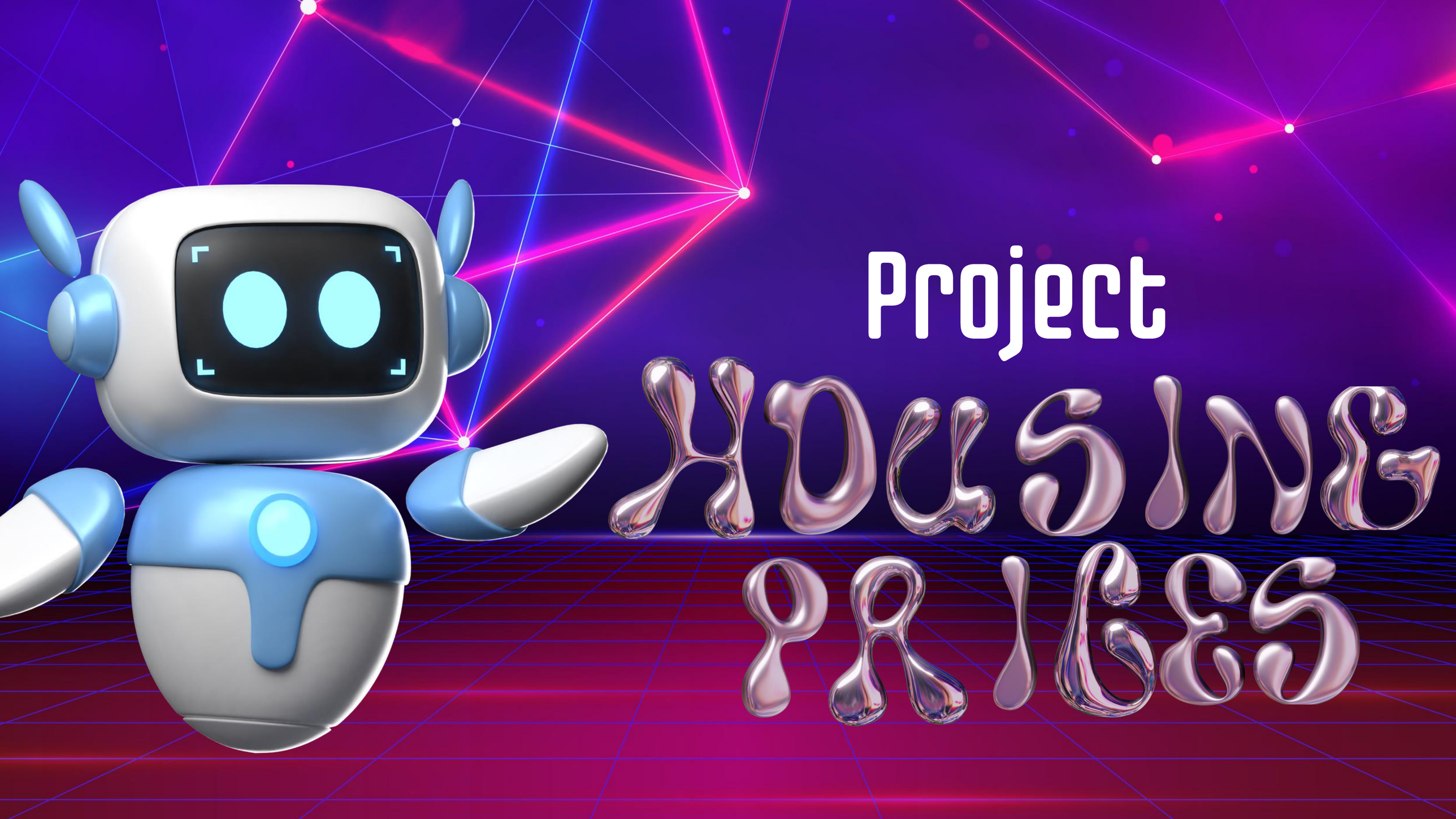
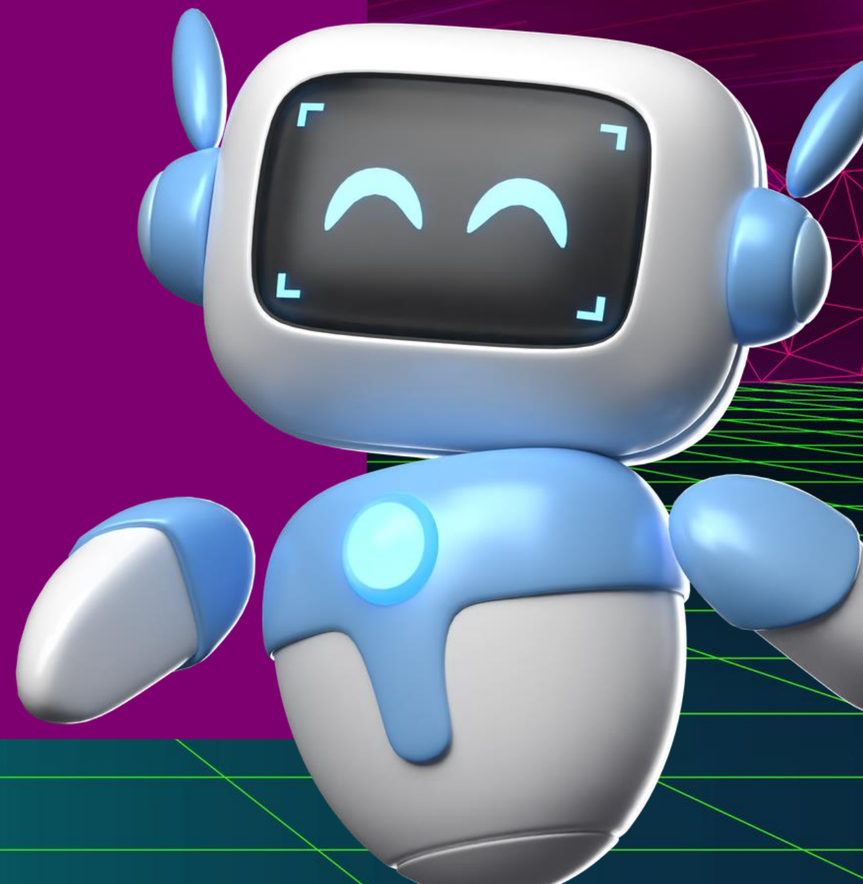

Project
HOUSING
PRICES

What are we going to learn?

- Define Problem
- Data Collection
- Data Representation
- Data Wrangling
- Data Preprocessing
- Data Analysis
- Data Modeling



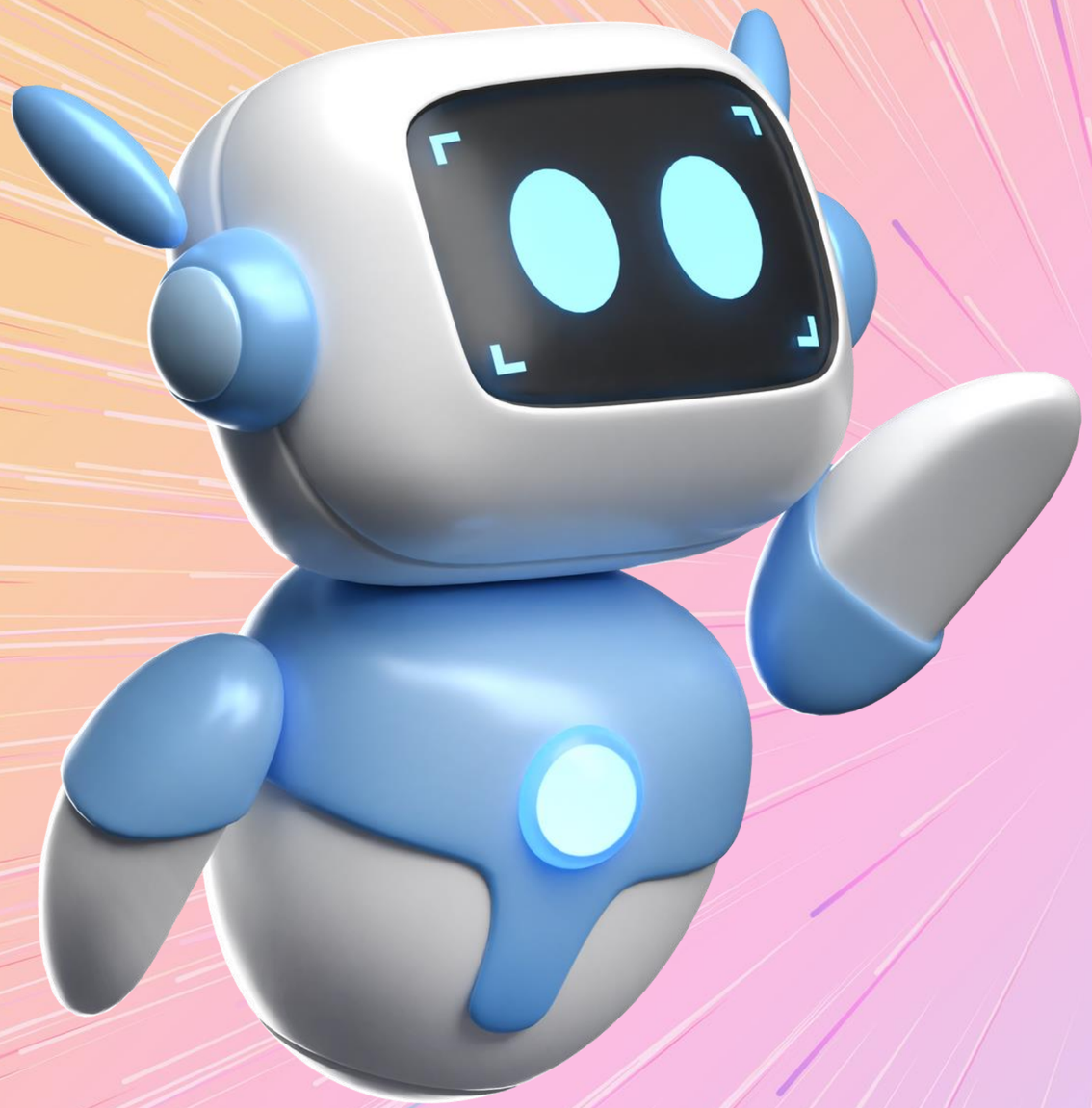
What is the problem?

- We want to build a machine learning model that can predict the price of a house based on its features — such as area, number of bedrooms, bathrooms, location, etc.

- This is a supervised learning problem because we have input data (features) and an output (house price).

Goal:

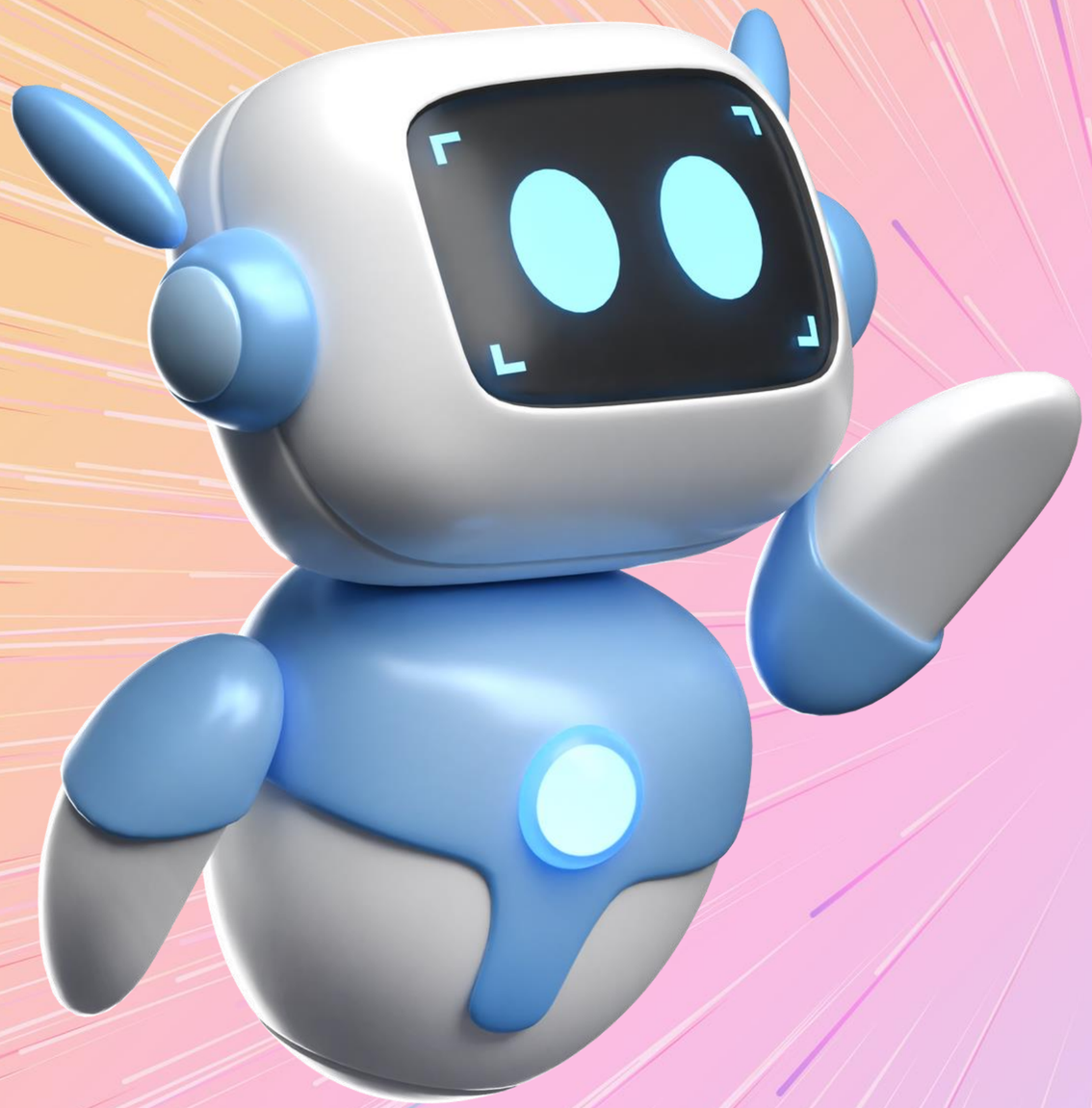
To estimate the house price using a regression model.



