar plots, scatter plots, histograms, and more. Matplotlib offers a lot of flexibility and customization options, allowing you to create visually appealing and informative visualizations. It is often used in conjunction with other libraries like NumPy and Pandas to analyze and visualize data. Matplotlib is a powerful tool for data visualization and is widely used in the data science community.

Matplotlib has several uses and it helps us in various ways:

1. **Data Visualization:**
   Matplotlib allows us to create a wide range of plots, including line plots, scatter plots, bar plots, histograms, and more. It helps us visually explore and present our data.

2. **Customization Options:**
   Matplotlib provides extensive customization options, allowing us to control aspects like colors, labels, axes, and annotations. This helps us create visually appealing and tailored visualizations.
3. **Integration with Other Libraries:**
   Matplotlib integrates well with other libraries like NumPy and Pandas, making it easy to visualize data from these libraries. It enhances our ability to analyze and interpret data effectively.
4. **Publication-Quality Plots:**
   Matplotlib enables the creation of high-quality plots suitable for publications and presentations. It provides tools for adjusting plot aesthetics and exporting plots in various formats.
5. **Interactive Visualization:**
   Matplotlib supports interactive features like zooming, panning, and adding interactivity to plots. This enhances the user experience and allows for more in-depth data exploration.

Overall, Matplotlib is a versatile library that empowers us to visualize data, customize plots, and create visually appealing and informative visualizations. It is a valuable tool for data analysis and presentation.

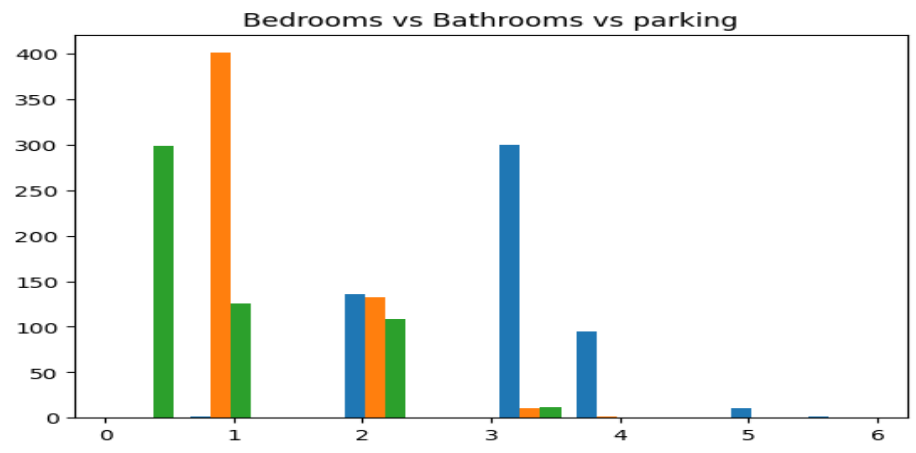
Visualization Code

```
#import the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#read the dataset
data = pd.read_csv("/content/Housing.csv")
data
```

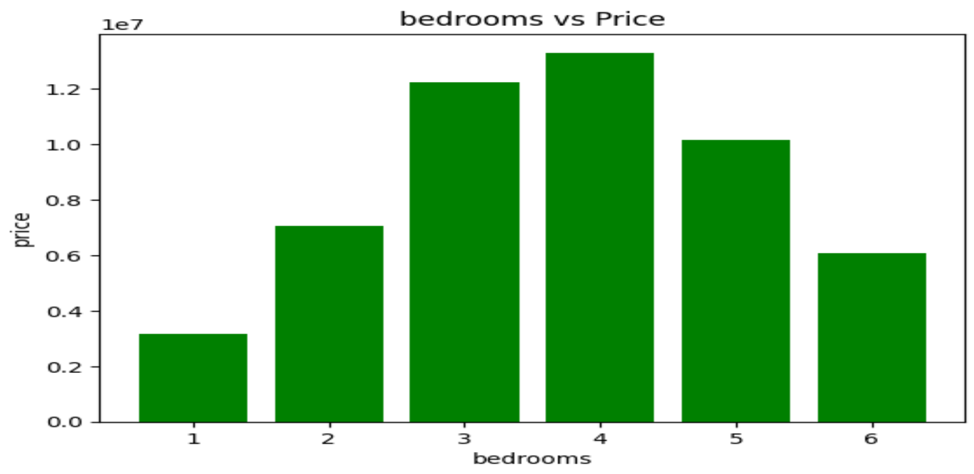
```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
```

```
2    12250000  9960        3         2       2     yes       no      yes
3    12215000  7500        4         2       2     yes       no      yes
4    11410000  7420        4         1       2     yes      yes      yes
..       ...   ...        ...       ...      ...     ...      ...      ...
540   1820000  3000        2         1       1     yes       no      yes
541   1767150  2400        3         1       1      no       no       no
542   1750000  3620        2         1       1     yes       no       no
543   1750000  2910        3         1       1      no       no       no
544   1750000  3850        3         1       2     yes       no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              ...             ...      ...      ...              ...
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished

[545 rows x 13 columns]
```

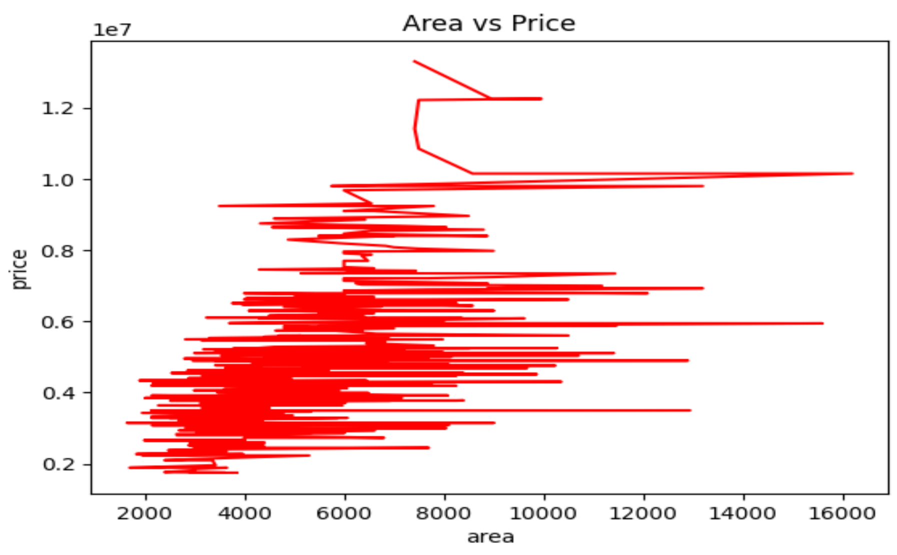
```
#plot the histogram
a=data[["bedrooms","bathrooms","parking"]]
plt.hist(a,bins=10)
plt.title("Bedrooms vs Bathrooms vs parking")
plt.show()
```



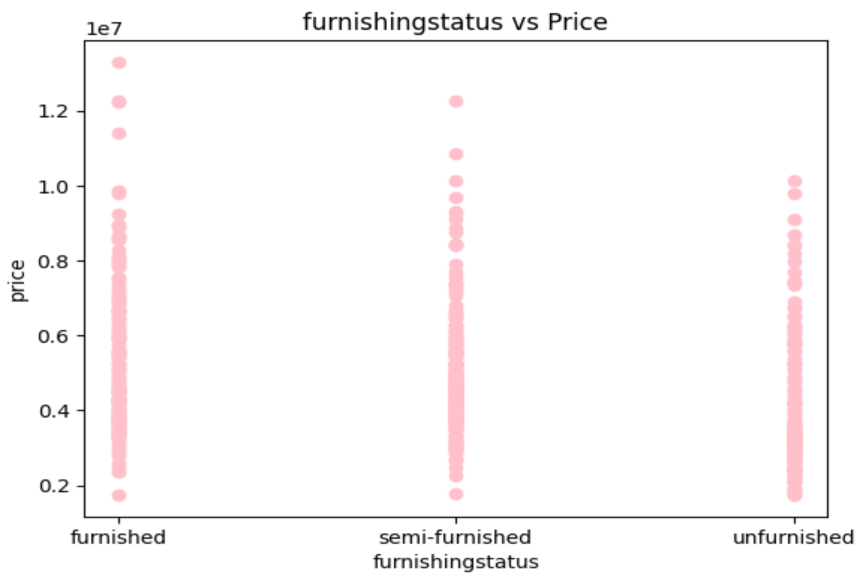
```
x=data.bedrooms
y=data.price
plt.bar(x,y,color='green')
plt.title("bedrooms vs Price")
plt.xlabel("bedrooms")
plt.ylabel("price")
plt.show()
```



```
x=data.area
y=data.price
plt.plot(x,y,color='r')
plt.title("Area vs Price")
plt.xlabel("area")
plt.ylabel("price")
plt.show()
```



```
x=data.furnishingstatus
y=data.price
plt.scatter(x,y,color='pink')
plt.title("furnishingstatus vs Price")
plt.xlabel("furnishingstatus")
plt.ylabel("price")
plt.show()
```



Insights:

- Histogram
- Bar plot
- Line plot
- Scatter plot

Histogram:

The histogram plot of "Bedrooms vs. Bathrooms vs. Parking", we can gather some insights:

1. **Bedrooms Distribution:**

   The histogram displays bedroom count distribution, highlighting common values and helping identify typical bedroom numbers in the dataset.

