PHASE -03

STUDENT NAME: *NAVYAA SHARMA*

REGISTER NUMBER: *2303717720522059*

INSTITUTION: *GOVERNMENT COLLEGE OF TECHNOLOGY*

DEPARTMENT: *INFORMATION TECHNOLOGY*

DATE OF SUBMISSION: *12-MAY-2025*

GITHUB REPOSITORY LINK: GO TO REPOSITORY*(CLICK HERE)*

1.PROBLEM STATEMENT:

Buying or selling a house can be tricky because it's hard to know the correct price. The price of a house depends on many things like its location, size, number of rooms, age, and nearby facilities. Guessing the price without proper data can lead to wrong decisions.

The goal of this project is to create a system that can predict the price of a house based on its features using machine Learning. This will help people make smarter choices when buying or selling houses.

2.ABSTRACT:

This project focuses on predicting house prices using linear regression to assist buyers and sellers in making informed decisions. The main objective is to build a model that estimates prices based on features like bedroom, bathroom, etc. The system was evaluated using performance metrics like R2 score and Mean Squared Error and mean absolute error. Results show that the model provides accurate and reliable predictions.

3.SYSTEM REQUIREMENTS:

Hardware Requirements:

Minimum 4 GB RAM 0r 8 GB

Processor: Intel Core i3 or higher

Storage: 2 GB of free disk space

Software Requirements:

Programming language: python

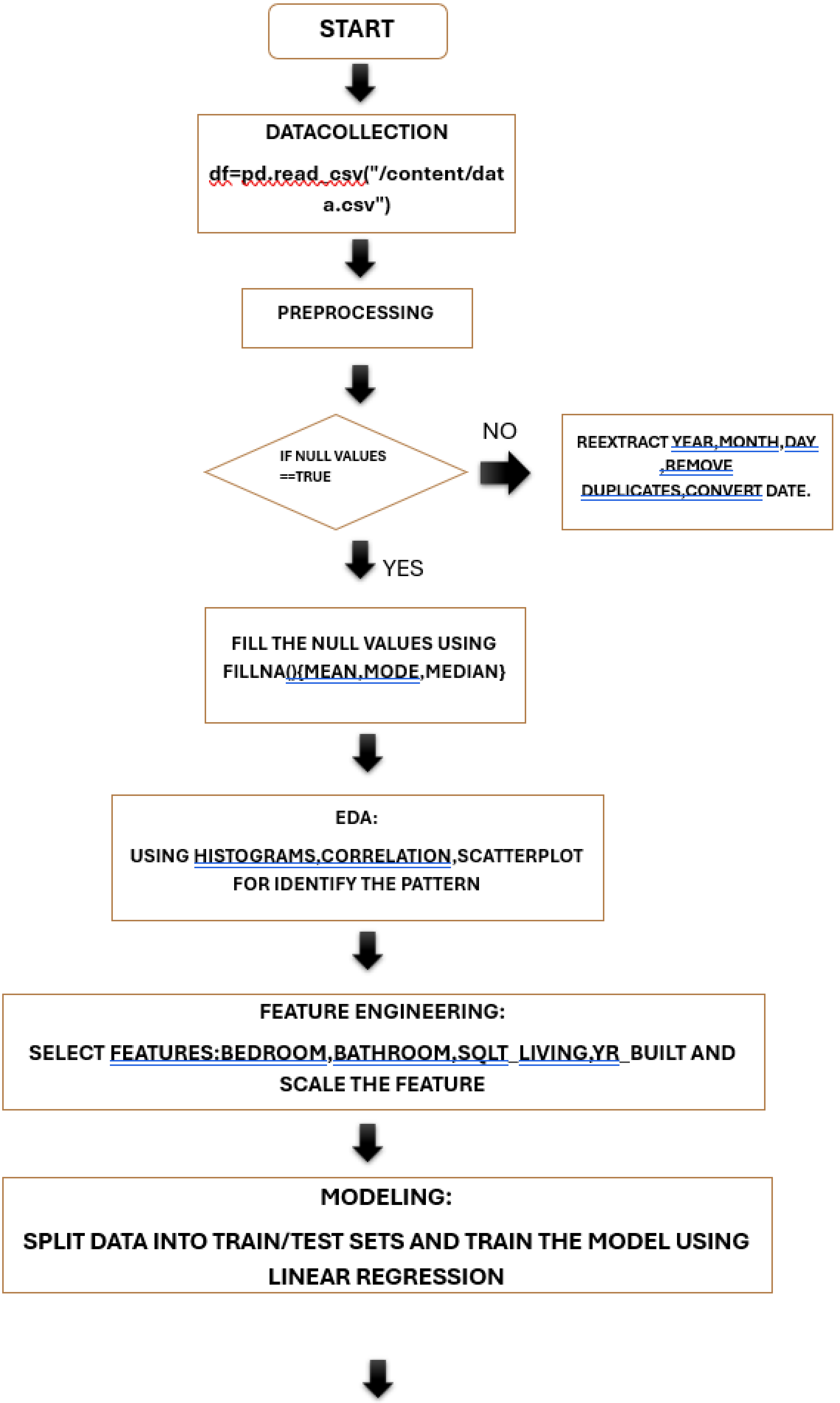
Required Libraries:

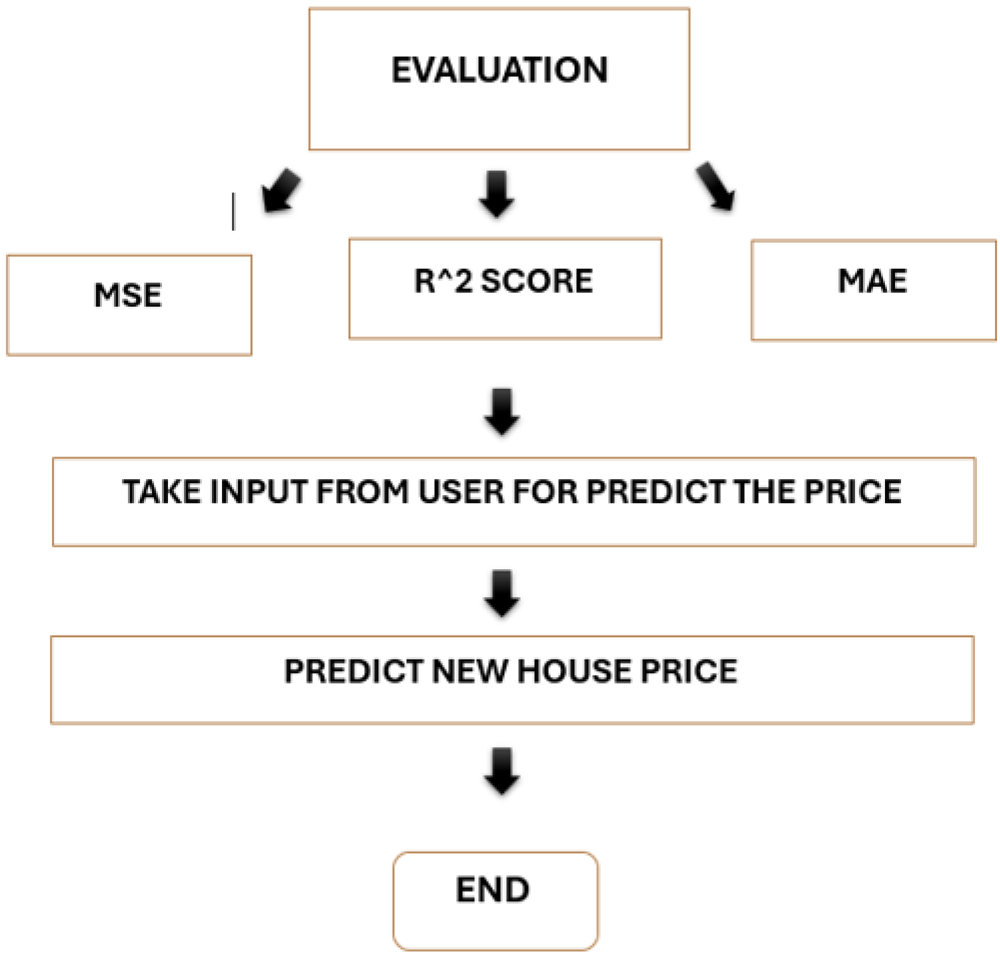
- ✓ **NumPy**
- ✓ **Pandas**
- ✓ **Matplotlib**
- ✓ **Seaborn**
- ✓ **Scikit-learn (linear model, standard scaler, accuracy metrics)**

4.OBJECTIVES:

The primary objective of this project is to develop a machine learning model that accurately predicts house prices based on various features such as bedroom,bathroom and other property details. The expected output is a predicted price value for a given set of input features. This prediction can help users make data-driven decisions in buying or selling properties.