Data Collection

Main sources:

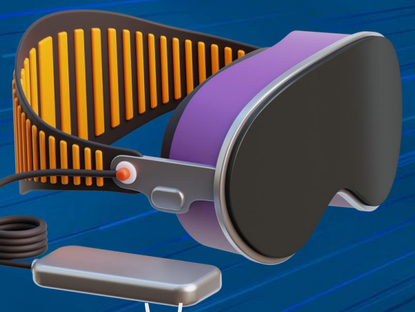
- **Ready datasets** (CSV, Excel, SQL)
- **Public platforms:**
- **Kaggle** (general), Roboflow (images), HuggingFace (NLP)
- **APIs:** Get live data (e.g., Twitter API, Weather API)
- **Web scraping:** Extract data from websites
- **Manual entry:** If data isn't available online



Project use case:

- Dataset was loaded from Kaggle using:

1. python
2. CopyEdit
3. df = pd.read_csv("Housing.csv")



Data Representation

We convert raw data into a structured, numeric format that machine learning models can understand.

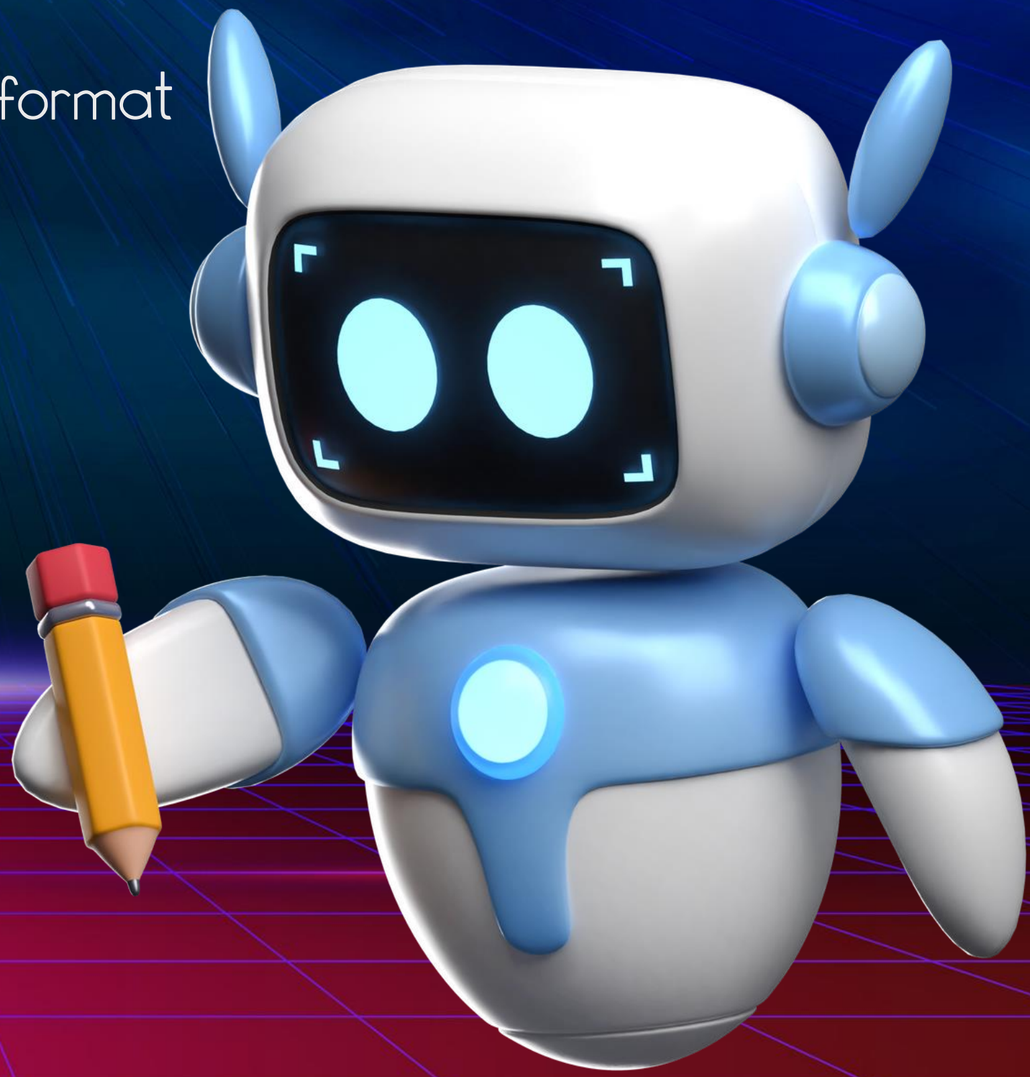
In our project:

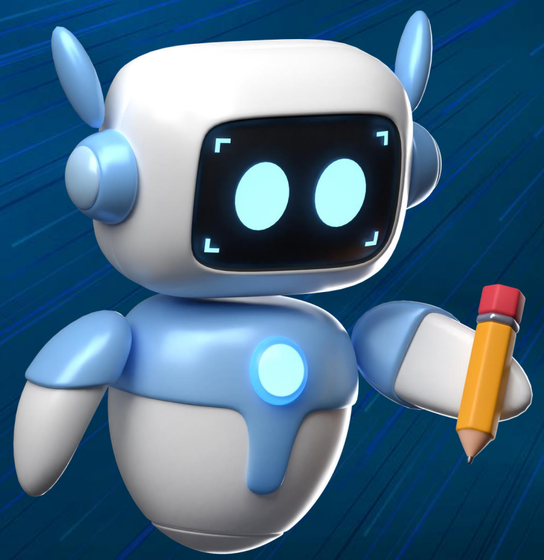
Data is stored in a DataFrame

(rows = houses, columns = features)

Features include:

area, bedrooms, bathrooms, furnishing status, price



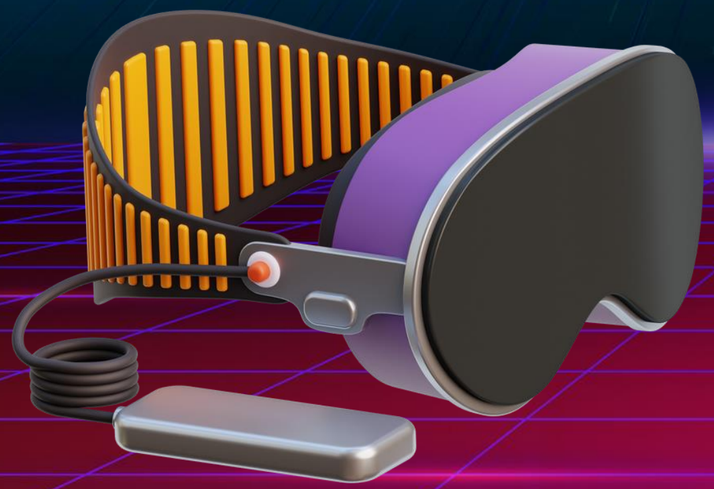


Data Wrangling

We clean and organize the data to make it ready for modeling.

Common steps:

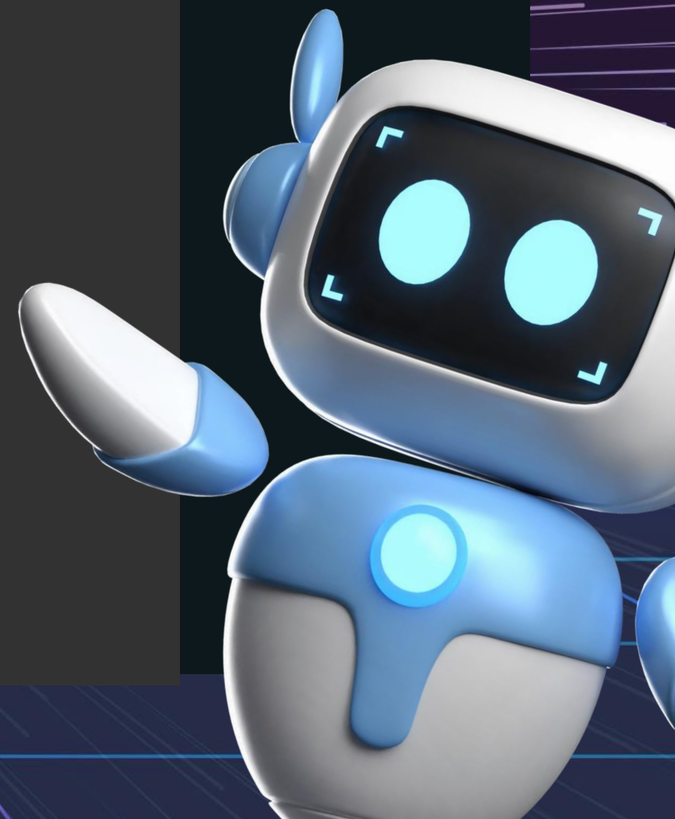
- Drop unneeded columns
- Drop duplicates
- Handle missing values



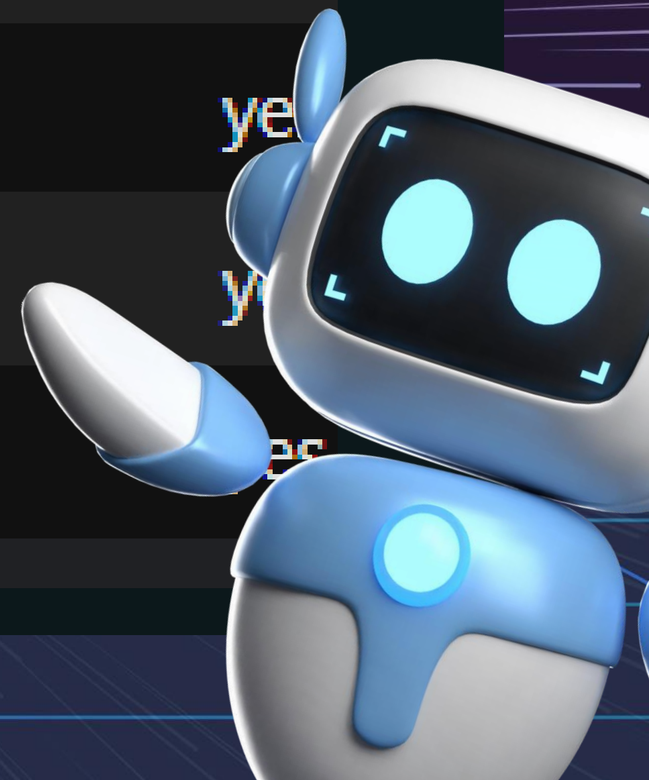
In our project:

- Checked for missing values using => [df.isnull().sum()]
- Cleaned the dataset before training the model

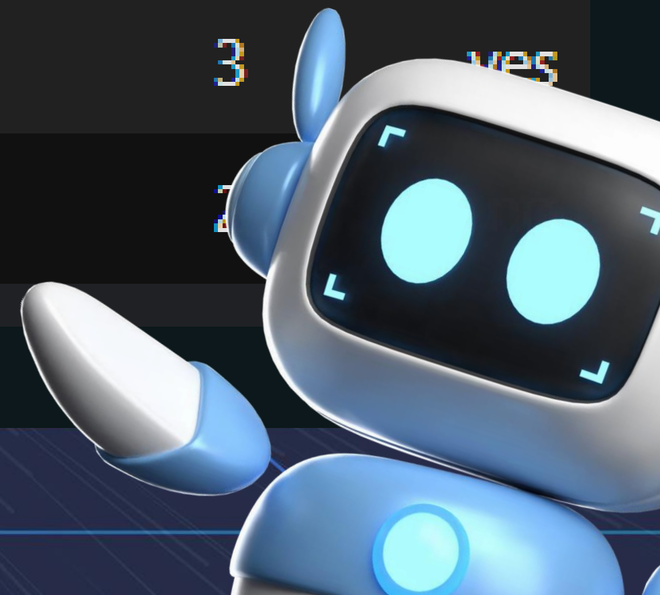
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, SGDRegressor, Lasso, Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error

from sklearn.svm import SVR, LinearSVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor, ExtraTreesRegressor