2. **Bathrooms Distribution:**

   The histogram shows the bathroom counts' distribution, helping us identify patterns and understand the bathroom configurations in the housing data.

3. **Parking Distribution:**

   The histogram represents parking availability distribution, giving insights into parking arrangements and facilities in the housing data.

- **Bedrooms:** Most properties have 2 to 4 bedrooms, with fewer having more.
- **Bathrooms:** Properties mostly feature 1 to 3 bathrooms, fewer with more.
- **Parking:** 1 to 2 parking spaces are common, while fewer or more are rarer.

Bar Plot:

The bar plot "Bedrooms vs. Price" illustrates the relationship between the number of bedrooms and property prices.

- The plot shows how property prices relate to the number of bedrooms.
- More bedrooms generally correspond to higher prices, indicating a positive correlation.
- Outliers suggest exceptions, potentially due to other factors.
- Consider location, size, and amenities; they're not factored in here.

- ➢ Buyers can estimate price ranges by bedrooms, aiding sellers in competitive pricing.
- ➢ While bedroom-count impacts prices, comprehensive analysis includes more variables.

Line Plot:

- ➢ There is a positive relationship between the area of properties and their prices.
- ➢ As the area increases, the price tends to be higher.
- ➢ The size of the property is an important factor in determining its price.
- ➢ Properties with larger areas are generally more expensive than those with smaller areas.

Scatter Plot:

- ➢ The scatter plot shows the relationship between furnishing status and the price of properties.
- ➢ There are three categories of furnishing status: furnished, semi-furnished, and unfurnished.
- ➢ Properties that are furnished tend to have higher prices compared to semi-furnished and unfurnished properties.
- ➢ Semi-furnished properties have moderate prices, while unfurnished properties have lower prices.
- ➢ Furnishing status is an important factor in determining the price of a property.

Recommendations of Insights

Histogram (Bedrooms, Bathrooms, Parking):

Property developers should focus on building properties with 2 to 4 bedrooms, as they are in higher demand.

Consider offering properties with 1 to 3 bathrooms, as they are more aligned with the preferences of potential buyers.

Properties with 1 to 2 parking spaces might be more appealing, so urban developers should consider providing parking options for residents.

Bar Plot (Bedrooms vs. Price):

For buyers seeking a higher number of bedrooms, be prepared for an increase in property prices due to the positive correlation.

Sellers can set premium prices for properties with more bedrooms, catering to buyers willing to invest in larger spaces.

Remember to factor in other variables beyond bedrooms, like location and amenities, for accurate pricing.

Line Plot (Area vs. Price):

Buyers looking for more spacious properties should expect a corresponding increase in price, given the positive correlation between area and price.

Sellers can capitalize on this relationship by justifying higher prices for larger properties.

While area is important, also consider other variables to determine the overall value of a property.

Scatter Plot (Furnishing Status vs. Price):

Sellers with fully furnished properties can set higher prices, as these properties tend to command premium prices.

Semi-furnished properties can appeal to a wider range of buyers, offering a balance between price and convenience.

Unfurnished properties might attract price-conscious buyers, and sellers can adjust prices accordingly.

These recommendations can help guide decision-making for both buyers and sellers in the real estate market, taking into account various aspects such as bedrooms, bathrooms, parking, area, and furnishing status.

Task-3

Apply the concepts of linear and logistic regression to build machine learning models for predicting housing prices.

To predict housing prices, we can use linear regression and logistic regression models:

1. **Linear Regression:**
   This model predicts the continuous numerical value of the housing price based on input features like the number of bedrooms, square footage, location, etc. It finds the best-fit line that minimizes the difference between the predicted and actual prices.
   It is useful for predicting numerical values, such as housing prices, based on input features. Linear regression helps us understand the relationship between the independent variables (features) and the dependent variable (target). It provides insights into how changes in the input variables impact the predicted output. This model is widely used in various fields, including finance, economics, and social science
2. **Logistic Regression:**
   Although commonly used for classification tasks, logistic regression can also be used for binary prediction, such as predicting whether a house will be sold above or below a certain price threshold. It estimates the probability of a binary outcome based on input features.
   It is beneficial for binary classification tasks, where the outcome is either one of two categories. Logistic regression helps us estimate the probability of an event occurring based on input variables. It is commonly used in areas like healthcare (predicting disease presence), marketing (predicting customer churn), and fraud detection (predicting fraudulent transactions). Logistic regression provides interpretable results and can help identify the key factors influencing the binary outcome.

Both models require a labeled dataset with features (e.g., number of bedrooms, square footage) and corresponding target values (actual housing prices). By training these models on the dataset, we can make predictions on new, unseen data.

It's important to note that the choice between linear and logistic regression depends on the nature of the problem and the type of prediction required. Linear regression is suitable for predicting continuous values, while logistic regression is appropriate for binary classification tasks.

Both linear and logistic regression models are valuable tools for making predictions and understanding relationships within data. They provide insights into patterns, relationships, and trends, which can aid decision-making and problem-solving in various domains. Both linear and logistic regression models are valuable tools for making predictions and understanding relationships within data. They provide insights into patterns, relationships, and trends, which can aid decision-making and problem-solving in various domains.

#Linear Regression

##Find out the predicted values for "bedrooms", for training data "price" Vs "area"

```
#import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

#read the data file from "tips_csv"
data=pd.read_csv("/content/Housing.csv")
data
```

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        ...   ...       ...        ...      ...      ...
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
```

```
0             no          yes      2     yes       furnished
1             no          yes      3      no       furnished
2             no           no      2     yes   semi-furnished
3             no          yes      3     yes       furnished
4             no          yes      2      no       furnished
..           ...          ...    ...     ...           ...
540             no          no      2      no     unfurnished
541             no          no      0      no   semi-furnished
542             no          no      0      no     unfurnished
543             no          no      0      no       furnished
544             no          no      0      no     unfurnished

[545 rows x 13 columns]
```

```
#test the data explicity with inout feature and output variable
X=data[["area","bedrooms"]]#input feature
y=data["price"]#output variable
#divide the dataset int, text parameter
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,
random_state=42)
```

```
#check the model is work or not
model=LinearRegression()
model.fit(X_train,y_train)
```

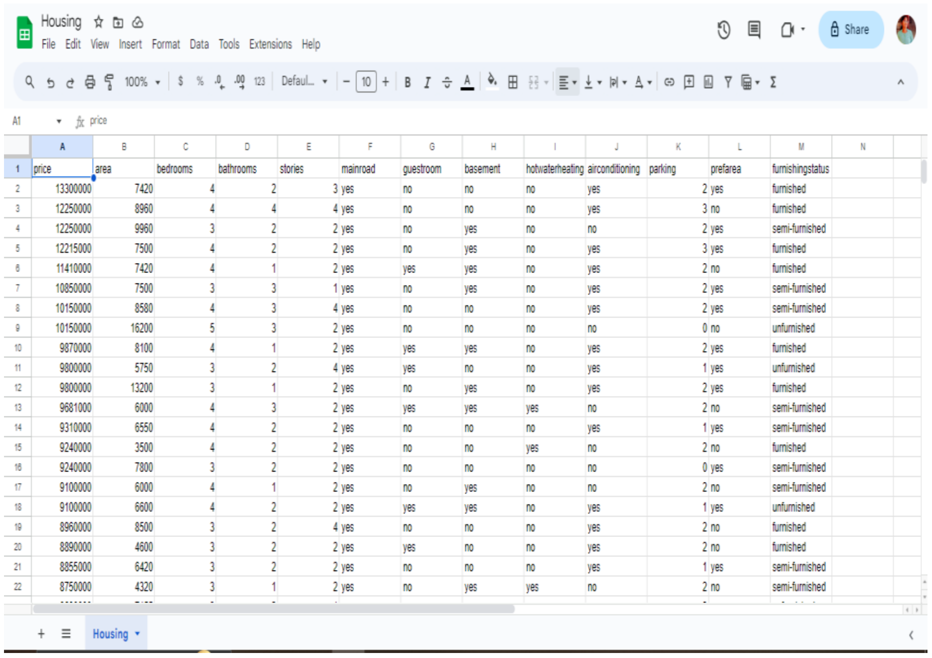
```
LinearRegression()
```

```
#final step predict future steps
y_pred = model.predict(X_test)
```

```
#calculate the mse value
mse= mean_squared_error(y_test,y_pred)
print(f"Mean squared error:{mse}")
```