5.FLOWCHART OR PROJECT WORKFLOW:


START
DATACOLLECTION
df=pd.read_csv("/content/data.csv")
PREPROCESSING
IF NULL VALUES ==TRUE
NO
REEXTRACT YEAR,MONTH,DAY ,REMOVE DUPLICATES,CONVERT DATE.
YES
FILL THE NULL VALUES USING FILLNA(){MEAN,MODE,MEDIAN}
EDA:
USING HISTOGRAMS,CORRELATION,SCATTERPLOT FOR IDENTIFY THE PATTERN
FEATURE ENGINEERING:
SELECT FEATURES:BEDROOM,BATHROOM,SQLT_LIVING,YR_BUILT AND SCALE THE FEATURE
MODELING:
SPLIT DATA INTO TRAIN/TEST SETS AND TRAIN THE MODEL USING LINEAR REGRESSION



6.DATASET DESCRIPTION:

- ✓ **We used the House Sales in King County, USA dataset, which is a well known dataset available on Kaggle.**
- ✓ **Type of Data: The dataset is structured, consisting of tabular data with both numerical and categorical variables that describe the properties.**
- ✓ **Number of Records and Features: There are around 21,600 records (rows) and 21 features (columns) .**

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

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_re
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	

7.DATA PREPROCESSING:

Handling Missing Values:

The dataset was checked for missing values using df.isnull().sum(). Fortunately, there were no major missing values in the selected features, so no imputation was needed.

Removing Duplicates:

Duplicates were removed using df.drop_duplicates (inplace=True), ensuring data integrity.

FEATURE SCALING:

Numerical features were standardized using StandardScaler() from sklearn, which scales data to a mean of O and standard deviation of 1.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   datetime64[ns]
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
 18  year           4600 non-null   int32
 19  month          4600 non-null   int32
 20  day            4600 non-null   int32
dtypes: datetime64[ns](1), float64(4), int32(3), int64(9), object(4)
memory usage: 700.9+ KB
```

```
print("\nmissing values:\n",df.isnull().sum())
```

```

missing values:
 date            0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
year             0
month            0
day              0
dtype: int64
```

```
print("duplicated rowes before:",df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("duplicated rowes after:",df.duplicated().sum())
```

```
duplicated rowes before: 0
duplicated rowes after: 0
```

```
df.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_base
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.00
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240652	3.451739	1827.265435	312.08
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778405	0.677230	862.168977	464.13
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	370.000000	0.00
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000	1190.000000	0.00
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000000	3.000000	1590.000000	0.00
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000	2300.000000	610.00
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	9410.000000	4820.00

8.EXPLORATORY DATA ANALYSIS:

Histogram:

Used to observe the distribution of numerical features like price, bedrooms, and sqft_living.