from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor


from sklearn.ensemble import VotingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
```



price	area	bedrooms	bathrooms	stories	mainroad
13300000	7420	4	2	3	yes
12250000	8960	4	4	4	yes
12250000	9960	3	2	2	ye
12215000	7500	4	2	2	y
11410000	7420	4	1	2	es

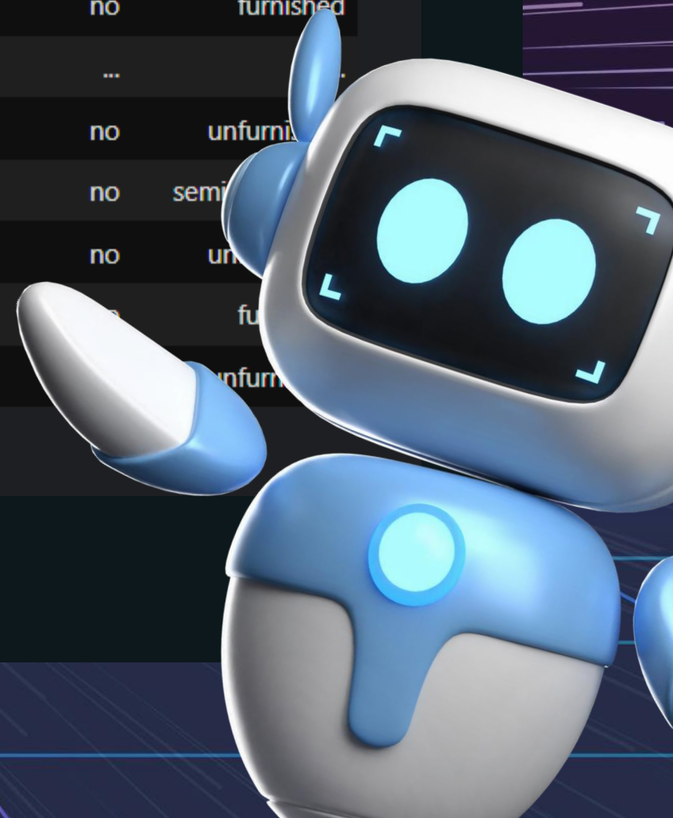


guestroom	basement	hotwaterheating	airconditioning	parking	prefarea
no	no	no	yes	2	yes
no	no	no	yes	3	no
no	yes	no	no	2	yes
no	yes	no	yes	3	yes
yes	yes	no	yes	2	

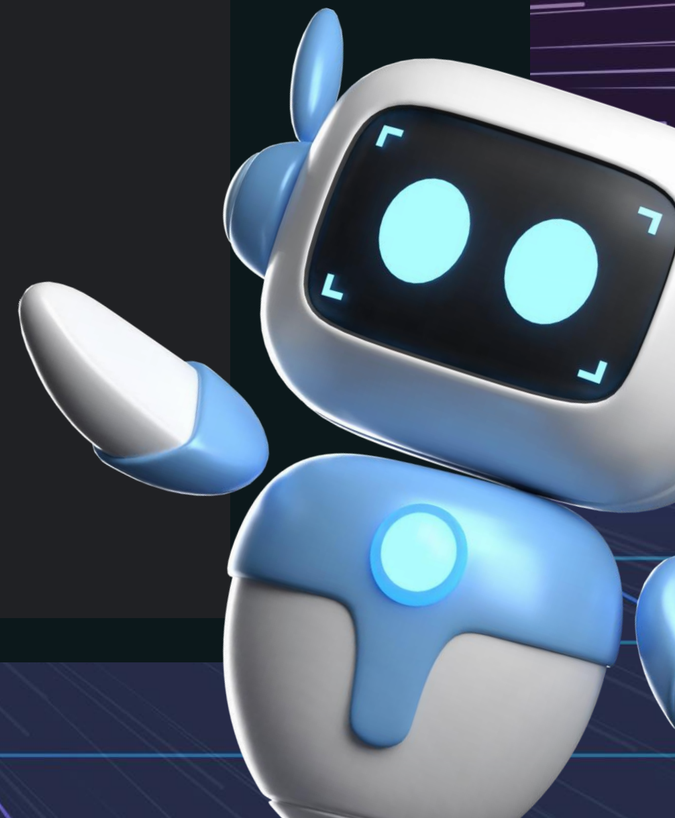


	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurni
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	un
543	1750000	2910	3	1	1	no	no	no	no	no	0		fu
544	1750000	3850	3	1	2	yes	no	no	no	no	0		nfurn

545 rows × 13 columns

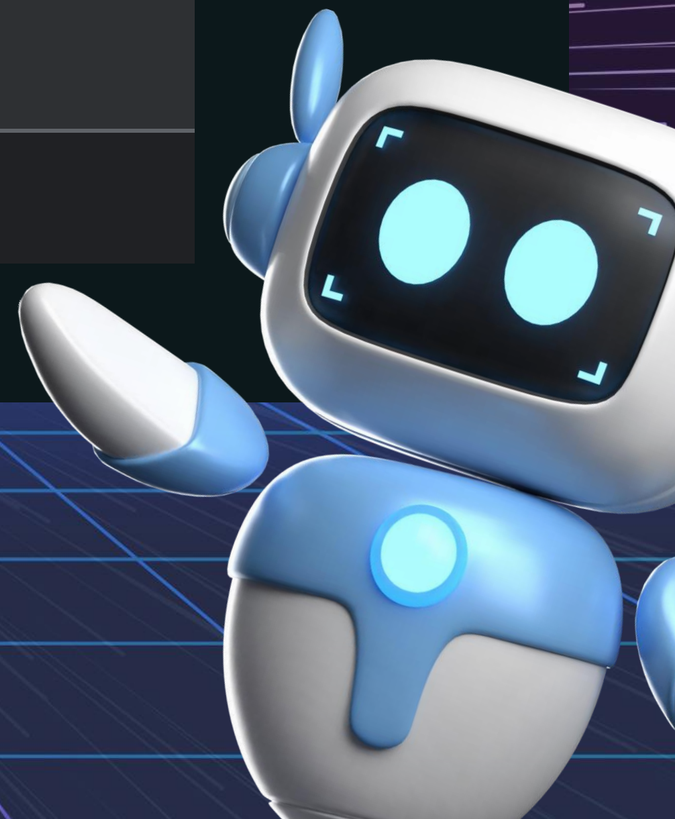


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```