```
Mean squared error:3280176595474.013
```

```
#visuliaze the prediciton
plt.scatter(y_test,y_pred,color='blue')
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title("Actual Vs Predicted values")
plt.show()
```



price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
10850000	7500	3	3	1	yes	no	yes	no	yes	2	yes	semi-furnished
10150000	8580	4	3	4	yes	no	no	no	yes	2	yes	semi-furnished
10150000	16200	5	3	2	yes	no	no	no	no	0	no	unfurnished
9870000	8100	4	1	2	yes	yes	yes	no	yes	2	yes	furnished
9800000	5750	3	2	4	yes	yes	no	no	yes	1	yes	unfurnished
9800000	13200	3	1	2	yes	no	yes	no	yes	2	yes	furnished
9681000	6000	4	3	2	yes	yes	yes	yes	no	2	no	semi-furnished
9310000	6550	4	2	2	yes	no	no	no	yes	1	yes	semi-furnished
9240000	3500	4	2	2	yes	no	no	yes	no	2	no	furnished
9240000	7800	3	2	2	yes	no	no	no	no	0	yes	semi-furnished
9100000	6000	4	1	2	yes	no	yes	no	no	2	no	semi-furnished
9100000	6600	4	2	2	yes	yes	yes	no	yes	1	yes	unfurnished
8960000	8500	3	2	4	yes	no	no	no	yes	2	no	furnished
8890000	4600	3	2	2	yes	yes	no	no	yes	2	no	furnished
8855000	6420	3	2	2	yes	no	no	no	yes	1	yes	semi-furnished
8750000	4320	3	1	2	yes	no	yes	yes	no	2	no	semi-furnished

Insights:

Scatter plot:

- The scatter plot shows the relationship between the actual values and the predicted values of the housing prices.
- The blue dots represent the data points, where the x-axis represents the actual values and the y-axis represents the predicted values.
- If the predicted values align closely with the actual values, the points will be closer to a diagonal line.
- If the points are scattered far away from the diagonal line, it indicates a larger difference between the actual and predicted values.
- By visually comparing the actual and predicted values, you can assess the performance of the linear regression model in predicting housing prices

#Logistic Regression

##1.Import the libraries which is required for prediction
##2.import the dataset your using workspace
##3.Use the apporipriate argument of sklearn library to train ,test and split the dataset
##4.Fit your values with arange function using FeatureScaling
##5.Check your model accuracy and precision using confusion matrix

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

data = pd.read_csv("/content/Housing.csv")
data
```

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        ...   ...       ...        ...      ...      ...
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              ...             ...      ...      ...              ...
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished

[545 rows x 13 columns]
```

```
X=data[["area"]]
y=data[["price"]]
X_test,X_train,y_test,y_train = train_test_split(X,y,test_size=0.4,random_state=0)

print(X_train)
```

```
      area
239   4000
113   9620
325   3460
66   13200
479   3660
..     ...
68    6000
210   4646
20    4320
504   3185
380   4500

[218 rows x 1 columns]
```

```
print(y_train)
```

```
       price
239  4585000
113  6083000
325  4007500
66   6930000
479  2940000
..       ...
68   6860000
210  4900000
20   8750000
504  2653000
380  3605000

[218 rows x 1 columns]
```

```
print(X_test)
```

```
      area
14    7800
170   5500
330   4050
492   2650
249   4990
..     ...
```

```
70    4000
277  10360
9     5750
359   3600
192   6600

[327 rows x 1 columns]
```

```python
print(y_test)
```

```
       price
14   9240000
170  5250000
330  3990000
492  2800000
249  4543000
..       ...
70   6790000
277  4305000
9    9800000
359  3710000
192  5040000