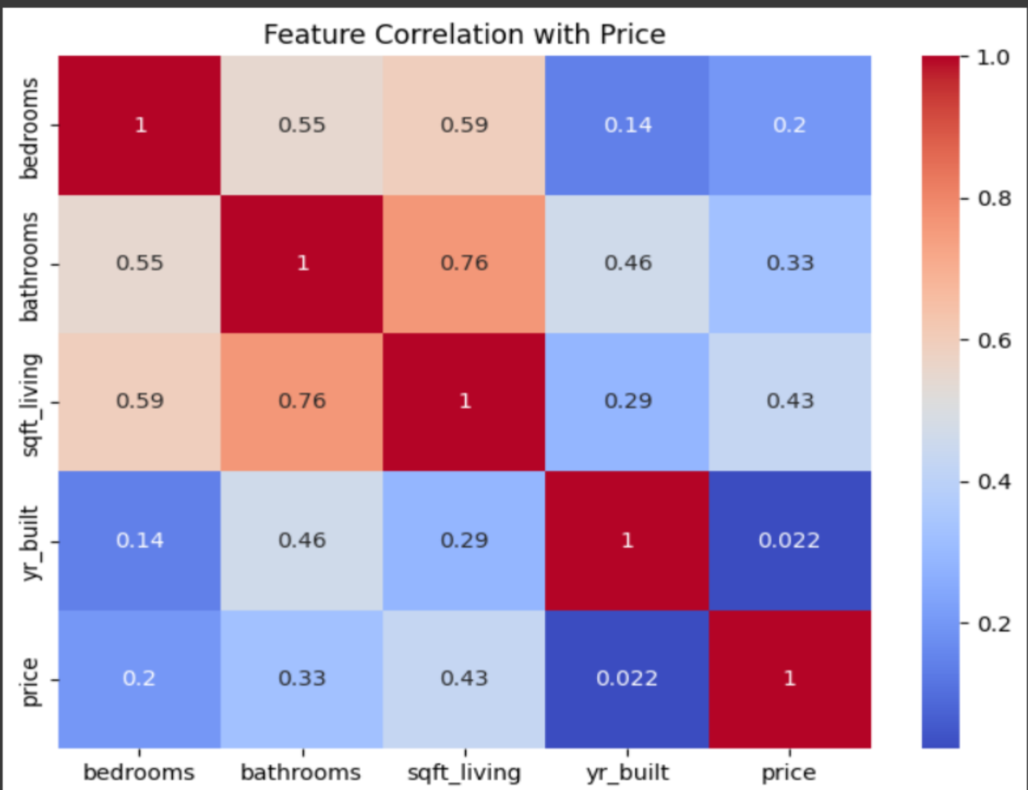
Insight: Most houses are priced under $600,000 and have 2-4 bedrooms.