```
df.duplicated().sum()
```

```
0
```



Data Analysis

We analyze the dataset to discover patterns, relationships, and trends that help us understand the data before building the model.

In our project:

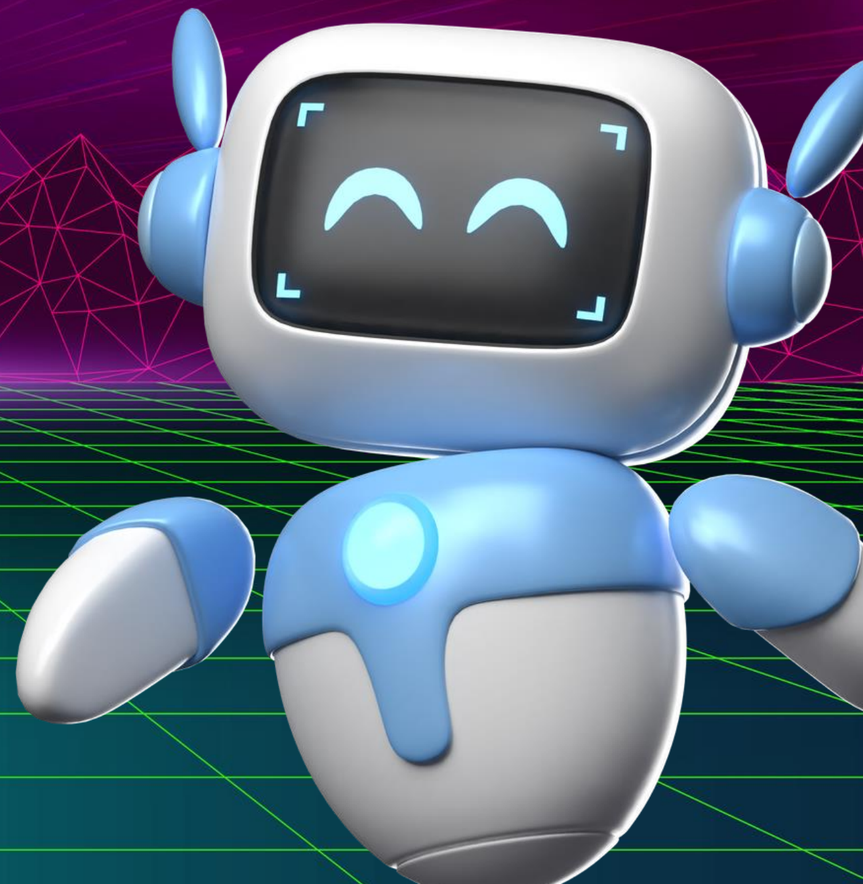
- Used .describe() for statistics
- Used plots and graphs to visualize price trends and feature relationships

Types of Data Analysis:

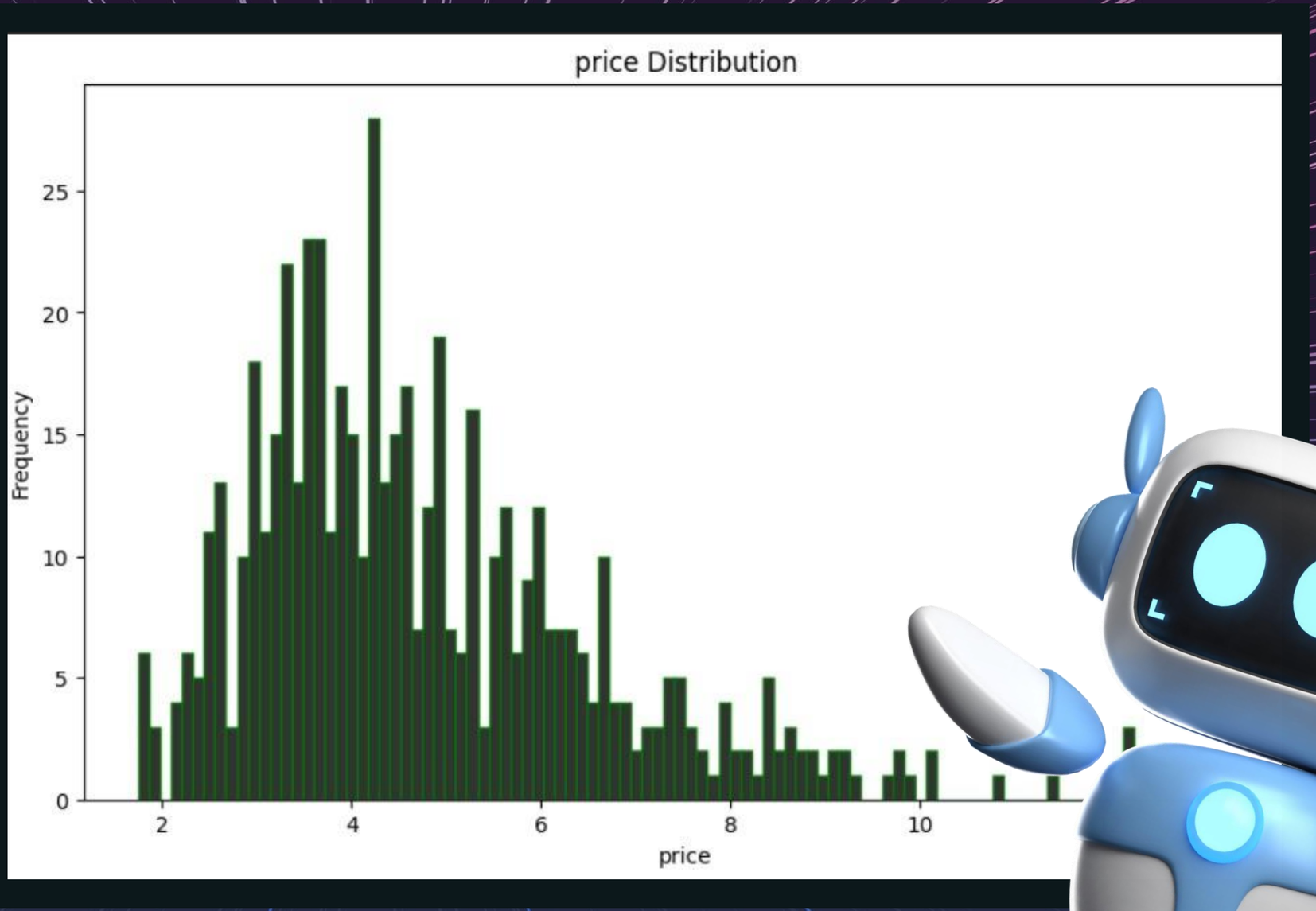
Univariate Analysis

Analysis of a single variable.

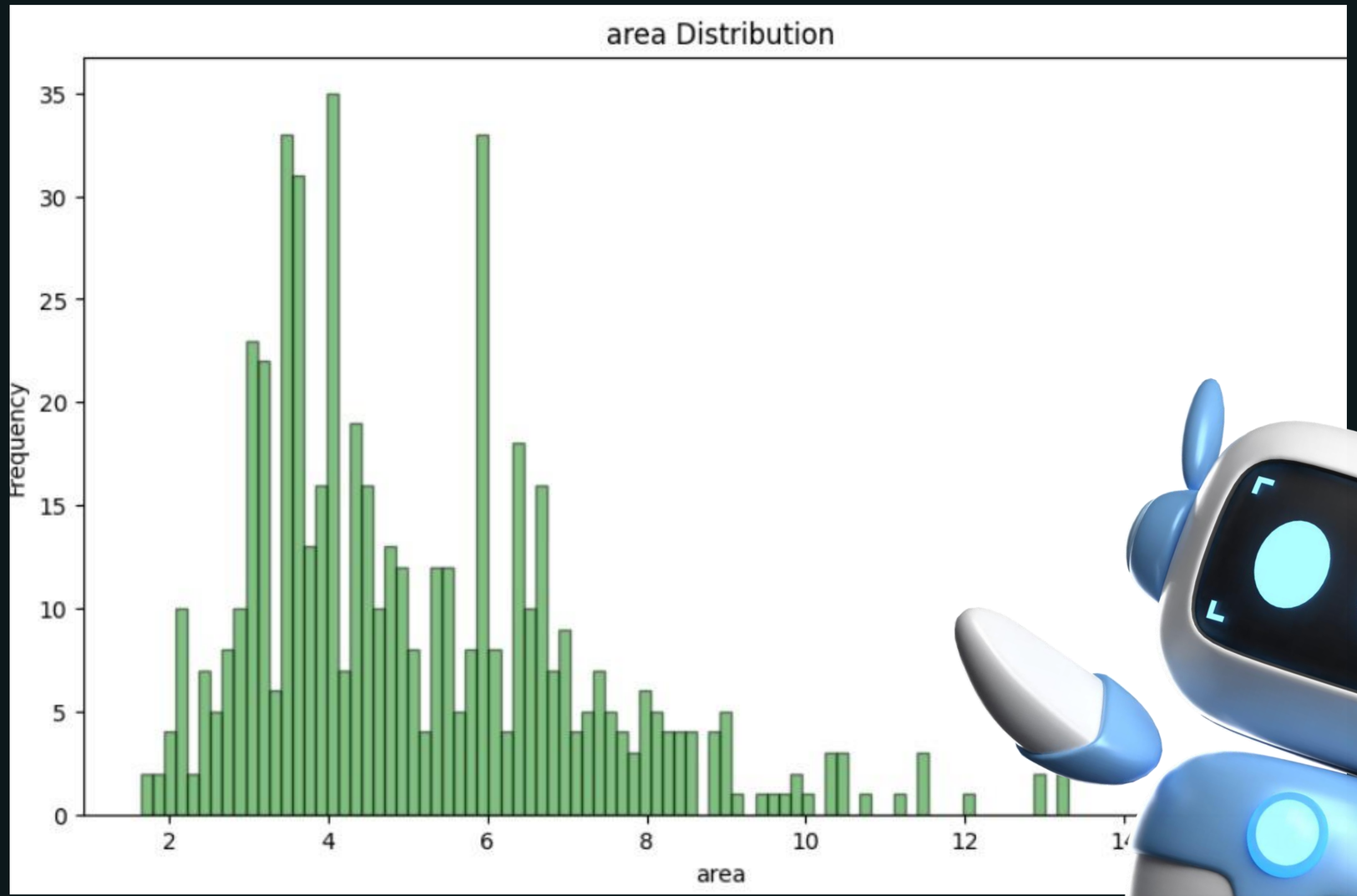
- Helps understand the distribution, range, and central tendency (e.g., average price of houses).



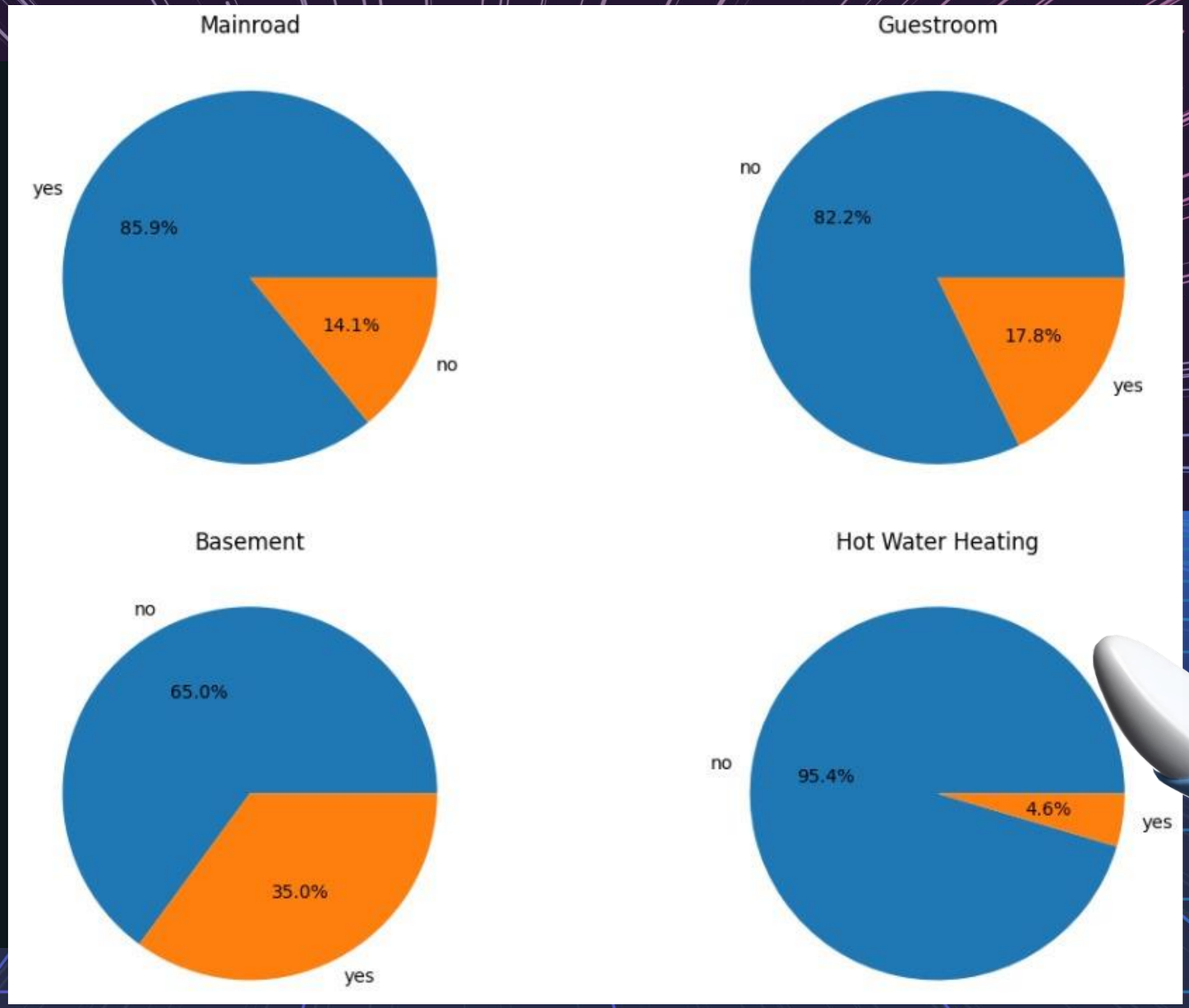
Histogram



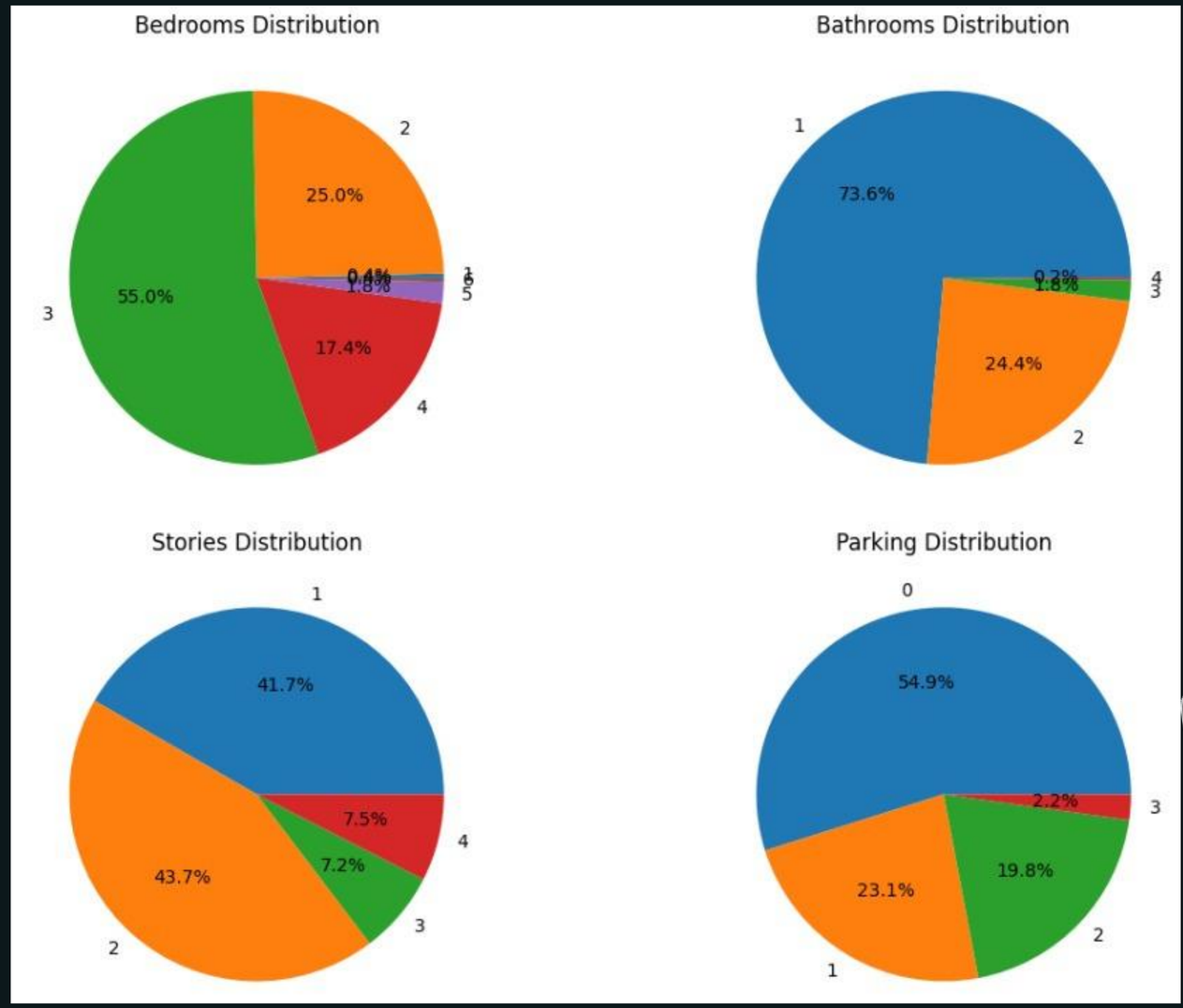
Histogram


area Distribution
Frequency
area
0
5
10
15
20
25
30
35
2
4
6
8
10
12

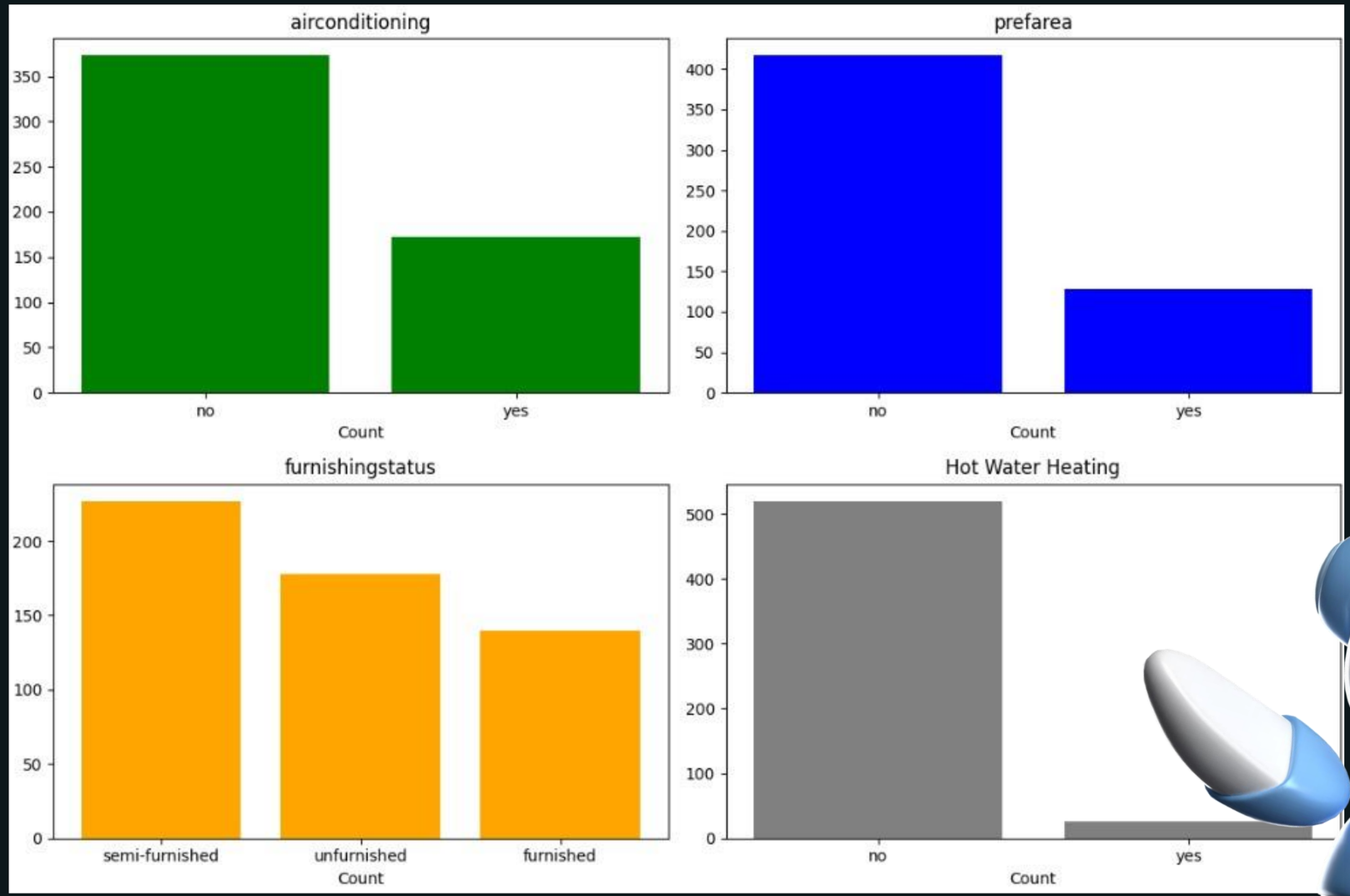
Pie Chart


Mainroad
yes
85.9%
14.1%
no
Guestroom
no
82.2%
17.8%
yes
Basement
no
65.0%
35.0%
yes
Hot Water Heating
no
95.4%
4.6%
yes

Pie Chart


Bedrooms Distribution
2
25.0%
1
6
5
1.8%
3
55.0%
17.4%
4
Bathrooms Distribution
1
73.6%
0.2%
4
3
1.8%
24.4%
2
Stories Distribution
1
41.7%
7.5%
4
43.7%
7.2%
3
2
Parking Distribution
0
54.9%