[327 rows x 1 columns]
```

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```python
print(X_train)
```

```
[[-5.47275221e-01]
 [ 1.82306217e+00]
 [-7.75030059e-01]
 [ 3.33299240e+00]
 [-6.90676415e-01]
 [ 4.43880096e-01]
 [-6.10540454e-01]
 [-7.66594695e-01]
 [-7.53098112e-01]
 [ 1.56156588e+00]
 [ 2.96261219e-01]
 [-5.64145949e-01]
 [-7.79247742e-01]
 [ 3.21567312e-01]
 [ 2.89934696e-01]
 [-1.20945133e+00]
 [-1.67683823e-01]
 [ 3.04696583e-01]
 [-9.26866618e-01]
 [-3.36391111e-01]
 [ 1.20306289e+00]
 [ 4.86056918e-01]
 [-1.25507001e-01]
 [-1.51734213e+00]
 [ 4.73403871e-01]
 [-9.05778207e-01]
 [ 7.18029439e-01]
 [ 1.61295389e-01]
 [-8.42512974e-01]
 [ 4.73403871e-01]
 [-3.78567933e-01]
 [-1.27482540e+00]
 [ 1.30850495e+00]
 [-5.30404492e-01]
 [-9.18431254e-01]
 [ 4.81839236e-01]
 [ 2.96261219e-01]
 [-9.53859785e-01]
 [-7.79247742e-01]
 [-1.31700222e+00]
 [ 1.35068177e+00]
 [ 2.54084397e-01]
 [ 2.11907575e-01]
 [ 8.32682679e-02]
 [-7.49723966e-01]
 [ 2.96261219e-01]
 [-4.79792305e-01]
 [-5.30404492e-01]
 [-5.27873883e-01]
 [ 1.69730753e-01]
```

```
[-1.92989916e-01]
[ 1.84288528e+00]
[-1.53843054e+00]
[-6.10540454e-01]
[ 2.45649033e-01]
[ 1.78166117e-01]
[ 1.24523971e+00]
[-2.77343560e-01]
[-7.15982509e-01]
[ 7.18029439e-01]
[ 3.20646194e+00]
[ 5.49322151e-01]
[-1.33387295e+00]
[ 2.96261219e-01]
[-1.12088000e+00]
[-1.04622702e+00]
[ 7.83403513e-01]
[-3.74350251e-01]
[-3.36391111e-01]
[ 4.22791685e-01]
[-1.32965527e+00]
[ 8.53771090e-02]
[-7.15982509e-01]
[-4.37615483e-01]
[-6.40064229e-01]
[-1.02387331e+00]
[-7.91900788e-01]
[ 1.38442323e+00]
[-1.00700258e+00]
[-5.47275221e-01]
[-1.32965527e+00]
[ 1.00736244e+00]
[-3.32173429e-01]
[ 7.82138208e-01]
[-2.94214289e-01]
[-8.67819068e-01]
[-7.91900788e-01]
[ 2.11907575e-01]
[-9.05778207e-01]
[-5.15642604e-01]
```

```
[ 8.96859164e-01]
[ 1.56156588e+00]
[-6.99111780e-01]
[-9.05778207e-01]
[-1.32965527e+00]
[ 1.54469515e+00]
[-1.32332875e+00]
[ 5.28233740e-01]
[ 1.04279097e+00]
[ 1.30850495e+00]
[ 2.19421821e+00]
[-7.66594695e-01]
[-1.05888007e+00]
[-9.69043440e-01]
[ 4.73403871e-01]
[-4.11533569e-02]
[ 3.21567312e-01]
[-3.36391111e-01]
[ 3.38438041e-01]
[-7.07547144e-01]
[-5.47275221e-01]
[-9.52172712e-01]
[-7.15982509e-01]
[ 1.92428655e+00]
[-5.60771804e-01]
[-6.14758136e-01]
[ 1.89898045e+00]
[-6.99111780e-01]
[-7.49723966e-01]
[ 3.33299240e+00]
[-1.09557391e+00]
[-4.20744755e-01]
[-3.36391111e-01]
[-1.27615842e-01]
[-9.39941433e-01]
[-3.98812807e-01]
[ 1.51095369e+00]
[ 3.22501974e+00]
[ 7.85512354e-01]
[-7.15982509e-01]
```

```
[ 5.70410562e-01]
[-5.26186810e-01]
[-9.69043440e-01]
[-6.99111780e-01]
[ 2.63295583e-02]
[ 4.59829706e+00]
[ 2.79390490e-01]
[-1.39081166e+00]
[ 1.02346508e-03]
[-9.69043440e-01]
[-8.00336153e-01]
[ 8.53771090e-02]
[ 4.64968507e-01]
[ 8.53771090e-02]
[ 2.96261219e-01]
[ 9.18369343e-01]
[-8.93125161e-01]
[ 9.03607455e-01]
[ 5.49322151e-01]
[-9.81696487e-01]
[ 1.30850495e+00]
[-4.03874026e-01]
[-9.34526161e-02]
[ 2.59068033e+00]
[-2.52037467e-01]
[ 2.96261219e-01]
[ 1.38442323e+00]
[ 1.18197448e+00]
[-1.02977806e+00]
[-1.42377730e-01]
[-5.47275221e-01]
[-8.21424564e-01]
[ 2.08455847e+00]
[ 2.96261219e-01]
[-3.36391111e-01]
[-5.51492903e-01]
[-4.12309390e-01]
[-1.43087964e+00]
[-7.53941648e-01]
[-1.22210437e+00]
```

```
[-1.42377730e-01]
[ 5.17689534e-01]
[-6.94894098e-01]
[ 6.12587384e-01]
[ 2.96261219e-01]
[-5.64145949e-01]
[-6.31628865e-01]
[-3.99656344e-01]
[-9.69043440e-01]
[ 9.45882948e-03]
[ 2.32995986e-01]
[ 9.71090371e-01]
[ 2.96261219e-01]
[ 2.09678975e+00]
[ 4.32002871e-02]
[-4.96663034e-01]
[ 1.13136229e+00]
[-6.10540454e-01]
[ 6.50546524e-01]
[-7.03329462e-01]
[-2.94214289e-01]
[ 1.17353912e+00]
[-1.15883914e+00]
[-5.59928267e-01]
[ 2.96261219e-01]
[-5.30404492e-01]
[ 2.96261219e-01]
[ 1.02346508e-03]
[-2.09860645e-01]
[-2.94214289e-01]
[ 8.95172091e-01]
[-1.00700258e+00]
[ 8.02383083e-01]
[-6.99111780e-01]
[-6.52717276e-01]
[ 9.28913549e-01]
[ 5.74628244e-01]
[-5.13533763e-01]
[-3.27955746e-01]
[ 2.96261219e-01]
```

```
 [ 5.55648674e-01]
 [ 4.03390347e-01]
 [ 1.16088607e+00]
 [ 2.96261219e-01]
 [-2.74812951e-01]
 [-4.12309390e-01]
 [-8.91016320e-01]
 [-3.36391111e-01]]
```

```python
print(X_test)
```

```
[[ 1.05544401e+00]
 [ 8.53771090e-02]
 [-5.26186810e-01]
 [-1.11666232e+00]
 [-1.29724683e-01]
 [ 1.18197448e+00]
 [-7.03329462e-01]
 [ 2.47891176e+00]
 [-1.92989916e-01]
 [-7.49723966e-01]
 [ 5.24016058e-01]
 [-5.47275221e-01]
 [-4.96663034e-01]
 [ 6.88505663e-01]
 [-6.35003010e-01]
 [-7.49723966e-01]
 [-1.22210437e+00]
 [-1.45998165e+00]
 [ 3.84832545e-01]
 [-8.17206881e-01]
 [-2.20404850e-01]
 [-7.53941648e-01]
 [-1.46595412e-01]
 [ 1.24523971e+00]
 [ 8.65648316e-01]
 [ 4.33335890e-01]
 [-3.74350251e-01]
 [ 2.27857185e+00]
 [ 6.75852617e-01]
 [-9.31084301e-01]
```

```
[-3.91220979e-01]
[ 5.95716655e-01]
[ 6.42886981e-02]
[ 2.45649033e-01]
[ 4.48097778e-01]
[-9.69043440e-01]
[-1.32965527e+00]
[ 2.57802729e+00]
[-7.01220621e-01]
[-3.95438662e-01]
[-7.58159331e-01]
[-9.18431254e-01]
[ 1.27553931e-01]
[ 8.02383083e-01]
[ 6.42886981e-02]
[-8.67819068e-01]
[ 2.74329272e-01]
[-1.25507001e-01]
[-6.81465230e-02]
[ 2.96261219e-01]
[ 6.33675795e-01]
[-7.75030059e-01]
[-7.49723966e-01]
[ 7.18029439e-01]
[-9.31084301e-01]
[ 1.96646337e+00]
[-7.91900788e-01]
[-3.78567933e-01]
[ 2.19421821e+00]
[-6.48499593e-01]
[-6.82241051e-01]
[ 6.33675795e-01]
[-7.03329462e-01]
[ 1.35911713e+00]
[-4.37615483e-01]
[-7.66594695e-01]
[-1.38448514e+00]
[-9.69043440e-01]
[ 4.32002871e-02]
[-6.10540454e-01]
```

```
[-3.32173429e-01]
[-9.69043440e-01]
[ 2.11907575e-01]
[-1.22294791e+00]
[-9.05778207e-01]
[ 2.47757874e-01]
[-6.54826117e-01]
[-2.09860645e-01]
[ 2.79390490e-01]
[ 8.90954409e-01]
[-1.13353305e+00]
[ 2.96261219e-01]
[ 1.51095369e+00]
[-4.62921577e-01]
[-4.12309390e-01]
[-8.57696630e-01]
[-7.03329462e-01]
[-8.93125161e-01]
[-2.03534122e-01]
[-1.32965527e+00]
[-6.99111780e-01]
[ 5.49322151e-01]
[-1.25373699e+00]
[ 7.18029439e-01]
[ 1.02346508e-03]
[ 8.53771090e-02]
[-5.47275221e-01]
[-1.67683823e-01]
[-8.93125161e-01]
[-2.80717706e-01]
[-1.19047176e+00]
[ 6.59825424e-01]
[ 1.50884485e+00]
[-8.42512974e-01]
[-1.31658045e+00]
[ 6.75852617e-01]
[-1.03863520e+00]
[-5.47275221e-01]
[-1.25507001e-01]
[-7.91900788e-01]
```

```
[ 8.53771090e-02]
[-1.41190007e+00]
[-8.91016320e-01]
[-2.16187168e-01]
[-4.79792305e-01]
[-6.10540454e-01]
[-7.41288602e-01]
[-3.99656344e-01]
[-1.05339708e+00]
[-2.22513691e-01]
[-5.97887407e-01]
[ 4.48097778e-01]
[ 1.01326719e+00]
[ 1.47721223e+00]
[ 3.13131948e-01]
[-5.13955531e-01]
[ 4.31648817e-01]
[ 2.59911570e+00]
[ 1.29669544e+00]
[ 1.05544401e+00]
[-5.75111923e-01]
[-2.94214289e-01]
[ 5.28233740e-01]
[ 4.86056918e-01]
[-2.30949056e-01]
[ 2.96261219e-01]
[-6.99111780e-01]
[-8.93125161e-01]
[ 4.98709964e-01]
[-7.58159331e-01]
[ 8.53771090e-02]
[-5.05098399e-01]
[-9.69043440e-01]
[-7.79247742e-01]
[-7.03329462e-01]
[-5.30404492e-01]
[ 3.38438041e-01]
[ 2.96261219e-01]
[-6.73805687e-01]
[ 5.24016058e-01]
[ 5.49322151e-01]
[ 5.49322151e-01]
[ 4.32002871e-02]
[ 5.49322151e-01]
[ 1.77244999e+00]
[ 8.27689176e-01]
[ 5.55648674e-01]
[-7.79247742e-01]
[ 4.48097778e-01]
[-1.21289319e-01]
[-9.00716989e-01]
[ 2.24560622e-01]
[-6.65370322e-01]
[-1.28881147e-01]
[ 4.32002871e-02]
[ 2.32995986e-01]
[-5.05098399e-01]
[-8.93125161e-01]
[ 1.01326719e+00]
[ 4.73403871e-01]
[-3.36391111e-01]
[-2.94214289e-01]
[-5.53179976e-01]
[-5.89452043e-01]
[-7.22730800e-01]
[-5.38064035e-02]
[-9.69043440e-01]
[ 9.28913549e-01]
[-1.13353305e+00]
[ 4.34523613e+00]
[ 4.48097778e-01]
[-1.05888007e+00]
[ 5.07145329e-01]
[-9.69043440e-01]
[-7.28635555e-01]
[-7.58159331e-01]
[-9.69043440e-01]
[-5.33778638e-01]
[-5.47275221e-01]
[-8.67819068e-01]
```

```
[-7.96118470e-01]
[-1.76119187e-01]
[ 2.96261219e-01]
[-5.13533763e-01]
[-5.30404492e-01]
[ 2.96261219e-01]
[-5.30404492e-01]
[-5.76798996e-01]
[ 5.78845926e-01]
[-1.76119187e-01]
[ 9.21321721e-01]
[-2.09860645e-01]
[ 1.00483183e+00]
[ 4.48097778e-01]
[-3.36391111e-01]
[-1.32965527e+00]
[ 9.03607455e-01]
[ 5.99934337e-01]
[ 1.16088607e+00]
[-9.73261123e-01]
[-5.47275221e-01]
[ 8.52995269e-01]
[ 1.63157940e+00]
[-1.42377730e-01]
[ 3.97485592e-01]
[-7.15982509e-01]
[-1.03230867e+00]
[-2.09860645e-01]
[ 3.38438041e-01]
[-3.11085018e-01]
[-4.12309390e-01]
[ 2.96261219e-01]
[-4.29180119e-01]
[-7.66594695e-01]
[ 8.95172091e-01]
[-3.36391111e-01]
[-5.07207240e-01]
[-9.43737347e-01]
[ 1.44424660e-01]
[-7.15982509e-01]
```

```
[ 5.49322151e-01]
[-4.46050848e-01]
[-2.01425280e-01]
[ 1.24523971e+00]
[-7.58159331e-01]
[-1.07575080e+00]
[ 7.53879737e-01]
[-7.15982509e-01]
[-5.47275221e-01]
[ 2.96261219e-01]
[-1.05339708e+00]
[-5.05098399e-01]
[-1.41190007e+00]
[ 6.44220000e-01]
[ 5.24016058e-01]
[ 2.86482968e+00]
[-7.79247742e-01]
[-4.20744755e-01]
[-8.93125161e-01]
[-9.69043440e-01]
[ 2.54084397e-01]
[-9.69043440e-01]
[ 4.48097778e-01]
[-3.78567933e-01]
[ 2.96261219e-01]
[ 4.48097778e-01]
[-2.09860645e-01]
[-8.46730657e-01]
[-3.36391111e-01]
[-1.17149219e+00]
[ 1.18197448e+00]
[ 2.23717085e-01]
[ 5.07145329e-01]
[ 5.49322151e-01]
[ 1.56156588e+00]
[-6.52717276e-01]
[-3.36391111e-01]
[ 4.32002871e-02]
[-6.48499593e-01]
[-1.09557391e+00]
```

```
[ 8.15457897e-01]
[-6.95315866e-01]
[-8.80472114e-01]
[-5.59928267e-01]
[-9.18431254e-01]
[-3.36391111e-01]
[ 6.12587384e-01]
[ 2.19421821e+00]
[-1.17992755e+00]
[-1.59248458e-01]
[ 7.26464803e-01]
[-7.66594695e-01]
[-9.88866547e-01]
[ 1.56156588e+00]
[-8.91016320e-01]
[ 4.32002871e-02]
[-7.03329462e-01]
[ 1.24523971e+00]
[-4.92445352e-01]
[ 7.18029439e-01]
[-1.10232220e+00]
[ 4.98709964e-01]
[-3.36391111e-01]
[-6.94894098e-01]
[-6.24880573e-01]
[-7.58159331e-01]
[-4.11533569e-02]
[-6.22417679e-02]
[-9.69043440e-01]
[ 8.53771090e-02]
[ 1.11870925e+00]
[ 1.13136229e+00]
[-7.58159331e-01]
[-1.01965563e+00]
[-3.78567933e-01]
[ 9.45882948e-03]
[-1.01965563e+00]
[-1.08636272e-01]
[-6.52717276e-01]
[ 8.53771090e-02]
```

```
 [ 3.17349630e-01]
 [ 2.96261219e-01]
 [-5.07207240e-01]
 [-8.93125161e-01]
 [ 1.13979766e+00]
 [-1.17071636e-01]
 [-6.31628865e-01]
 [-5.64145949e-01]
 [ 2.96261219e-01]
 [-5.30404492e-01]
 [-6.99111780e-01]
 [-7.03329462e-01]
 [-5.47275221e-01]
 [ 2.13517066e+00]
 [ 1.90819164e-01]
 [-7.15982509e-01]
 [ 5.49322151e-01]]
```

```python
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
LogisticRegression(random_state=0)
```