Heatmap (Correlation Matrix):

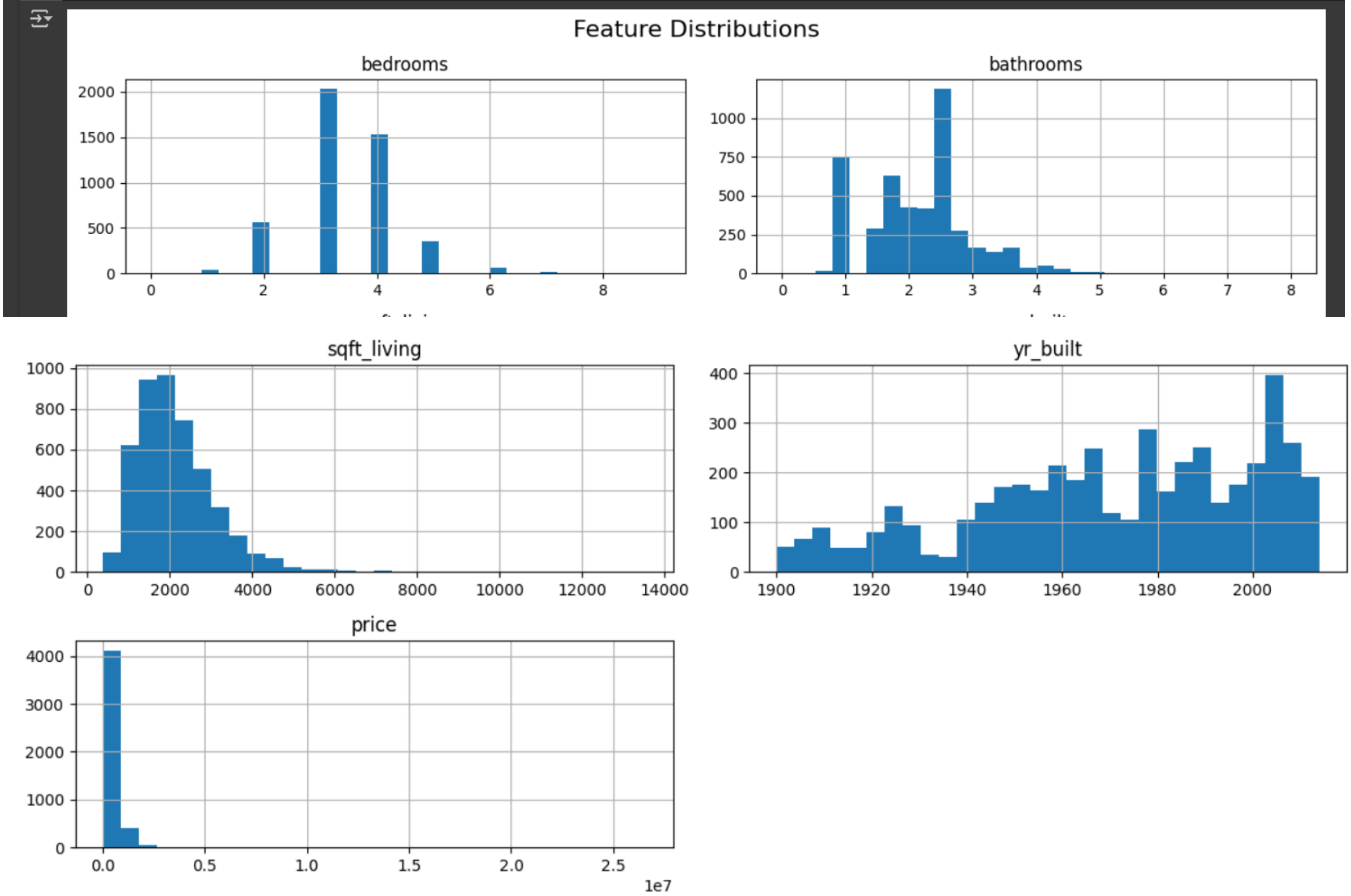
Used to identify relationships between features.