2.2%
3
23.1%
19.8%
2
1

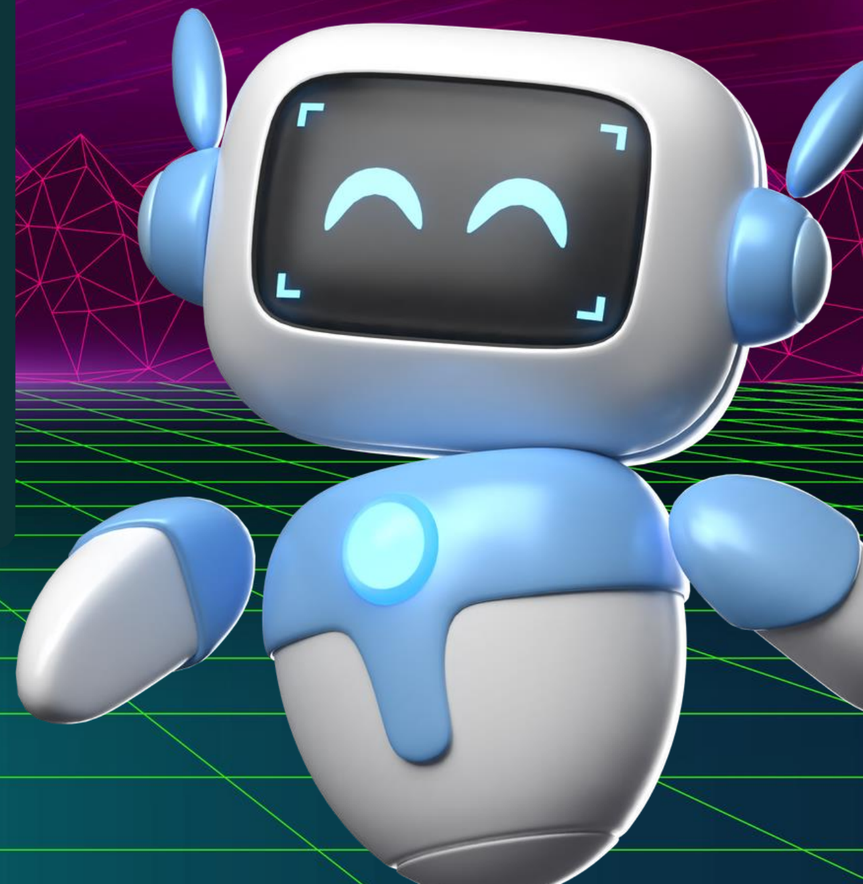
Bar Chart


airconditioning
350
300
250
200
150
100
50
0
no
yes
Count
prefarea
400
350
300
250
200
150
100
50
0
no
yes
Count
furnishingstatus
200
150
100
50
0
semi-furnished
unfurnished
furnished
Count
Hot Water Heating
500
400
300
200
100
0
no
yes
Count

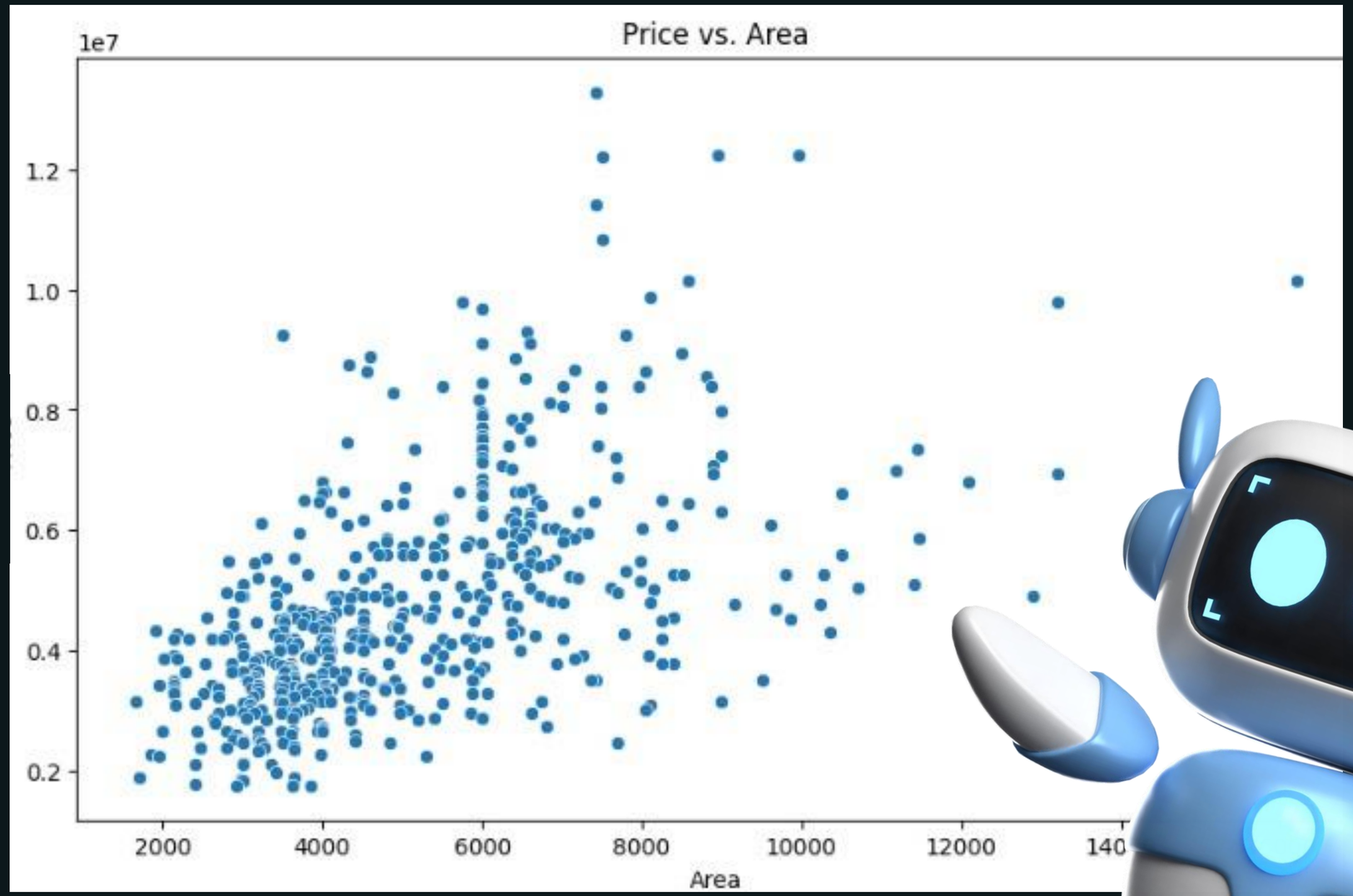
Bivariate Analysis

Analysis of two variables together.

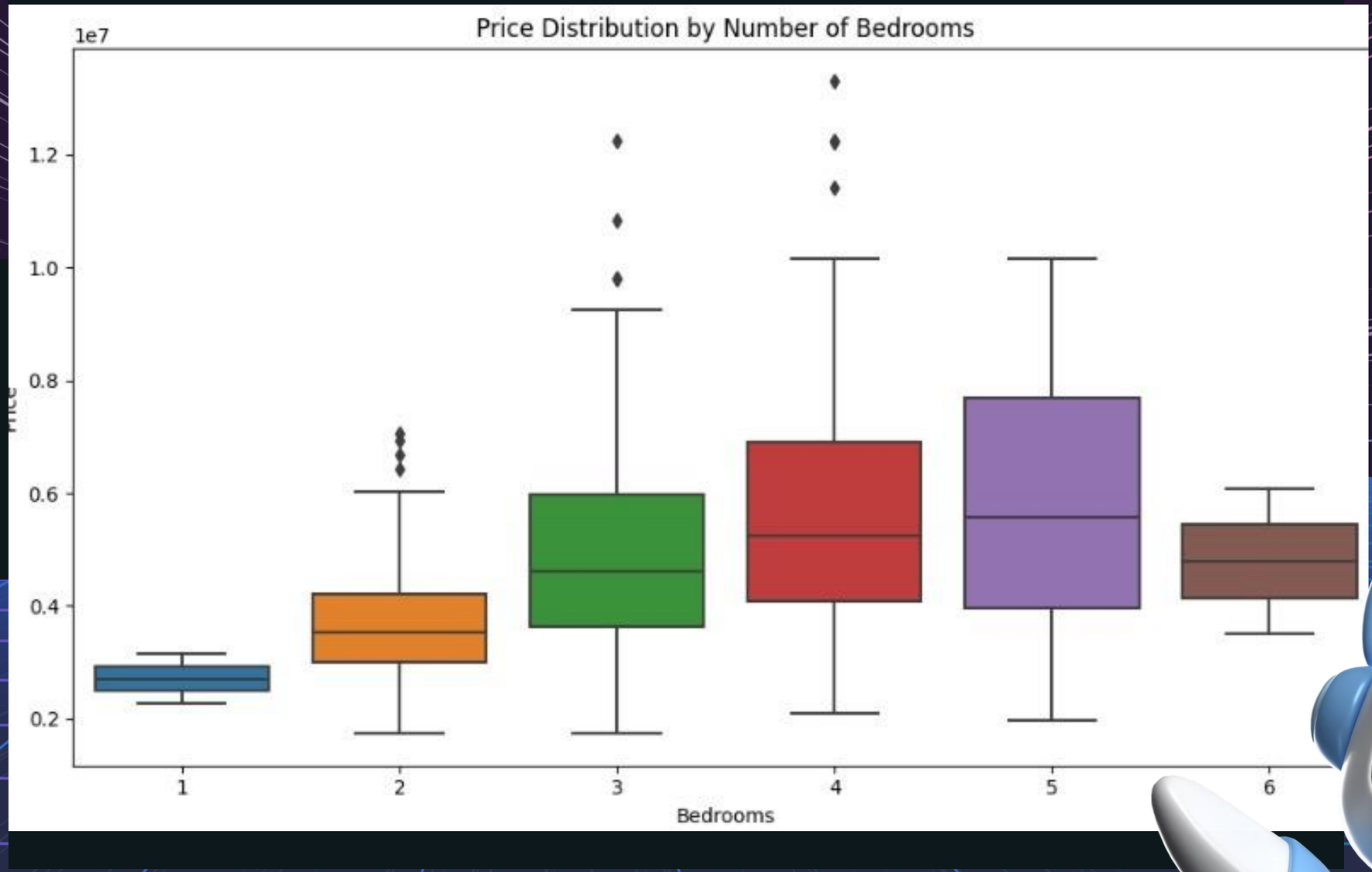
→ Helps identify relationships (e.g., how area affects house price).



scatterplot

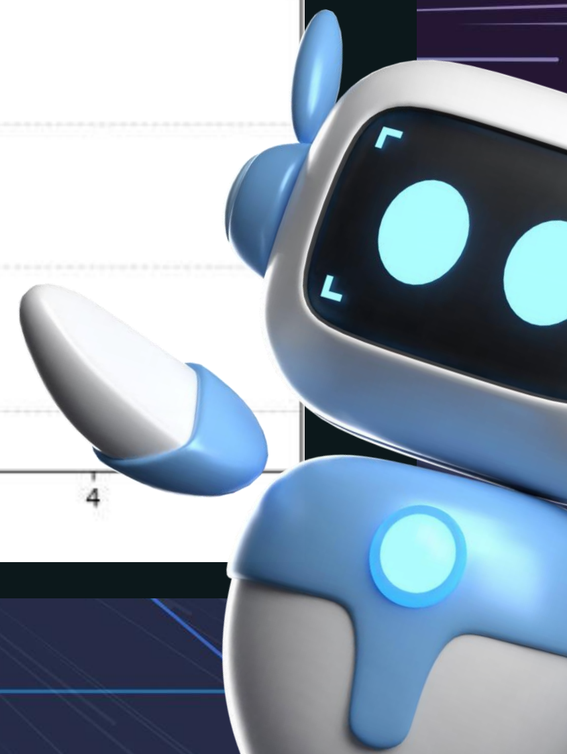


boxplot

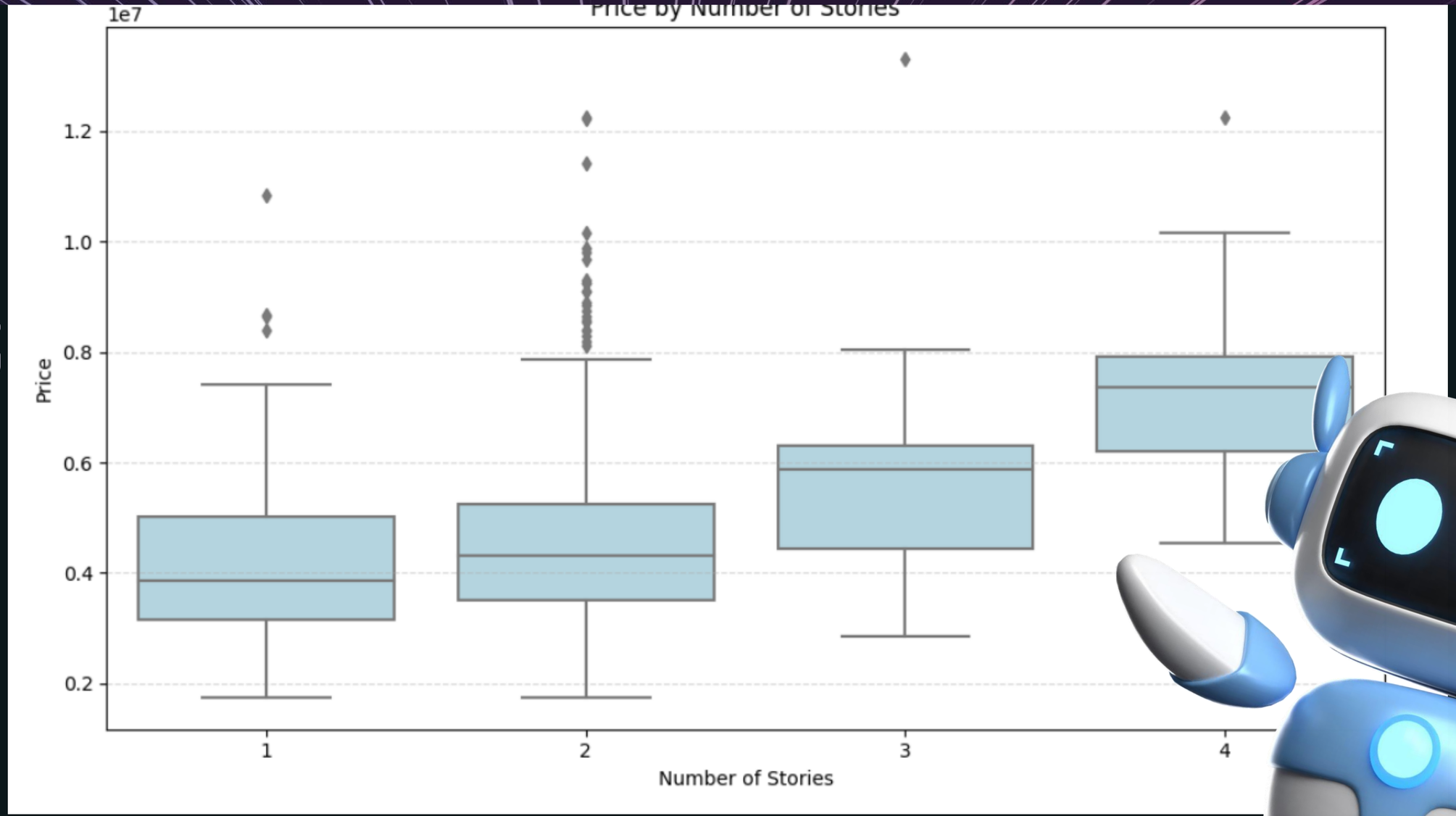

Price Distribution by Number of Bedrooms
1e7
1.2
1.0
0.8
0.6
0.4
0.2
1
2
3
4
5
6
Bedrooms

boxplot

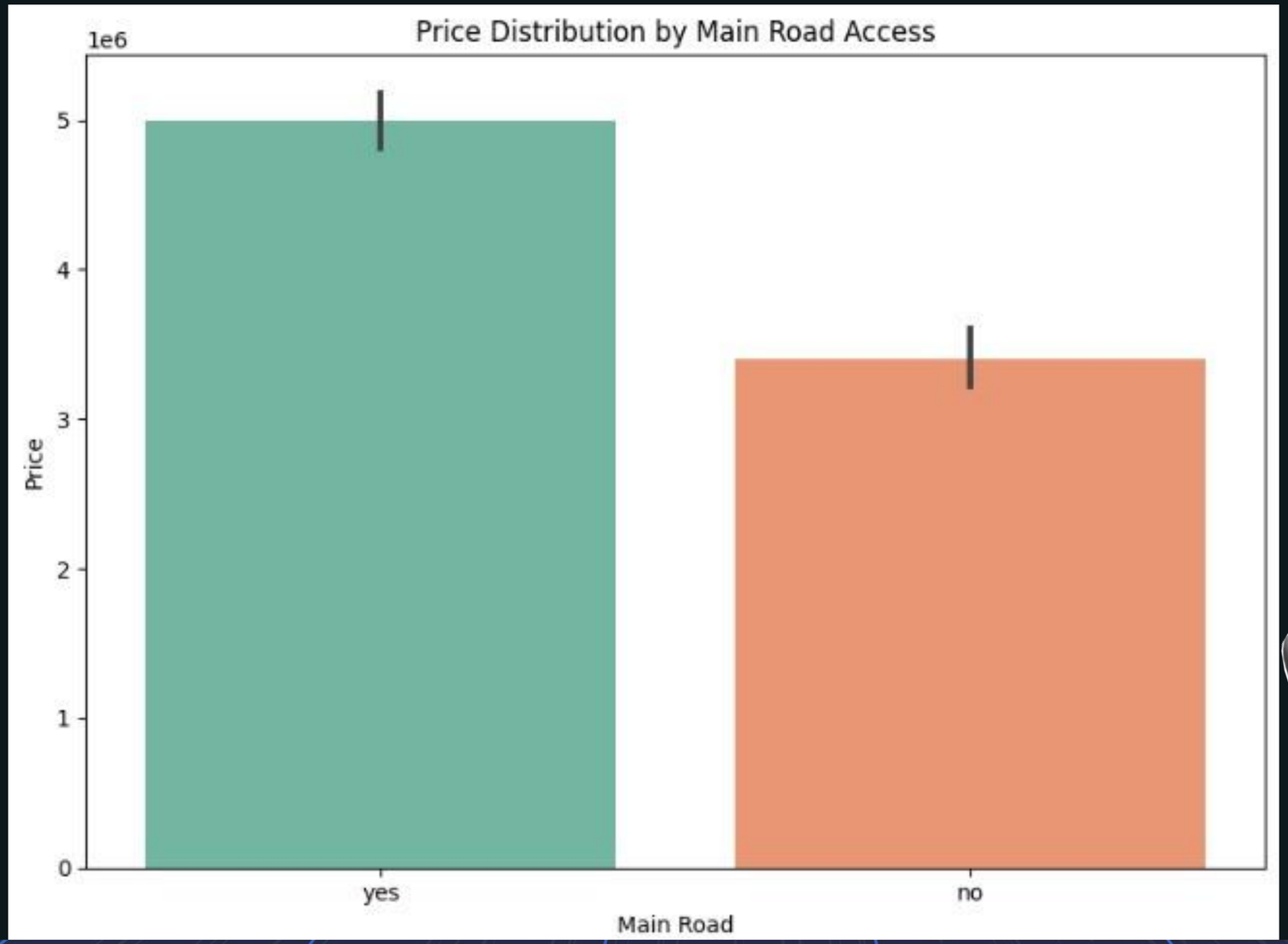

Price by Number of Bathrooms
1e7
1.2
1.0
0.8
0.6
0.4
0.2
Price
1
2
3
4
Number of Bathrooms

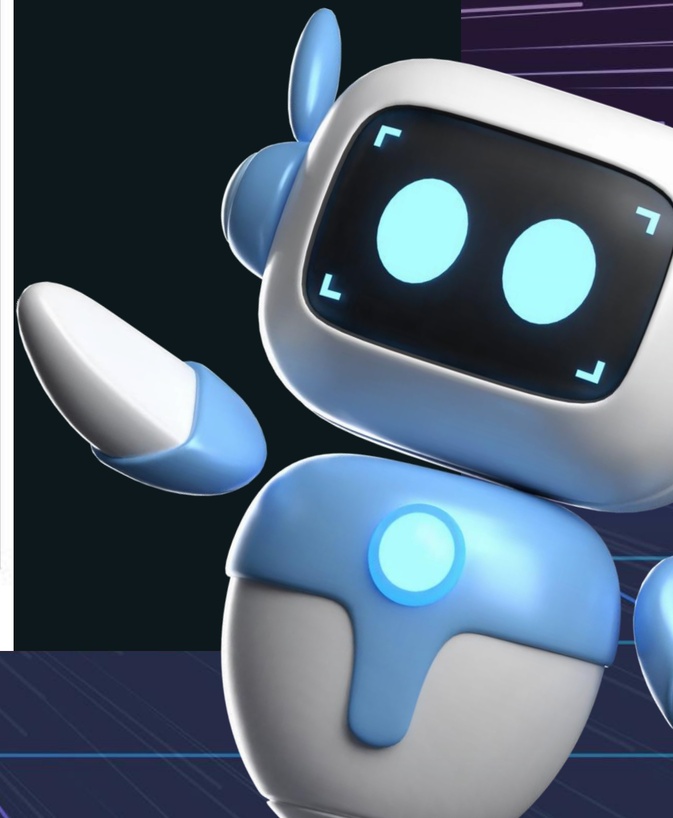


boxplot

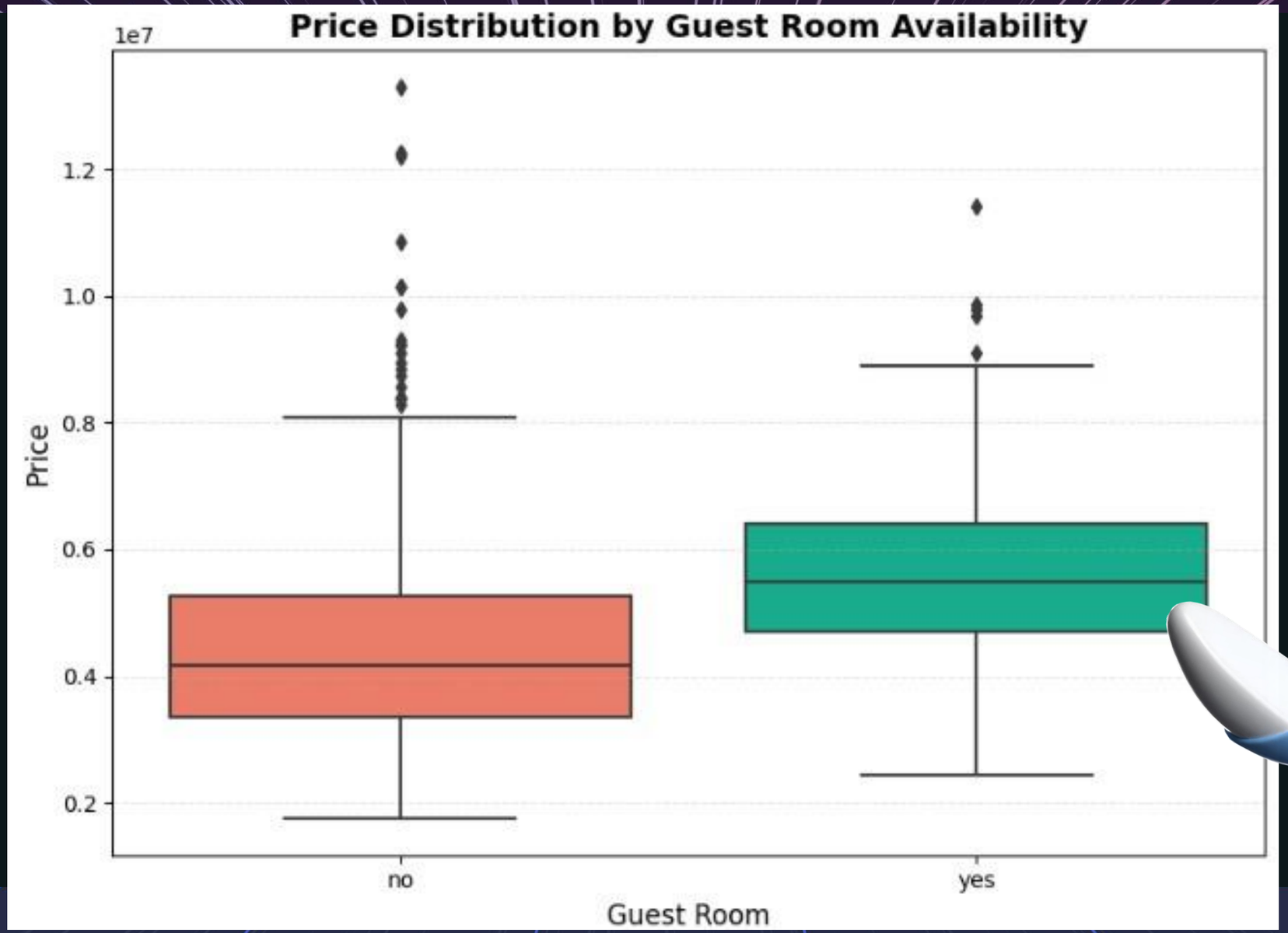

Price by Number of Stories
1e7
1.2
1.0
0.8
0.6
0.4
0.2
Price
1
2
3
4
Number of Stories

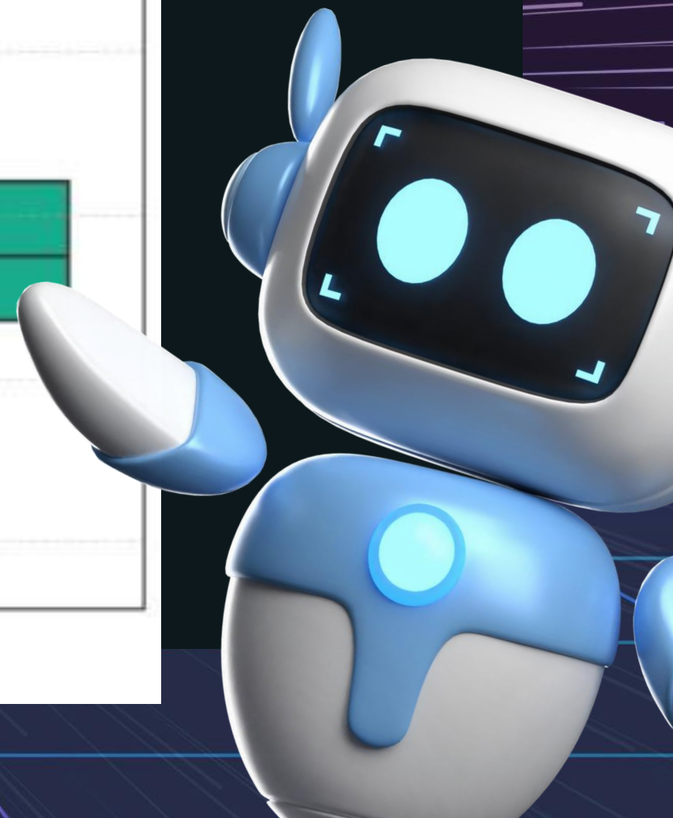
barplot


Price Distribution by Main Road Access
1e6
5
4
3
2
1
0
Price
yes