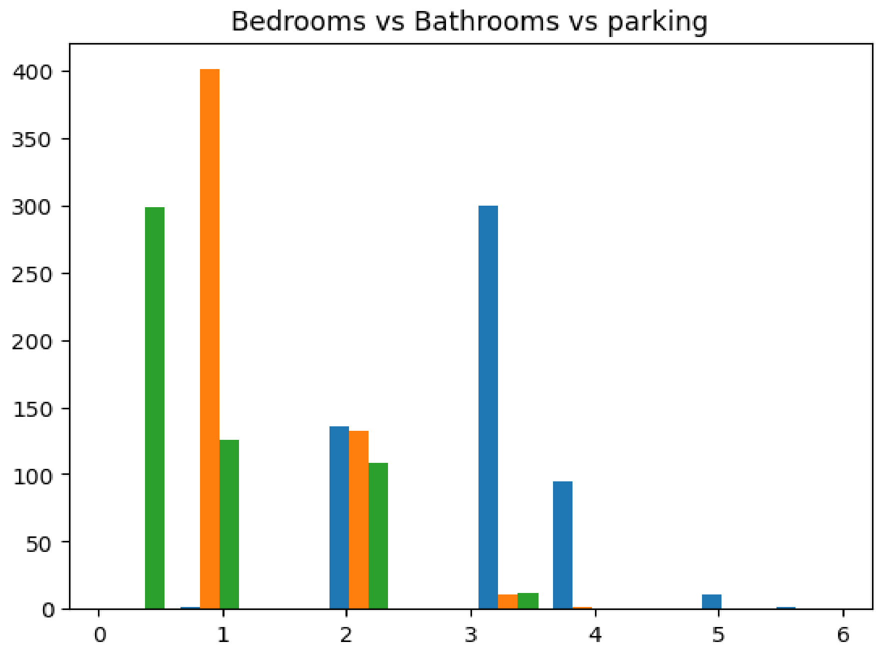
```python
y_pred = classifier.predict(X_test)
```

```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

```
0.021406727828746176
```

```
#visuliaze the prediciton
plt.scatter(y_test,y_pred,color='blue')
plt.xlabel('area')
plt.ylabel('price')
plt.title("area vs price")
plt.show()
```



Insights:

Scatter plot:

- ➢ There is a wide range of prices for a given area, indicating that other factors also influence the price.
- ➢ The predicted prices are not closely aligned with the actual prices, suggesting that the model may not be accurately capturing the relationship between area and price.
- ➢ The scattered distribution of points indicates that the model's predictions are not consistent and may need further improvement.

- ➢ It might be beneficial to explore additional features or consider using a different model to enhance the prediction accuracy.

Recommendations of linear and logistic regression

Based on the insights from both the linear and logistic regression analyses, it seems that the predicted prices in the scatter plot are not closely aligned with the actual prices. This indicates that both models may not accurately capture the relationship between the predictors and the housing prices.

To improve the prediction accuracy, I recommend considering the following:

1. **Explore additional features:**

   Include more relevant variables in the analysis to capture other factors that may influence housing prices, such as location, amenities, or market trends.

2. **Try different models:**

   Consider using alternative regression models, such as decision trees, random forests, or support vector machines, to see if they provide better predictions.

3. **Evaluate data quality:**

   Double-check the cleanliness and completeness of the dataset to ensure accurate analysis and modeling.

By incorporating these recommendations, you can enhance the accuracy of your predictions and gain better insights into the housing market.

Task 4:

Evaluate the accuracy of different machine learning algorithms such as KNN and DT for predicting housing prices.

- ➢ KNN
- ➢ DT

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a popular machine learning algorithm used for classification and regression tasks. It works based on the idea that similar data points are likely to have similar labels or values.

Here are some uses of KNN:

Classification:

KNN can be used to classify data into different categories based on their similarity to labeled data points. For example, it can be used to classify emails as spam or not spam based on their similarity to previously labeled emails.

Regression:

KNN can also be used for regression tasks, where it predicts a continuous value based on the similarity to neighboring data points. For instance, it can predict the price of a house based on the similarity to nearby houses with known prices.

To use KNN, you typically follow these steps:

1. **Prepare the data:**

   Clean and preprocess the data, ensuring it is in a suitable format for KNN.

2. **Choose the value of K:**

   Determine the number of nearest neighbors to consider when making predictions. This value affects the model's performance.

3. **Calculate distances:**

   Measure the distance between the new data point and all other data points in the dataset. Common distance metrics include Euclidean distance and Manhattan distance.

4. **Select K nearest neighbors:**

   Identify the K data points with the shortest distances to the new data point.

5. **Make predictions:**

For classification, assign the most common class label among the K neighbors to the new data point. For regression, compute the average or weighted average of the K neighbors' values.

KNN is a simple yet powerful algorithm that can be used in various domains, such as healthcare, finance, and recommendation systems. It's important to choose the appropriate value of K and select relevant features to achieve accurate predictions.

#Knearest neighbors:

```
#import the required libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