Insight: sqft_living showed a strong positive correlation with price, making it a key predictor. yr_built had a weaker but noticeable influence

```
plt.figure(figsize=(8,6))
sns.heatmap(df [features +['price']].corr(), annot=True,cmap='coolwarm')
plt.title("Feature Correlation with Price")
plt.show()
```



```python
features=['bedrooms','bathrooms','sqft_living','yr_built']
x=df[features]
y=df['price']
df [features + ['price']].hist(bins=30,figsize=(12,8))
plt.suptitle("Feature Distributions", fontsize=16)
plt.tight_layout()
plt.show()
```

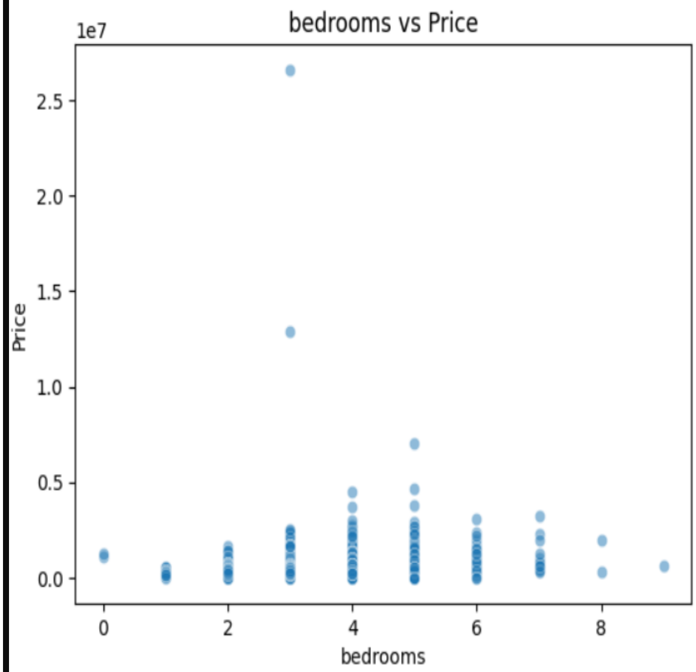


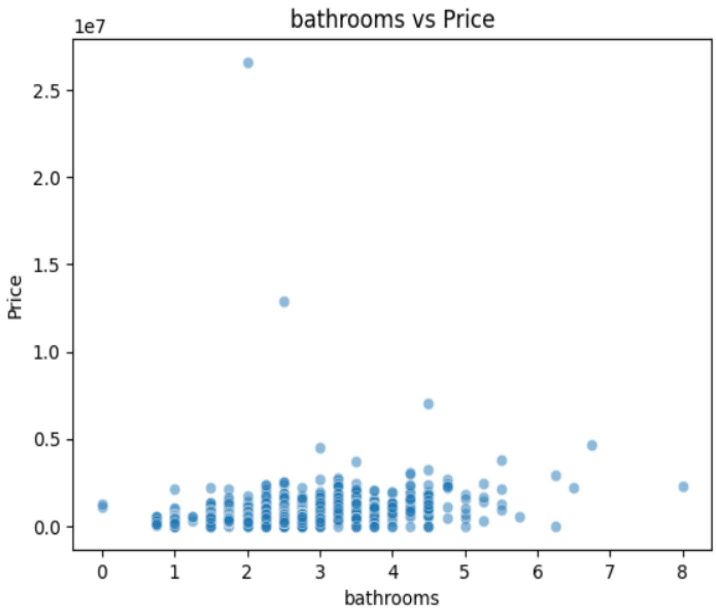
Scatter plot:

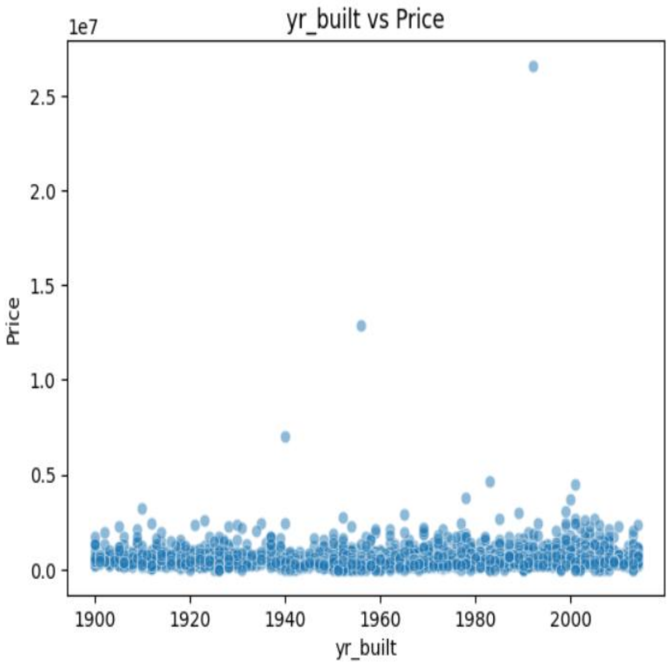
To visualize how house prices vary with square footage

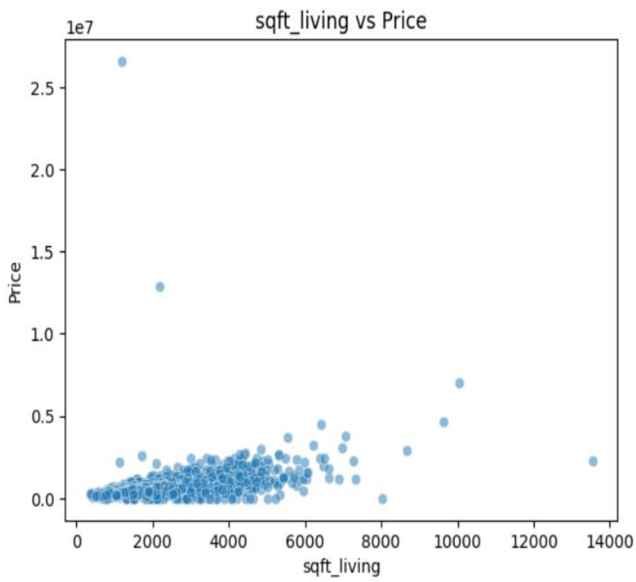
Insight:

This plot usually shows a positive trend—as square footage increases, house price tends to increase, though some variability exists due to other influencing factors









9.FEATURE ENGINEERING:

New Feature Creation:

Extracted year, month, and day from the date column using pd.to_datetime().

Purpose: To capture time-based patterns in housing trends.

Feature Selection:

Selected the most relevant features: bedrooms, bathrooms, sqft_living, yr_built

Reason: These features showed measurable correlation with price and were less likely to introduce noise.