no
Main Road



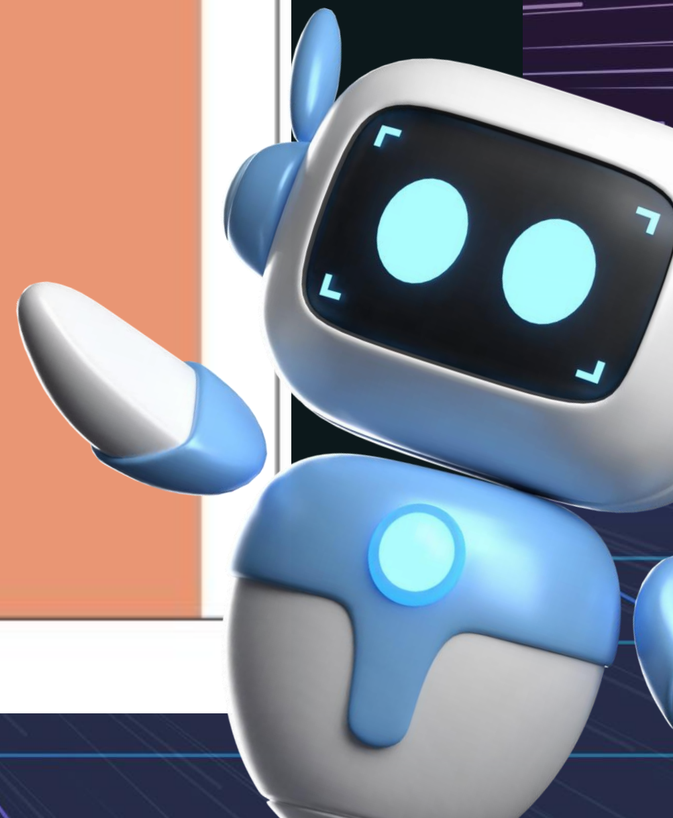
boxplot


Price Distribution by Guest Room Availability
1e7
1.2
1.0
0.8
0.6
0.4
0.2
Price
no
yes
Guest Room



barplot


Price Distribution by hotwaterheating Access
1e6
6
5
4
3
2
1
0
Price
no
yes
hotwaterheating

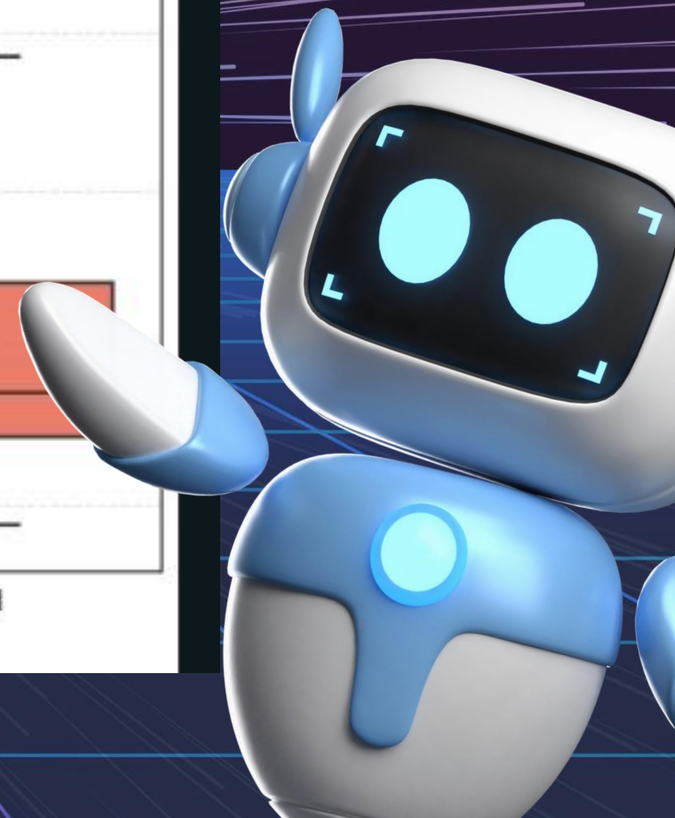


barplot


Price Distribution by airconditioning
1e7
1.2
1.0
0.8
0.6
0.4
0.2
price
yes
no
airconditioning

boxplot

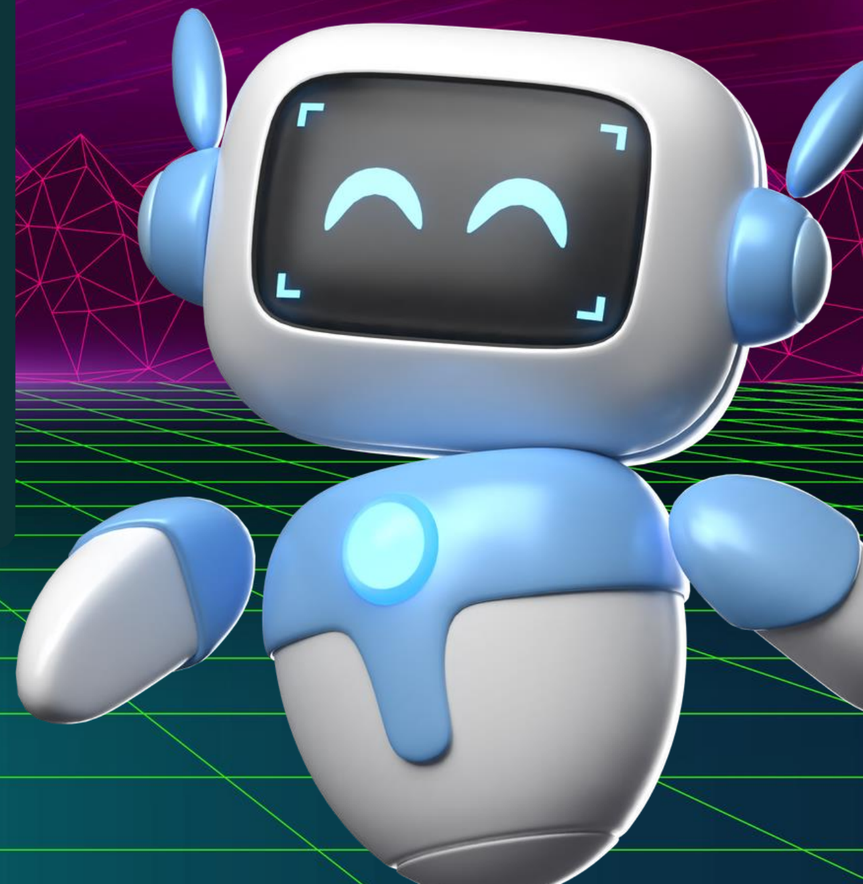

Price Distribution by Furnishing Status
1e7
1.2
1.0
0.8
0.6
0.4
0.2
Price
furnished
semi-furnished
unfurnished
Furnishing Status



Multivariate Analysis

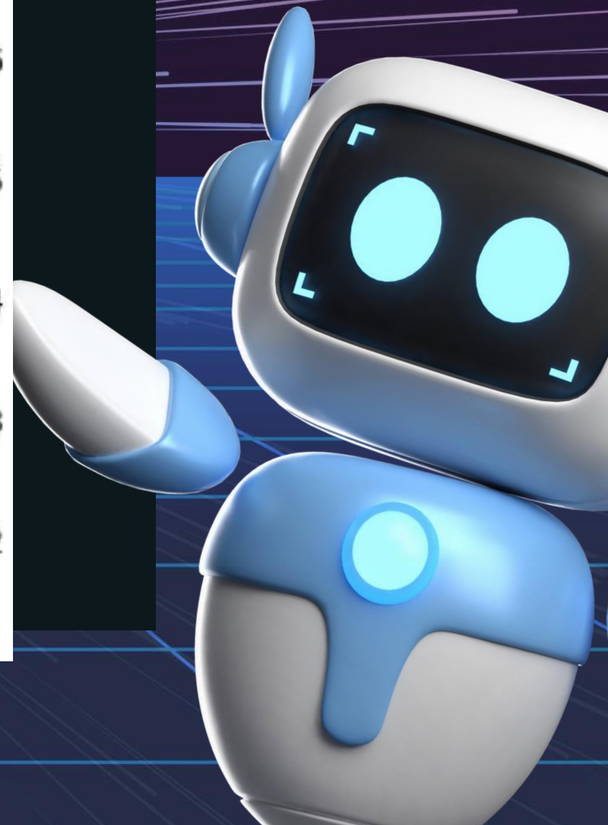
- Analysis involving more than two variables.

  → Helps understand complex interactions (e.g., area + bedrooms + location → price).

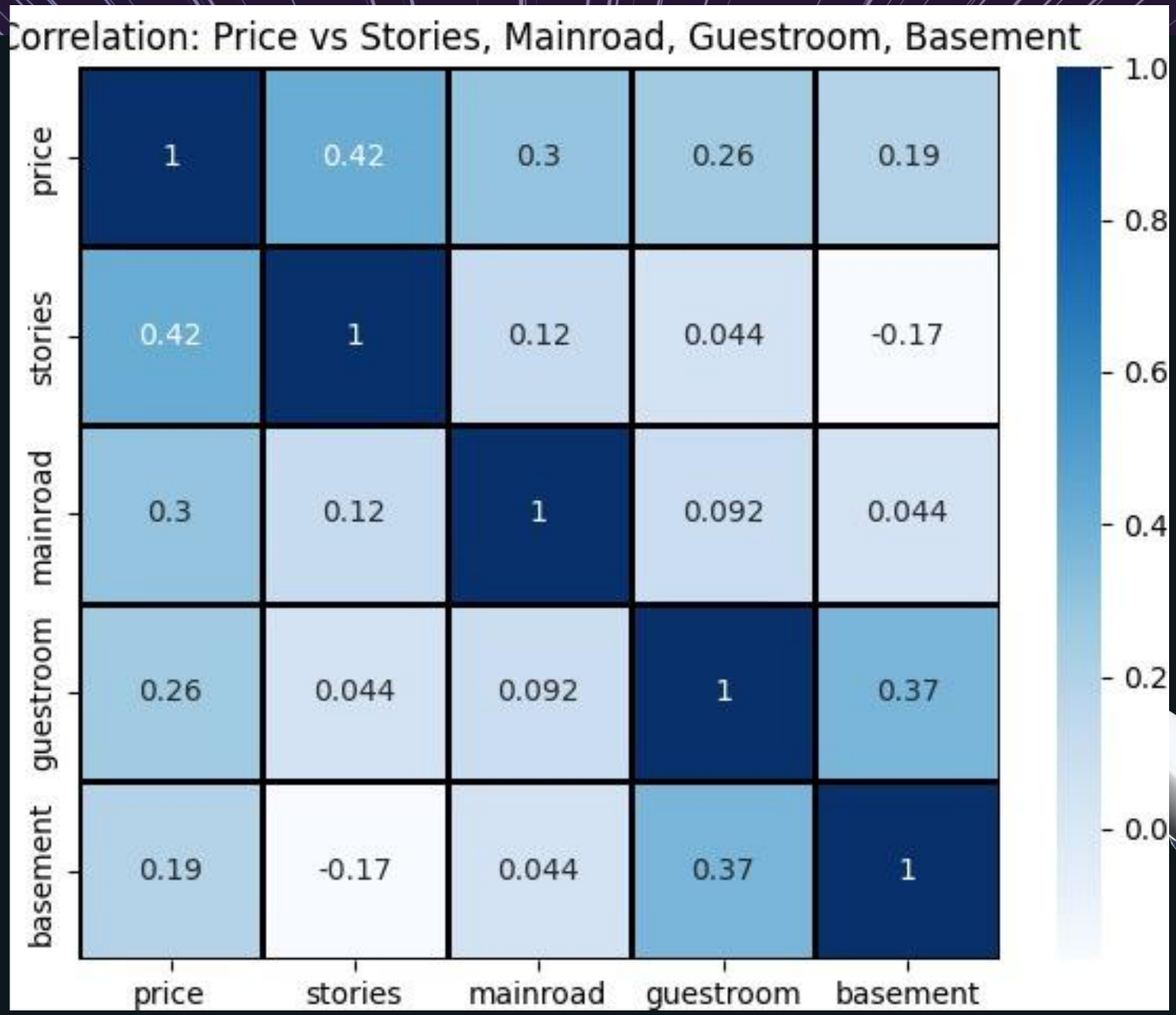


heatmap

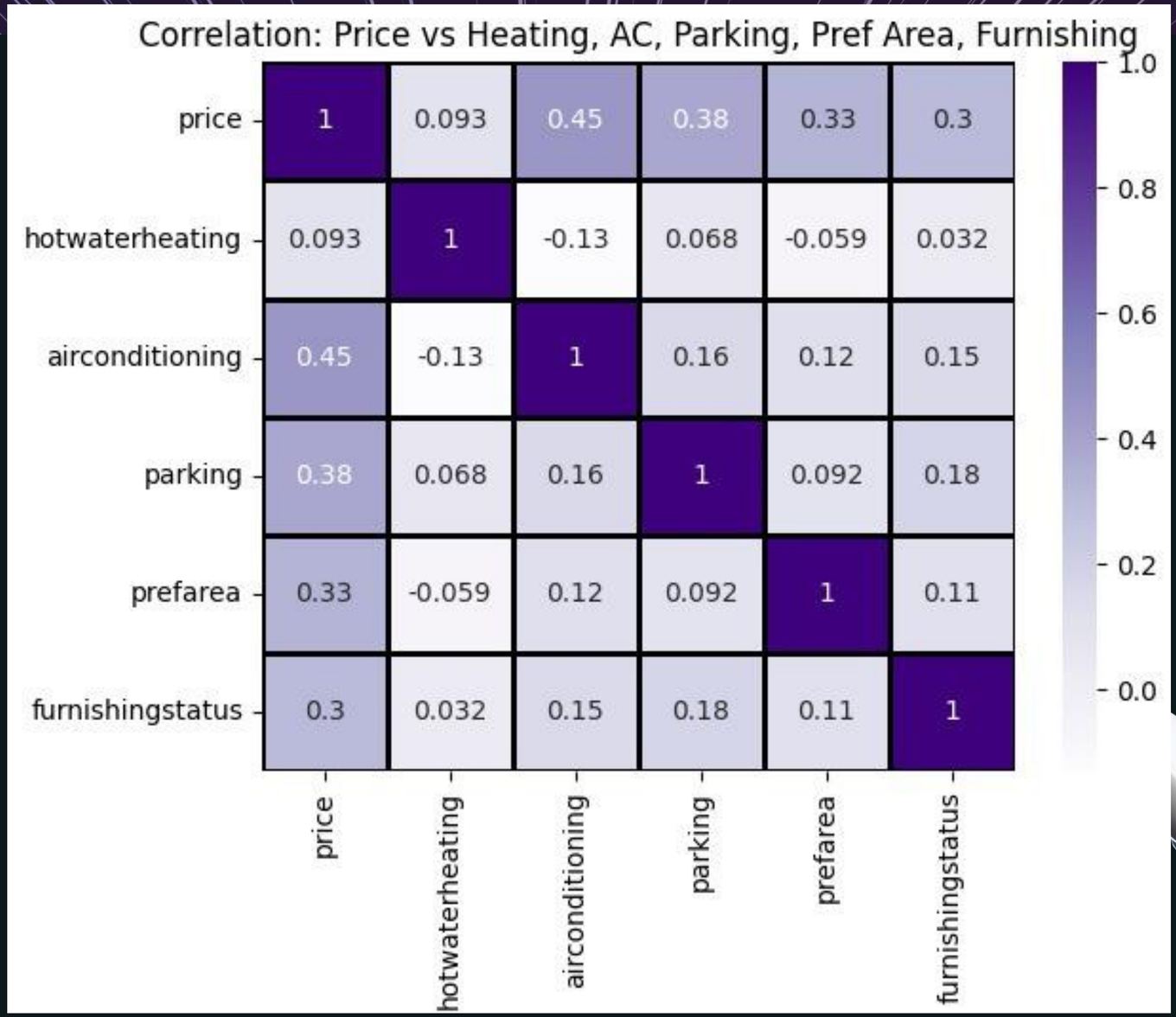

Correlation between Price and Area, Bedrooms, Bathrooms
price
area
bedrooms
bathrooms
1
0.54
0.37
0.52
0.54
1
0.15
0.19
0.37
0.15
1
0.37
0.52
0.19
0.37
1
price
area
bedrooms
bathrooms
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2



heatmap