#load the dataset
data = pd.read_csv("/content/Housing.csv")
data
```

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        ...   ...       ...        ...      ...      ...       ...      ...
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              ...             ...      ...      ...              ...
```

```
540         no        no    2    no     unfurnished
541         no        no    0    no   semi-furnished
542         no        no    0    no     unfurnished
543         no        no    0    no       furnished
544         no        no    0    no     unfurnished

[545 rows x 13 columns]
```

```
X=data.iloc[:,1:2].values
y=data.iloc[:,0:1].values
print(X)
print(y)
```

```
[[ 7420]
 [ 8960]
 [ 9960]
 [ 7500]
 [ 7420]
 [ 7500]
 [ 8580]
 [16200]
 [ 8100]
 [ 5750]
 [13200]
 [ 6000]
 [ 6550]
 [ 3500]
 [ 7800]
 [ 6000]
 [ 6600]
 [ 8500]
 [ 4600]
 [ 6420]
 [ 4320]
 [ 7155]
 [ 8050]
 [ 4560]
 [ 8800]
 [ 6540]
 [ 6000]
 [ 8875]
 [ 7950]
 [ 5500]
 [ 7475]
 [ 7000]
 [ 4880]
 [ 5960]
 [ 6840]
 [ 7000]
 [ 7482]
 [ 9000]
 [ 6000]
 [ 6000]
 [ 6550]
 [ 6360]
 [ 6480]
 [ 6000]
 [ 6000]
 [ 6000]
 [ 6000]
 [ 6600]
 [ 4300]
 [ 7440]
 [ 7440]
 [ 6325]
 [ 6000]
 [ 5150]
 [ 6000]
 [ 6000]
 [11440]
 [ 9000]
 [ 7680]
 [ 6000]
 [ 6000]
 [ 8880]
 [ 6240]
 [ 6360]
 [11175]
 [ 8880]
 [13200]
 [ 7700]
 [ 6000]
 [12090]
 [ 4000]
 [ 6000]
 [ 5020]
 [ 6600]
 [ 4040]
 [ 4260]
 [ 6420]
 [ 6500]
 [ 5700]
 [ 6000]
 [ 6000]
 [ 4000]
 [10500]
 [ 6000]
 [ 3760]
 [ 8250]
 [ 6670]
 [ 3960]
 [ 7410]
 [ 8580]
 [ 5000]
 [ 6750]
 [ 4800]
 [ 7200]
 [ 6000]
 [ 4100]
 [ 9000]
 [ 6400]
 [ 6600]
 [ 6000]
 [ 6600]
 [ 5500]
 [ 5500]
 [ 6350]
 [ 5500]
 [ 4500]
 [ 5450]
 [ 6420]
 [ 3240]
 [ 6615]
 [ 6600]
 [ 8372]
 [ 4300]
 [ 9620]
 [ 6800]
 [ 8000]
 [ 6900]
 [ 3700]
 [ 6420]
 [ 7020]
 [ 6540]
 [ 7231]
 [ 6254]
 [ 7320]
 [ 6525]
 [15600]
 [ 7160]
 [ 6500]
 [ 5500]
 [11460]
 [ 4800]
 [ 5828]
 [ 5200]
 [ 4800]
 [ 7000]
 [ 6000]
 [ 5400]
 [ 4640]
 [ 5000]
 [ 6360]
 [ 5800]
 [ 6660]
 [10500]
 [ 4800]
 [ 4700]
 [ 5000]
 [10500]
 [ 5500]
 [ 6360]
 [ 6600]
 [ 5136]
 [ 4400]
 [ 5400]
 [ 3300]
 [ 3650]
 [ 6100]
 [ 6900]
 [ 2817]
 [ 7980]
 [ 3150]
 [ 6210]
 [ 6100]
 [ 6600]
 [ 6825]
 [ 6710]
 [ 6450]
 [ 7800]
 [ 4600]
 [ 4260]
 [ 6540]
 [ 5500]
 [10269]
 [ 8400]
 [ 5300]
 [ 3800]
 [ 9800]
 [ 8520]
 [ 6050]
 [ 7085]
 [ 3180]
 [ 4500]
 [ 7200]
 [ 3410]
 [ 7980]
 [ 3000]
 [ 3000]
 [11410]
 [ 6100]
 [ 5720]
 [ 3540]
 [ 7600]
 [10700]
 [ 6600]
 [ 4800]
 [ 8150]
 [ 4410]
 [ 7686]
 [ 2800]
 [ 5948]
 [ 4200]
 [ 4520]
 [ 4095]
 [ 4120]
 [ 5400]
 [ 4770]
 [ 6300]
 [ 5800]
 [ 3000]
 [ 2970]
 [ 6720]
 [ 4646]
 [12900]
 [ 3420]
 [ 4995]
 [ 4350]
 [ 4160]
 [ 6040]
 [ 6862]
 [ 4815]
 [ 7000]
 [ 8100]
 [ 3420]
 [ 9166]
 [ 6321]
 [10240]
 [ 6440]
 [ 5170]
 [ 6000]
```

[3630]
[9667]
[5400]
[4320]
[3745]
[4160]
[3880]
[5680]
[2870]
[5010]
[4510]
[4000]
[3840]
[3760]
[3640]
[2550]
[5320]
[5360]
[3520]
[8400]
[4100]
[4990]
[3510]
[3450]
[9860]
[3520]
[4510]
[5885]
[4000]
[8250]
[4040]
[6360]
[3162]
[3510]
[3750]
[3968]
[4900]
[2880]
[4880]
[4920]

```
[ 4950]
[ 3900]
[ 4500]
[ 1905]
[ 4075]
[ 3500]
[ 6450]
[ 4032]
[ 4400]
[10360]
[ 3400]
[ 6360]
[ 6360]
[ 4500]
[ 2175]
[ 4360]
[ 7770]
[ 6650]
[ 2787]
[ 5500]
[ 5040]
[ 5850]
[ 2610]
[ 2953]
[ 2747]
[ 4410]
[ 4000]
[ 2325]
[ 4600]
[ 3640]
[ 5800]
[ 7000]
[ 4079]
[ 3520]
[ 2145]
[ 4500]
[ 8250]
[ 3450]
[ 4840]
[ 4080]
```

[4046]
[4632]
[5985]
[6060]
[3600]
[3680]
[4040]
[5600]
[5900]
[4992]
[4340]
[3000]
[4320]
[3630]
[3460]
[5400]
[4500]
[3460]
[4100]
[6480]
[4500]
[3960]
[4050]
[7260]
[5500]
[3000]
[3290]
[3816]
[8080]
[2145]
[3780]
[3180]
[5300]
[3180]
[7152]
[4080]
[3850]
[2015]
[2176]
[3350]

[3150]
[4820]
[3420]
[3600]
[5830]
[2856]
[8400]
[8250]
[2520]
[6930]
[3480]
[3600]
[4040]
[6020]
[4050]
[3584]
[3120]
[5450]
[3630]
[3630]
[5640]
[3600]
[4280]
[3570]
[3180]
[3000]
[3520]
[5960]
[4130]
[2850]
[2275]
[3520]
[4500]
[4000]
[3150]
[4500]
[4500]
[3640]
[3850]
[4240]

[3650]
[4600]
[2135]
[3036]
[3990]
[7424]
[3480]
[3600]
[3640]
[5900]
[3120]
[7350]
[3512]
[9500]
[5880]
[12944]
[4900]
[3060]
[5320]
[2145]
[4000]
[3185]
[3850]
[2145]
[2610]
[1950]
[4040]
[4785]
[3450]
[3640]
[3500]
[4960]
[4120]
[4750]
[3720]
[3750]
[3100]
[3185]
[2700]
[2145]

[4040]
[4775]
[2500]
[3180]
[6060]
[3480]
[3792]
[4040]
[2145]
[5880]
[4500]
[3930]
[3640]
[4370]
[2684]
[4320]
[3120]
[3450]
[3986]
[3500]
[4095]
[1650]
[3450]
[6750]
[9000]
[3069]
[4500]
[5495]
[2398]
[3000]
[3850]
[3500]
[8100]
[4960]
[2160]
[3090]
[4500]
[3800]
[3090]
[3240]