Feature Transformation:

Applied Standard Scaling using StandardScaler to normalize numerical data.

10.MODEL BUILDING:

Linear Regression (Baseline Model):

Chosen for its simplicity and interpretability.Works well when the relationship between variables is linear.

Why Linear Regression Was Chosen:

The dataset had mostly numeric features with a nearly linear relationship between predictors and the target (price).

```
[8] features=['bedrooms','bathrooms','sqft_living','yr_built']
    x=df[features]
    y=df['price']
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
    scaler = StandardScaler()
    x_train_scaled = scaler.fit_transform(x_train)
    x_test_scaled = scaler.transform(x_test)
    model = LinearRegression()
    model.fit(x_train_scaled, y_train)
    y_pred = model.predict(x_test_scaled)
```

11.MODEL EVALUATION:

Mean Squared Error (MSE):

Measures average squared difference between actual and predicted values.

Mean Absolute Error (MAE):

Measures the average magnitude of errors without considering their direction.

R2 Score :

Measures how well the model explains variance in the data.

```
[9] print("\nModel Evaluation:")
    print("MSE:",mean_squared_error(y_test,y_pred))
    print("R2 Score:", r2_score(y_test, y_pred))
    print("MAE:", mean_absolute_error(y_test,y_pred))
```

```
Model Evaluation:
MSE: 807464431127.1014
R2 Score: 0.04137569530157792
MAE: 208682.34920298946
```

12.DEPLOYMENT:

Platform used: vercel

Tools:

- ✓ vercel
- ✓ joblib
- ✓ python

File deployed:

- ✓ **App.py-model logic**
- ✓ **Model.pkl-trained linear regression model**
- ✓ **Scaler.pkl- saved scaler for preprocessing inputs**

Public link: app link (CLICK HERE)

Sample output:

13.SOURCE CODE:

```
#IMPORT LIBRARIES

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

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean_squared_error,
r2_score, mean_absolute_error

#IMPORT DATASET

df=pd.read_csv("/content/data.csv")

df.head()

#NEW FEATURES

df['date'] = pd.to_datetime(df['date']) df['year'] =
df['date'].dt.yeardf['month']=df['date'].dt.month df['day']
= df['date'].dt.day

df.info()

#REMOVE DUPLICATES

print("duplicated rowes before:",df.duplicated().sum())
df.drop_duplicates(inplace=True)
```

```
print("duplicated rowes after:",df.duplicated().sum())
#HANDLE MISSING VALUE
print("\nmissing values:\n",df.isnull().sum())
#REMOVE COLUMN
df.columns
df=df.drop(['date','sqft_lot','floors','waterfront','view','condit
ion','y r_renovated','street','city','country'],axis=1)
#FEATURES ENGINEERING
features=['bedrooms','bathrooms','sqft_living','yr_built']
x=df[features] y=df['price'] df [features + ['price']].
#HISTOGRAM
hist(bins=30,figsize=(12,8)) plt.suptitle("Feature
Distributions", fontsize=16) plt.tight_layout() plt.show()
plt.figure(figsize=(8,6))
#CORRELATION
sns.heatmap(df [features +['price']].corr(),
annot=True,cmap='coolwarm') plt.title("Feature
Correlation with Price") plt.show()
#BEDROOM VS PRICE
import seaborn as sns sns.scatterplot(data=df,
x='bedrooms',y='price', alpha=0.5) plt.title('bedrooms vs
Price') plt.xlabel('bedrooms') plt.ylabel('Price') plt.show()
#BATHROOM VS PRICE
```

```
sns.scatterplot(data=df, x='bathrooms',y='price',
alpha=0.5) plt.title('bathrooms vs Price')
plt.xlabel('bathrooms') plt.ylabel('Price') plt.show()

#YRBUILT VS PRICE

sns.scatterplot(data=df, x='yr_built',y='price', alpha=0.5)
plt.title('yr_built vs Price') plt.xlabel('yr_built')
plt.ylabel('Price')

#SQFTLIVING VS PRICE

plt.show() sns.scatterplot(data=df, x='sqft_living',y='price',
alpha=0.5) plt.title('sqft_living vs Price')
plt.xlabel('sqft_living') plt.ylabel('Price') plt.show()

#MODELING

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25, random_state=42) scaler =
StandardScaler() x_train_scaled =
scaler.fit_transform(x_train) x_test_scaled =
scaler.transform(x_test) model = LinearRegression()
model.fit(x_train_scaled, y_train) y_pred =
model.predict(x_test_scaled)


#EVALUATION

print("\nModel Evaluation:")
print("MSE:",mean_squared_error(y_test,y_pred))
```

```
print("R2 Score:", r2_score(y_test, y_pred)) print("MAE:",
mean_absolute_error(y_test,y_pred))

#VISUALISATION

plt.figure(figsize=(6,6)) plt.scatter(y_test, y_pred,
alpha=0.5) plt.plot([y_test.min(),
y_test.max()],[y_test.min(), y_test.max()], 'r- ')
plt.xlabel("Actual Price") plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted House Prices") plt.show()
df[['bedrooms','bathrooms','sqft_living','yr_built']].

#FOR INDENTIFY RANGE OF INPUT

describe()

#GET INPUT FROM USER AND PRDICT

bedrooms = int(input("Enter number of bedrooms: "))
bathrooms = float(input("Enter number of bathrooms: "))
sqft_living = int(input("Enter square footage (sqft_living):
")) yr_built = int(input("Enter year built: ")) new_house =
pd.DataFrame([{ 'bedrooms': bedrooms, 'bathrooms':
bathrooms, 'sqft_living': sqft_living, 'yr_built': yr_built }])
new_house_scaled = scaler.transform(new_house)
predicted_price = model.predict(new_house_scaled)
print("Predicted House Price: $",
round(predicted_price[0],2))

#WWW.VERCEL.COM(app.py)

from django.shortcuts import render
```

```python
from django.http import HttpResponse,JsonResponse
import joblib
import os
import pandas as pd
import json
from django.views.decorators.csrf import csrf_exempt
#LOAD THE MODEL
model = joblib.load(os.path.join('models', 'model.pkl'))
scaler = joblib.load(os.path.join('models', 'scaler.pkl'))
# Create your views here.
@csrf_exempt
def predict_price(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            bedrooms = data['bedrooms']
            bathrooms = data['bathrooms']
            sqft_living = data['sqft_living']
            yr_built = data['yr_built']
             input_df = pd.DataFrame([{
              'bedrooms': bedrooms,
```

```
            'bathrooms': bathrooms,
            'sqft_living': sqft_living,
            'yr_built': yr_built
        }])
input_scaled = scaler.transform(input_df)
predicted_price = model.predict(input_scaled)[0]
return JsonResponse({'predicted_price':
round(predicted_price, 2)})
except Exception as e:
        return JsonResponse({'error': str(e)}, status=400)
```

14.FUTURE SCOPE:

Interactive Map-Based UI:

Enhance the user experience by deploying an app that includes an interactive map where users can click on a region and get average or predicted property prices in that area.

API Integration for Real-Time Data:

Connect the app to real estate APIs (like Zillow or Realtor) to fetch live listings and continuously update the dataset for dynamic predictions.

11.Team Members and Contributions

NAME AND ROLE/RESPONSIBILITY

- ✓ NAVYAA SHARMA
- ✓ [2303717720522059]

❖ Handle missing values,datatype conversions, duplicates, and initial cleaning.
❖ Generate new features from the date column.
❖ Ensure the dataset is prepared for analysis and modeling

- ✓ THARUN S
- ✓ [2303717720521055]

❖ Create visualizations(Actual vs Predicted plots).
❖ Summarize insights and model performance visually.
❖ Compile the final project documentation and coordinate presentation.
❖ Ensure code notebooks and charts are clean and explainable for stakeholders.

- ✓ THILIBAN P
- ✓ [2303717720521056]

❖ Perform detailed EDA with visualizations (histograms, boxplots, scatterplots).
❖ Analyze relationships between features and price.

- ✓ SURYAHARI k
- ✓ [2303717720521052]

❖ Choose and implement the regression model(Linear Regression).
❖ Handle data splitting, feature scaling, training, and predictions.
❖ Evaluate model using MSE, MAE, R2score.Recommend improvements based on results