Correlation: Price vs Stories, Mainroad, Guestroom, Basement
price
stories
mainroad
guestroom
basement
1
0.42
0.3
0.26
0.19
0.42
1
0.12
0.044
-0.17
0.3
0.12
1
0.092
0.044
0.26
0.044
0.092
1
0.37
0.19
-0.17
0.044
0.37
1
price
stories
mainroad
guestroom
basement
1.0
0.8
0.6
0.4
0.2
0.0

heatmap


Correlation: Price vs Heating, AC, Parking, Pref Area, Furnishing
price
hotwaterheating
airconditioning
parking
prefarea
furnishingstatus
1
0.093
0.45
0.38
0.33
0.3
0.093
1
-0.13
0.068
-0.059
0.032
0.45
-0.13
1
0.16
0.12
0.15
0.38
0.068
0.16
1
0.092
0.18
0.33
-0.059
0.12
0.092
1
0.11
0.3
0.032
0.15
0.18
0.11
1
1.0
0.8
0.6
0.4
0.2
0.0

Data Preprocessing

This step for machine learning by fixing issues and making it suitable for training.

Key Steps in Preprocessing:

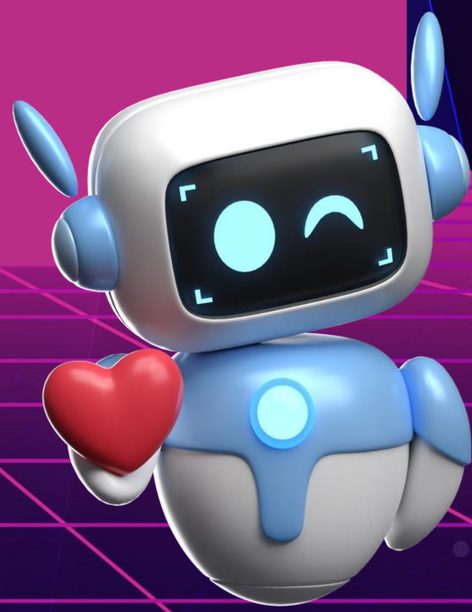
Key Steps:

- **Drop unneeded columns**
- **Handle outliers**
- **Encoding**: Convert text into numbers (e.g., "furnished" → 0,1,2)
- **Scaling:** Normalize features using MinMaxScaler or StandardScaler
- **Balancing**: Ensure fair representation of all classes if needed

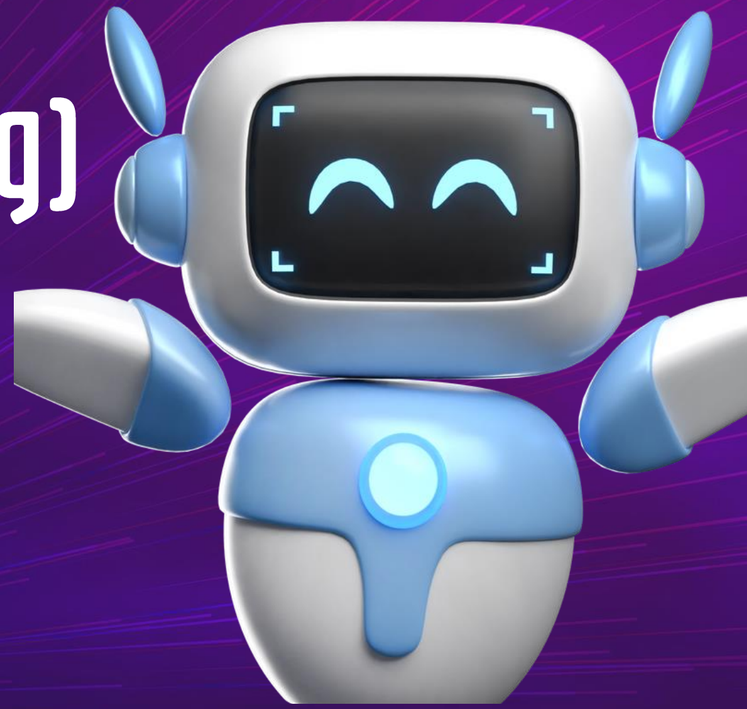
- **Splitting:**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2)
```