[2835]
[4600]
[5076]
[3750]
[3630]
[8050]
[4352]
[3000]
[5850]
[4960]
[3600]
[3660]
[3480]
[2700]
[3150]
[6615]
[3040]
[3630]
[6000]
[5400]
[5200]
[3300]
[4350]
[2640]
[2650]
[3960]
[6800]
[4000]
[4000]
[3934]
[2000]
[3630]
[2800]
[2430]
[3480]
[4000]
[3185]
[4000]
[2910]
[3600]

```
 [ 4400]
 [ 3600]
 [ 2880]
 [ 3180]
 [ 3000]
 [ 4400]
 [ 3000]
 [ 3210]
 [ 3240]
 [ 3000]
 [ 3500]
 [ 4840]
 [ 7700]
 [ 3635]
 [ 2475]
 [ 2787]
 [ 3264]
 [ 3640]
 [ 3180]
 [ 1836]
 [ 3970]
 [ 3970]
 [ 1950]
 [ 5300]
 [ 3000]
 [ 2400]
 [ 3000]
 [ 3360]
 [ 3420]
 [ 1700]
 [ 3649]
 [ 2990]
 [ 3000]
 [ 2400]
 [ 3620]
 [ 2910]
 [ 3850]]
[[13300000]
 [12250000]
 [12250000]
```

```
[12215000]
[11410000]
[10850000]
[10150000]
[10150000]
[ 9870000]
[ 9800000]
[ 9800000]
[ 9681000]
[ 9310000]
[ 9240000]
[ 9240000]
[ 9100000]
[ 9100000]
[ 8960000]
[ 8890000]
[ 8855000]
[ 8750000]
[ 8680000]
[ 8645000]
[ 8645000]
[ 8575000]
[ 8540000]
[ 8463000]
[ 8400000]
[ 8400000]
[ 8400000]
[ 8400000]
[ 8400000]
[ 8295000]
[ 8190000]
[ 8120000]
[ 8080940]
[ 8043000]
[ 7980000]
[ 7962500]
[ 7910000]
[ 7875000]
[ 7840000]
[ 7700000]
[ 7700000]
[ 7560000]
[ 7560000]
[ 7525000]
[ 7490000]
[ 7455000]
[ 7420000]
[ 7420000]
[ 7420000]
[ 7350000]
[ 7350000]
[ 7350000]
[ 7350000]
[ 7343000]
[ 7245000]
[ 7210000]
[ 7210000]
[ 7140000]
[ 7070000]
[ 7070000]
[ 7035000]
[ 7000000]
[ 6930000]
[ 6930000]
[ 6895000]
[ 6860000]
[ 6790000]
[ 6790000]
[ 6755000]
[ 6720000]
[ 6685000]
[ 6650000]
[ 6650000]
[ 6650000]
[ 6650000]
[ 6650000]
[ 6650000]
[ 6629000]
[ 6615000]
[ 6615000]
[ 6580000]
[ 6510000]
[ 6510000]
[ 6510000]
[ 6475000]
[ 6475000]
[ 6440000]
[ 6440000]
[ 6419000]
[ 6405000]
[ 6300000]
[ 6300000]
[ 6300000]
[ 6300000]
[ 6300000]
[ 6293000]
[ 6265000]
[ 6230000]
[ 6230000]
[ 6195000]
[ 6195000]
[ 6195000]
[ 6160000]
[ 6160000]
[ 6125000]
[ 6107500]
[ 6090000]
[ 6090000]
[ 6090000]
[ 6083000]
[ 6083000]
[ 6020000]
[ 6020000]
[ 6020000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5950000]
[ 5943000]
[ 5880000]
[ 5880000]
[ 5873000]
[ 5873000]
[ 5866000]
[ 5810000]
[ 5810000]
[ 5810000]
[ 5803000]
[ 5775000]
[ 5740000]
[ 5740000]
[ 5740000]
[ 5740000]
[ 5740000]
[ 5652500]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5600000]
[ 5565000]
[ 5565000]
[ 5530000]
[ 5530000]
[ 5530000]
[ 5523000]
[ 5495000]
[ 5495000]
[ 5460000]
[ 5460000]
[ 5460000]
[ 5460000]
[ 5425000]
[ 5390000]
[ 5383000]
[ 5320000]
[ 5285000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5250000]
[ 5243000]
[ 5229000]
[ 5215000]
[ 5215000]
[ 5215000]
[ 5145000]
[ 5145000]
[ 5110000]
[ 5110000]
[ 5110000]
[ 5110000]
[ 5075000]
[ 5040000]
[ 5040000]
[ 5040000]
[ 5040000]
[ 5033000]
[ 5005000]
[ 4970000]
[ 4970000]
[ 4956000]
[ 4935000]
[ 4907000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4900000]
[ 4893000]
[ 4893000]
[ 4865000]
[ 4830000]
[ 4830000]
[ 4830000]
[ 4830000]
[ 4795000]
[ 4795000]
[ 4767000]
[ 4760000]
[ 4760000]
[ 4760000]
[ 4753000]
[ 4690000]
[ 4690000]
[ 4690000]
[ 4690000]
[ 4690000]
[ 4690000]
[ 4655000]
[ 4620000]
[ 4620000]
[ 4620000]
[ 4620000]
[ 4620000]
[ 4613000]
[ 4585000]
[ 4585000]
[ 4550000]
[ 4550000]
[ 4550000]
[ 4550000]
[ 4550000]
[ 4550000]
[ 4550000]
[ 4543000]
[ 4543000]
[ 4515000]
[ 4515000]
[ 4515000]
[ 4515000]
[ 4480000]
[ 4480000]
[ 4480000]
[ 4480000]
[ 4480000]
[ 4473000]
[ 4473000]
[ 4473000]
[ 4445000]
[ 4410000]
[ 4410000]
[ 4403000]
[ 4403000]
[ 4403000]
[ 4382000]
[ 4375000]
[ 4340000]
[ 4340000]
[ 4340000]
[ 4340000]
[ 4340000]
[ 4319000]
[ 4305000]
[ 4305000]
[ 4277000]
[ 4270000]
[ 4270000]
[ 4270000]
[ 4270000]
[ 4270000]
[ 4270000]
[ 4235000]
[ 4235000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4200000]
[ 4193000]
[ 4193000]
[ 4165000]
[ 4165000]
[ 4165000]
[ 4130000]
[ 4130000]
[ 4123000]
[ 4098500]
[ 4095000]
[ 4095000]
[ 4095000]
[ 4060000]
[ 4060000]
[ 4060000]
[ 4060000]
[ 4060000]
[ 4025000]
[ 4025000]
[ 4025000]
[ 4007500]
[ 4007500]
[ 3990000]
[ 3990000]
[ 3990000]
[ 3990000]
[ 3990000]
[ 3920000]
[ 3920000]
[ 3920000]
[ 3920000]
[ 3920000]
[ 3920000]
[ 3920000]
[ 3885000]
[ 3885000]
[ 3850000]
[ 3850000]
[ 3850000]
[ 3850000]
[ 3850000]
[ 3850000]
[ 3850000]
[ 3836000]
[ 3815000]
[ 3780000]
[ 3780000]
[ 3780000]
[ 3780000]
[ 3780000]
[ 3780000]
[ 3773000]
[ 3773000]
[ 3773000]
[ 3745000]
[ 3710000]
[ 3710000]
[ 3710000]
[ 3710000]
[ 3710000]
[ 3703000]
[ 3703000]
[ 3675000]
[ 3675000]
[ 3675000]
[ 3675000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3640000]
[ 3633000]
[ 3605000]
[ 3605000]
[ 3570000]
[ 3570000]
[ 3570000]
[ 3570000]
[ 3535000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3500000]
[ 3493000]
[ 3465000]
[ 3465000]
[ 3465000]
[ 3430000]
[ 3430000]
[ 3430000]
[ 3430000]
[ 3430000]
[ 3430000]
[ 3423000]
[ 3395000]
[ 3395000]
[ 3395000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3360000]
[ 3353000]
[ 3332000]
[ 3325000]
[ 3325000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3290000]
[ 3255000]
[ 3255000]
[ 3234000]
[ 3220000]
[ 3220000]
[ 3220000]
[ 3220000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3150000]
[ 3143000]
[ 3129000]
[ 3118850]
[ 3115000]
[ 3115000]
[ 3115000]
[ 3087000]
[ 3080000]
[ 3080000]
[ 3080000]
[ 3080000]
[ 3045000]
[ 3010000]
[ 3010000]
[ 3010000]
[ 3010000]
[ 3010000]
[ 3010000]
[ 3010000]
[ 3003000]
[ 2975000]
[ 2961000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2940000]
[ 2870000]
[ 2870000]
[ 2870000]
[ 2870000]
[ 2852500]
[ 2835000]
[ 2835000]
[ 2835000]
[ 2800000]
[ 2800000]
[ 2730000]
[ 2730000]
[ 2695000]
[ 2660000]
[ 2660000]
[ 2660000]
[ 2660000]
[ 2660000]
[ 2660000]
[ 2660000]
[ 2653000]
[ 2653000]
[ 2604000]
[ 2590000]
[ 2590000]
[ 2590000]
[ 2520000]
[ 2520000]
[ 2520000]
[ 2485000]
[ 2485000]
[ 2450000]
[ 2450000]
[ 2450000]
[ 2450000]
[ 2450000]
[ 2450000]
[ 2408000]
[ 2380000]
```

```
 [ 2380000]
 [ 2380000]
 [ 2345000]
 [ 2310000]
 [ 2275000]
 [ 2275000]
 [ 2275000]
 [ 2240000]
 [ 2233000]
 [ 2135000]
 [ 2100000]
 [ 2100000]
 [ 2100000]
 [ 1960000]
 [ 1890000]
 [ 1890000]
 [ 1855000]
 [ 1820000]
 [ 1767150]
 [ 1750000]
 [ 1750000]
 [ 1750000]]
```

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

# Create a kNN classifier with k=3
k = 3
knn_classifier = KNeighborsClassifier(n_neighbors=k)

# Train the classifier on the training data
knn_classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)

KNeighborsClassifier(n_neighbors=3)
```

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

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.02
```

Decision Tree (DT)

Decision Trees (DT) is a popular machine learning algorithm used for both classification and regression tasks. It works by creating a tree-like model of decisions and their possible consequences.

Here are some uses of Decision Trees:

- **Classification:**

  Decision Trees can be used to classify data into different categories based on a set of features. For example, it can be used to classify whether an email is spam or not based on various attributes of the email.

- **Regression:**

  Decision Trees can also be used for regression tasks, where it predicts a continuous value based on a set of input features. For instance, it can predict the price of a house based on features such as the number of bedrooms, bathrooms, and square footage.

To use Decision Trees, you typically follow these steps

1. **Prepare the data:**

Clean and preprocess the data, ensuring it is in a suitable format for Decision Trees.

