```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_scor
import joblib
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# Function to convert area values to square feet
def convert_to_square_feet(area_value):
    if pd.isnull(area_value):
        return np.nan  # Return NaN if area_value is nu
    # Remove commas and extra spaces
    area_value = area_value.replace(',', '').strip()
    if 'Kanal' in area_value:
        # Convert kanal to square feet (1 kanal = 20 ma
        return float(area_value.replace('Kanal', '')) *
    elif 'Marla' in area_value:
        # Convert marla to square feet (1 marla = 272.2
        return float(area_value.replace('Marla', '')) *
    else:
        # If no unit is specified, assume it's already
        return float(area_value)

# File path to the CSV data
file_path = '/content/drive/MyDrive/zameen-property-dat

# Read the CSV into a DataFrame
df = pd.read_csv(file_path)

# Drop duplicates
df.drop_duplicates(inplace=True)

# Drop rows with missing values in specified columns
df.dropna(subset=['price', 'location', 'city', 'provinc

# Fill missing values with median for 'baths' and 'Unkn
df.fillna({'baths': df['baths'].median(), 'agency': 'Un
```

transformed_property_data.csv

9991 to 10000 of 168446 entries Filter

property_type	price	latitude
2	5500000.0	31.60337
2	17500000.0	31.592657
2	20000000.0	31.395119
2	17000000.0	31.395187
2	3850000.0	31.59071
2	10800000.0	31.395187
2	7000000.0	31.573032
2	45000000.0	31.503922
2	2900000.0	31.600584
2	7000000.0	31.584532

Show 10 per page

1 10 100 900 990

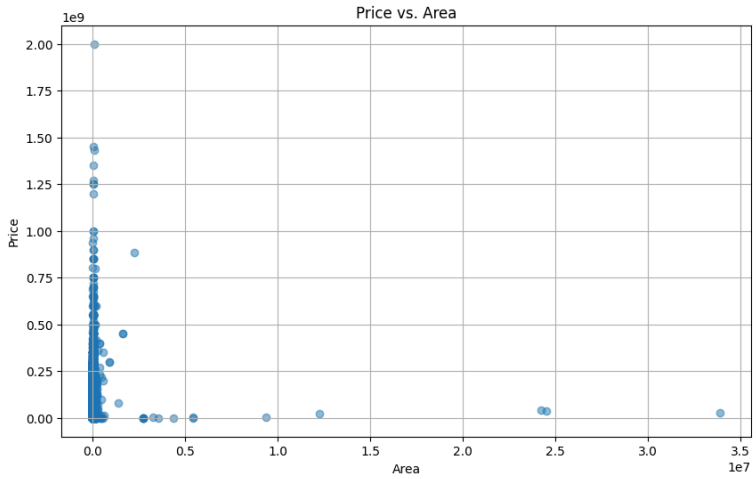
999 **1000** 1001 1010

1100 2000 10000 16000

16800 16840 16845

```
df.fillna({'baths': df['baths'].median(), 'agency': 'Un

# Assign numeric values to property types
property_type_mapping = {
    'Farm House': 0,
    'Flat': 1,
    'House': 2,
    'Lower Portion': 3,
    'Penthouse': 4,
    'Room': 5,
    'Upper Portion': 6
}
df['property_type'] = df['property_type'].map(property_

# Convert data types
df['price'] = df['price'].astype(np.float32)
df['latitude'] = df['latitude'].astype(np.float32)
df['longitude'] = df['longitude'].astype(np.float32)
df['baths'] = df['baths'].astype(np.int8)
df['bedrooms'] = df['bedrooms'].astype(np.int8)

# Extract numerical value from 'area', convert to squar
df['area'] = df['area'].apply(lambda x: convert_to_squa

# Convert 'date_added' to datetime
df['date_added'] = pd.to_datetime(df['date_added'])

# Convert 'date_added' to ordinal
df['date_added'] = df['date_added'].map(pd.Timestamp.to



# Drop unnecessary columns
unnecessary_columns = ['province_name', 'purpose', 'age
df.drop(columns=unnecessary_columns, inplace=True)


# Output path for the transformed data
output_path_csv = '/content/drive/MyDrive/transformed_p

# Save the transformed data to a CSV file without compr
df.to_csv(output_path_csv, index=False)
print(f"Transformed data saved to {output_path_csv}")

# Visualize the relationship between price and area
plt.figure(figsize=(10, 6))
plt.scatter(df['area'], df['price'], alpha=0.5)
plt.title('Price vs. Area')
plt.xlabel('Area')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```

```
Transformed data saved to /content/drive/MyDrive/t
```



```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_scc
import joblib

# Load the transformed property data
transformed_data_path = '/content/drive/MyDrive/transf
df = pd.read_csv(transformed_data_path)

# Define features (X) and target variable (y)
X = df.drop(columns=['price'])
y = df['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,

# Initialize the decision tree regressor
model = DecisionTreeRegressor(random_state=42)

# Train the model
model.fit(X_train, y_train)

# Predict on the testing set
y_pred = model.predict(X_test)

# Calculate root mean squared error (RMSE)
rmse = mean_squared_error(y_test, y_pred, squared=Fals
print("Root Mean Squared Error (RMSE):", rmse)

# Calculate R-squared (R2) score
r2 = r2_score(y_test, y_pred)
print("R-squared (R2) score:", r2)

# Save the trained model
model_path = '/content/decision_tree_model.joblib'
joblib.dump(model, model_path)
print("Trained model saved to:", model_path)
```

```
Root Mean Squared Error (RMSE): 25011860.84651548
R-squared (R2) score: 0.46028569187817137
Trained model saved to: /content/decision_tree_moc
```

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from datetime import datetime
import joblib

# Load the trained model
model_path = '/content/decision_tree_model.joblib'  # S
model = joblib.load(model_path)

# Function to convert area values to square feet
```

```
def convert_to_square_feet(area_value):
    if 'kanal' in area_value:
        return float(area_value.replace('kanal', '')) *
    elif 'marla' in area_value:
        return float(area_value.replace('marla', '')) *
    else:
        return float(area_value)

# Function to preprocess user input
def preprocess_input(user_input):
    user_input['latitude'] = float(user_input['latitude
    user_input['longitude'] = float(user_input['longitu
    user_input['baths'] = int(user_input['baths'])
    user_input['area'] = convert_to_square_feet(user_in
    user_input['bedrooms'] = int(user_input['bedrooms']
    user_input['date_added'] = datetime.strptime(user_i
    return user_input

# Define static input values
static_input = {
    'property_type': 2,
    'latitude': 31.60337,
    'longitude': 74.33013,
    'baths': 1,
    'area': '4 marla',
    'bedrooms': 1,
    'date_added': '2019-03-07'
}

# Preprocess static input
preprocessed_input = preprocess_input(static_input)

# Convert to DataFrame with correct feature order
input_df = pd.DataFrame([preprocessed_input])

# Ensure columns are in the correct order
input_df = input_df[['property_type', 'latitude', 'long
```