Label encoding and scaling were applied to prepare the dataset, and it was split into training and testing sets.



Model Building (Modeling)



Main Steps:

1. Training:

o Fit the model to training data to learn patterns

2. Testing:

o Evaluate the model on test data it hasn't seen before

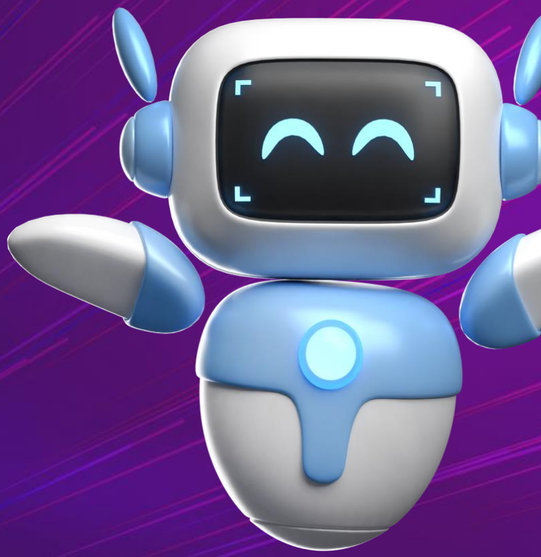
1.Evaluation:

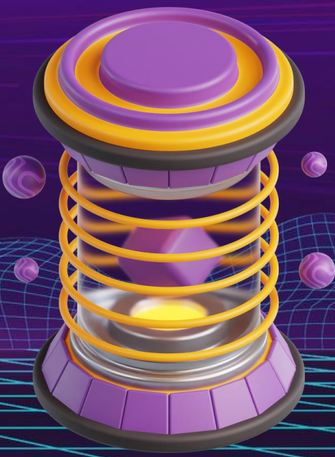
Measure performance using:

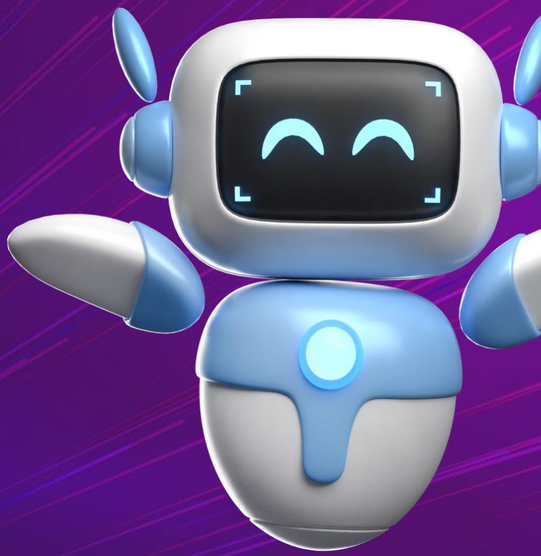
- MAE (Mean Absolute Error)
- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)

Regression models such as :

Linear Regression, Lasso, Ridge, Decision Tree, Random Forest, XGBoost, CatBoost, and LightGBM were implemented and tested. Each model’s predictions were compared using standard evaluation metrics.







Regression Models (Quick Overview)

🔷 Linear Models

LinearRegression: Fits a straight line to predict target values.

SGDRegressor: Uses fast gradient descent for large datasets.

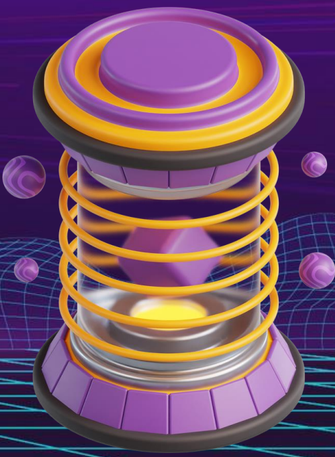
Lasso (L1): Shrinks some coefficients to zero for feature selection.

Ridge (L2): Shrinks all coefficients to reduce overfitting.

🔷 SVM Models

SVR: Uses support vectors for accurate regression, even nonlinear.

LinearSVR: A faster, linear-only version of SVR.



Ensemble Models

🔸 Bagging

BaggingRegressor: Averages results from multiple random models.

RandomForestRegressor: Many decision trees for better accuracy.

ExtraTreesRegressor: Adds randomness to improve speed and variance.

🔸 Boosting

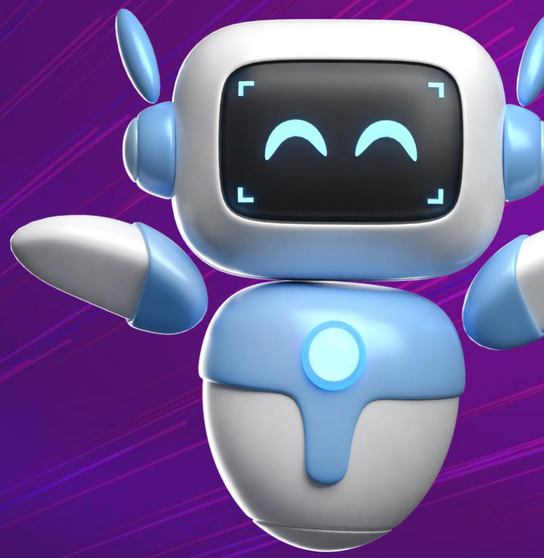
XGBoost: Builds trees that fix previous errors (high accuracy).

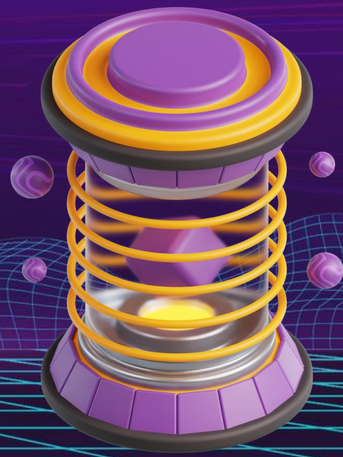
🔹 Distance-Based

KNeighborsRegressor: Predicts by averaging the K nearest neighbors.

🔹 Tree-Based

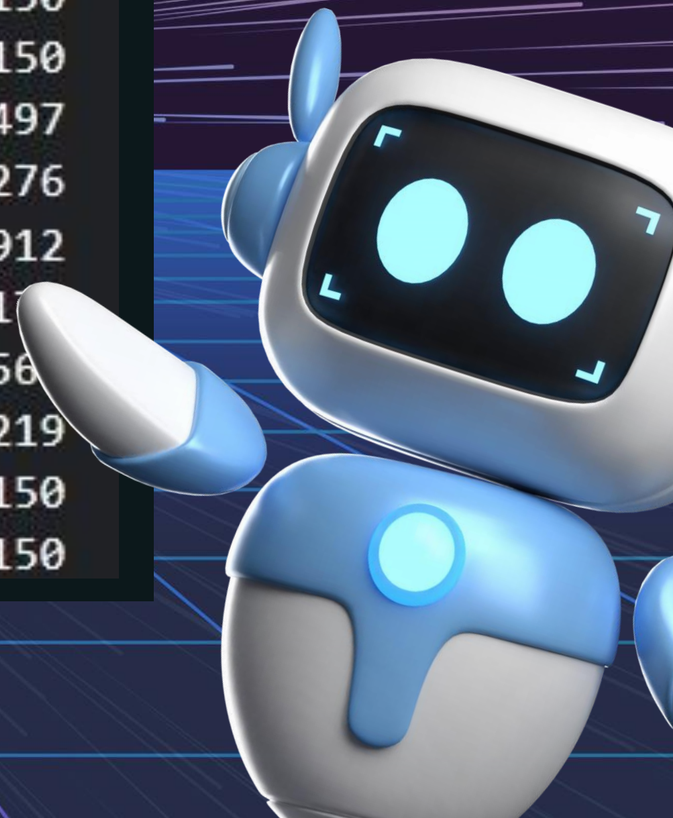
DecisionTreeRegressor: Splits data using rules; great for non-linear patterns.





results

	Model	MAE	RMSE	R2
0	SGDRegressor	1.747184e+12	NaN	NaN
1	Lasso Regression	8.986282e+05	NaN	NaN
2	Ridge Regression	8.999758e+05	NaN	NaN
3	Linear Regression	8.986293e+05	NaN	NaN
4	SVR-RBF	1.496433e+06	2.091369e+06	-0.041587
5	LinearSVR	4.758151e+06	5.180656e+06	-5.391509
6	KNN-5-distance	8.428207e+05	1.153479e+06	0.683150
7	DecisionTree	8.428207e+05	1.153479e+06	0.683150
8	ExtraTrees	8.825673e+05	1.218371e+06	0.646497
9	Bagging	9.017597e+05	1.234159e+06	0.637276
10	RandomForest	9.132300e+05	1.260026e+06	0.621912
11	CatBoost	9.010407e+05	1.289240e+06	0.6041
12	LightGBM	9.212714e+05	1.316014e+06	0.58756
13	XGBoost	9.841649e+05	1.351195e+06	0.565219
14	VotingRegressor	8.428207e+05	1.153479e+06	0.683150
15	StackingRegressor	8.428207e+05	1.153479e+06	0.683150

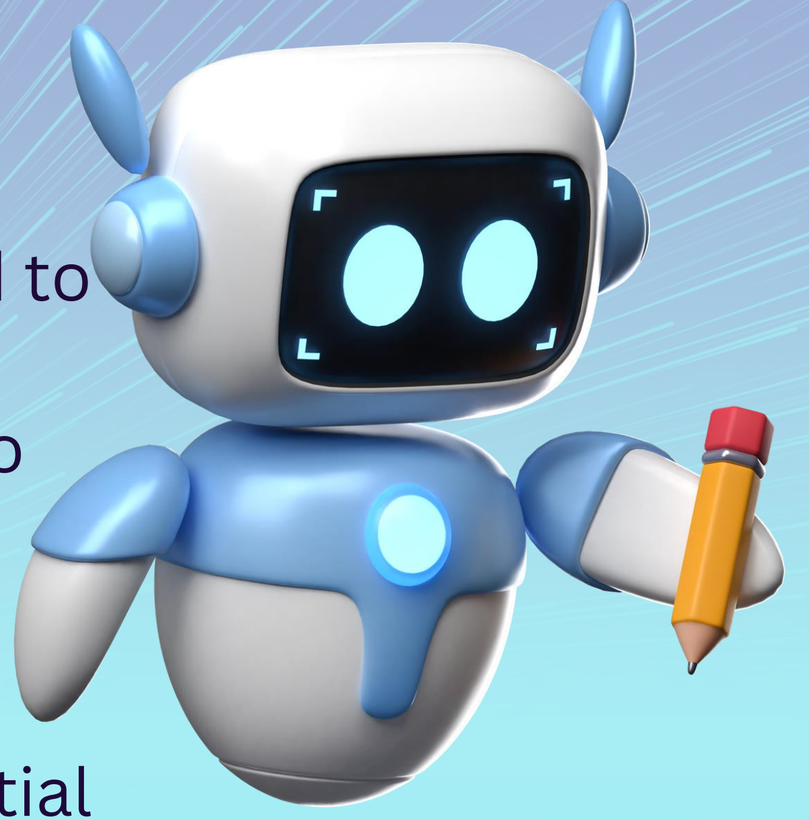


Conclusion

A complete machine learning pipeline was developed to predict house prices using structured tabular data. Every step, from data collection and preprocessing to modeling and evaluation, contributed to building an effective prediction model.

Key Learnings:

- Effective data cleaning and exploration are essential before modeling
- Feature encoding and scaling have a noticeable impact on prediction quality
- Comparing multiple algorithms helps select the most accurate one

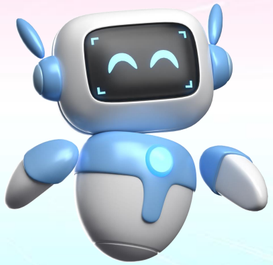


THANK YOU

Laila Mohamed Kamal Eldin

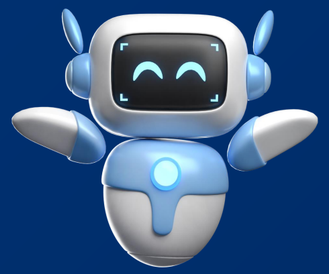
Mohamed Moharam Osman

Youssef Amr

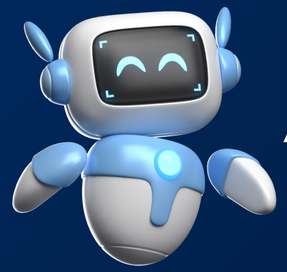


Abdelrahman Khaled Gaber

THANK YOU

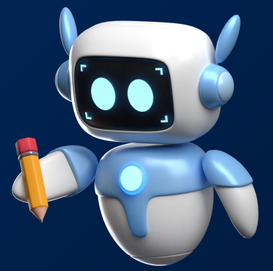


Laila Mohamed Kamal Eldin



Abdelrahman Khaled Gaber

Youssef Amr



Mohamed Moharam Osman

RESOURCES