2. Select features:

Choose the relevant features that will be used to make decisions and predictions.

3. Build the tree:

The algorithm recursively splits the data based on the selected features, creating a tree-like structure of decisions and outcomes.

4. Make predictions:

Once the tree is built, new data points can be classified or predicted by traversing the tree based on their feature values.

Decision Trees are versatile and can handle both numerical and categorical data. They are also interpretable, allowing you to understand the decision-making process. However, they can be prone to overfitting if not properly tuned or if the dataset is imbalanced.

When using Decision Trees, it's important to consider the trade-off between model complexity and accuracy. Additionally, techniques such as pruning and ensembling can be used to improve the performance of Decision Trees.

#Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

# Create a Decision Tree classifier
decision_tree = DecisionTreeClassifier()

# Train the classifier on the training data
decision_tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

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

```
array([ 3500000,  5880000,  3290000,  5600000,  3990000,  5390000,
        8960000,  4060000,  6107500,  2940000,  8960000,  3255000,
        4655000,  3010000,  2275000,  2870000,  3850000,  7560000,
        4200000,  7560000,  4620000,  8120000,  3500000,  3290000,
        3773000,  4515000,  3010000,  2100000,  6930000,  2100000,
        2653000,  3220000,  7560000,  6090000,  4900000,  3010000,
        4130000,  2310000,  2660000,  4235000,  5145000,  2870000,
        5950000,  4620000,  6419000,  3290000,  7560000,  4130000,
        3920000,  2100000,  8043000,  2100000,  5320000,  3570000,
        2345000,  2604000,  5075000,  1767150,  5383000,  3290000,
        2310000,  3290000,  5383000,  2940000,  3465000,  4550000,
        5040000,  2100000,  5250000,  4620000,  3780000,  4270000,
        5250000,  5600000,  2940000,  5250000,  2870000,  3780000,
        7560000,  3850000,  4270000,  3850000,  3920000,  5652500,
        3150000,  3640000,  4956000,  3150000, 11410000,  7560000,
        2240000,  3920000,  6160000,  4277000,  8960000,  4095000,
        3780000,  4200000,  3920000,  5040000,  6107500,  3010000,
        4200000,  2835000,  7560000,  5523000,  7560000,  7560000,
        5110000])
```

```python
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.01
```

Recommendations for KNN and DT:

K-Nearest Neighbors (KNN) algorithm, here are some recommendations:

- **Choose an appropriate value for K:**

  The value of K determines the number of neighbors considered for classification or regression. Experiment with different values of K to find the optimal balance between bias and variance.

- **Normalize the data:**

  Since KNN relies on distance calculations, it's important to normalize the features to ensure that no single feature dominates the distance calculation.

- **Handle missing values:**

  KNN does not handle missing values well. Consider imputing missing values or using techniques like mean imputation or KNN imputation before applying the algorithm.

Decision Trees (DT), here are some recommendations:

- **Prune the tree**:

  Decision Trees can easily overfit the training data. Pruning techniques like cost complexity pruning (also known as "weakest link pruning") can help prevent overfitting and improve generalization.

- **Handle categorical variables:**

Decision Trees typically work well with numerical features. For categorical variables, consider using techniques like one-hot encoding or ordinal encoding to convert them into numerical representations.

- **Feature selection**: If you have a large number of features, consider performing feature selection or dimensionality reduction techniques to improve the interpretability and performance of the Decision Tree.

Remember, the choice of algorithm depends on the specific problem and dataset. It's always a good idea to experiment, tune hyper parameters, and evaluate the performance of different algorithms to find the best solution for your particular task.

Task-5

Create a confusion matrix to validate the final output of the machine learning models.

Confusion matrix:

A confusion matrix is a valuable tool for validating the final output of machine learning models. It helps in evaluating the performance of classification models by comparing the predicted labels with the actual labels of the test data.

The confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives. These values allow you to calculate various performance metrics like accuracy, precision, recall, and F1 score. By analyzing the confusion matrix, you can gain insights into how well your model is performing and identify any areas of improvement.

The confusion matrix is particularly useful in scenarios where the class distribution is imbalanced or when you want to understand the specific types of errors made by the model. It helps you identify if the model is misclassifying certain classes more than others and provides a clear picture of its strengths and weaknesses.

Overall, the confusion matrix is an essential tool for evaluating the performance of machine learning models and making informed decisions about model improvements or adjustments. It helps you understand the effectiveness of your model in differentiating between classes and provides valuable insights for model evaluation and refinement.

```
from scipy.stats import chi2_contingency
 b=chi2_contingency(y_pred)
 b
```

```
Chi2ContingencyResult(statistic=0.0, pvalue=1.0, dof=0, expected_freq=array([
3500000.,  5880000.,  3290000.,  5600000.,  3990000.,  5390000.,
        8960000.,  4060000.,  6107500.,  2940000.,  8960000.,  3255000.,
        4655000.,  3010000.,  2275000.,  2870000.,  3850000.,  7560000.,
        4200000.,  7560000.,  4620000.,  8120000.,  3500000.,  3290000.,
        3773000.,  4515000.,  3010000.,  2100000.,  6930000.,  2100000.,
        2653000.,  3220000.,  7560000.,  6090000.,  4900000.,  3010000.,
        4130000.,  2310000.,  2660000.,  4235000.,  5145000.,  2870000.,
        5950000.,  4620000.,  6419000.,  3290000.,  7560000.,  4130000.,
        3920000.,  2100000.,  8043000.,  2100000.,  5320000.,  3570000.,
        2345000.,  2604000.,  5075000.,  1767150.,  5383000.,  3290000.,
        2310000.,  3290000.,  5383000.,  2940000.,  3465000.,  4550000.,
        5040000.,  2100000.,  5250000.,  4620000.,  3780000.,  4270000.,
        5250000.,  5600000.,  2940000.,  5250000.,  2870000.,  3780000.,
        7560000.,  3850000.,  4270000.,  3850000.,  3920000.,  5652500.,
        3150000.,  3640000.,  4956000.,  3150000., 11410000.,  7560000.,
        2240000.,  3920000.,  6160000.,  4277000.,  8960000.,  4095000.,
        3780000.,  4200000.,  3920000.,  5040000.,  6107500.,  3010000.,
        4200000.,  2835000.,  7560000.,  5523000.,  7560000.,  7560000.,
        5110000.]))
```

```
#chi_2_contingous test
print("statistic value is")
print(b.statistic)
pvalue=b.pvalue
print("p value is",pvalue)
if pvalue<0.05:
  print("it is Null Hypoyhises")
else:
  print("it is rejected and not null hypothises")
```

```
statistic value is
0.0
```

```
p value is 1.0
it is rejected and not null hypothises
```

```
#ttest chisquare
from scipy.stats import ttest_ind
t_stat,pvalue=ttest_ind(y_test,y_pred)
print("f_statistic value is",t_stat)
print("pvalue is",pvalue)
if pvalue<0.05:
  print("it is Null Hypothises")
else:
  print("it is rejected and not null hypothesis")
```

```
f_statistic value is [1.56882874]
pvalue is [0.1181516]
it is rejected and not null hypothesis
```

```
#f_oneway
#ttest chisquare
from scipy.stats import f_oneway
f_stat,pvalue=ttest_ind(y_test,y_pred)
print("f_statistic value is",f_stat)
print("pvalue is",pvalue)
if pvalue<0.05:
  print("it is Null Hypothises")
else:
  print("it is rejected and not null hypothesis")
```

```
f_statistic value is [1.56882874]
pvalue is [0.1181516]
it is rejected and not null hypothesis
```

Confusion matrix and validate your final output

```
from sklearn import metrics
cm=metrics.confusion_matrix(y_test,y_pred)
cm
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 1, 0, 0]])
```

Recommendations for confusion matrix

Based on the confusion matrix, here are some recommendations:

1. Analyze misclassified instances to identify patterns.

2. Adjust the decision threshold to prioritize precision or recall.

3. Collect more data for classes with consistent misclassifications.

4. Consider feature engineering to improve model performance.

Conclusion:

After conducting exploratory data analysis (EDA) on the housing price detection dataset and utilizing the seaborn and matplotlib libraries for visualization, we gained valuable insights into the data. We applied linear and logistic regression to build machine learning models for predicting housing prices. We also evaluated the accuracy of different machine learning algorithms, including KNN and DT, for predicting housing prices.

To validate the final output of the machine learning models, we created a confusion matrix, which provided a detailed breakdown of true positives, true negatives, false positives, and false negatives. This allowed us to assess the performance of the models and make informed decisions for improvement.

Based on the analyzed data, we generated visualizations and insights that helped in making recommendations. These insights included the relationship between the number of bedrooms and the price of properties, the impact of different levels of furnishing on property prices, and more.

In conclusion, by conducting EDA, utilizing visualization libraries, building machine learning models, evaluating accuracy, creating a confusion matrix, and generating insights, we gained a comprehensive understanding of the housing price dataset and made informed recommendations for further analysis and decision-making.

THANK YOU