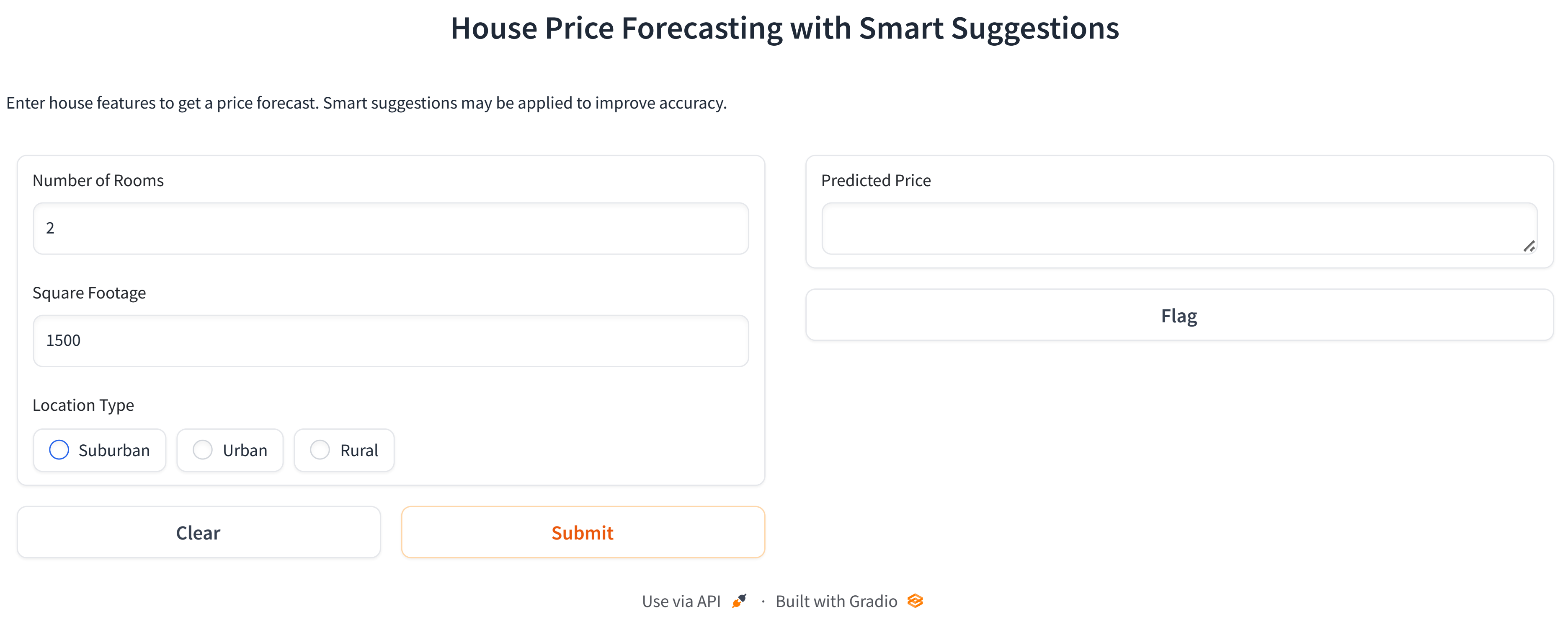
```python
# Install Gradio
!pip install gradio==3.50.0

# Import necessary libraries
import gradio as gr
import numpy as np  # Example import for numerical operations
# Import your specific model loading libraries (e.g., tensorflow, sklearn, pytorch)
# from your_model_library import load_model # Example

# --- Placeholder: Load your pre-trained model ---
# Replace this with the code to load your actual model
# Example (using a dummy function):
def load_my_model():
    # In a real scenario, you would load your trained model here.
    # For example, if you saved a scikit-learn model with pickle:
    # import pickle
    # with open('your_model.pkl', 'rb') as f:
    #     model = pickle.load(f)
    # return model
    print("Loading dummy model...")
    return "dummy_model" # Return a placeholder

# Load the model once when the application starts
model = load_my_model()

# --- Placeholder: Define your features ---
# Replace this with the actual feature names your model expects
feature_names = ['number_of_rooms', 'square_footage', 'location_type']

# --- Placeholder: Implement your smart suggestion techniques and prediction logic ---
def predict_house_price(*input_features):
    """
    This function takes input features, applies smart suggestions,
    makes a prediction, and returns the result.

    Args:
        *input_features: Variable number of arguments corresponding to the input fields.

    Returns:
        str: The predicted house price, potentially with suggestion information.
    """
    # Ensure the number of input features matches the expected number
    if len(input_features) != len(feature_names):
        return "Error: Incorrect number of input features provided."

    # Convert input features to a format your model can use (e.g., a numpy array or pandas DataFrame)
    # Using numpy array as an example:
    input_data = np.array([input_features])
```

```
        # --- Apply your smart suggestion techniques here ---
        # This is where you'll implement your logic to analyze inputs and suggest changes.
        # Example: Suggest increasing the number of rooms if it's very low
        suggestion_applied = False
        # Assuming 'number_of_rooms' is the first feature (index 0)
        if input_data[0, 0] < 3:
            input_data[0, 0] = 3  # Suggest setting number of rooms to at least 3
            suggestion_applied = True

        # --- Make a prediction using your loaded model ---
        # Replace this with your actual model prediction code.
        # Example (using the dummy model placeholder):
        # prediction = model.predict(input_data)[0]
        # Since we have a dummy model, let's return a dummy prediction
        prediction = 300000 + (input_data[0, 0] * 50000) + (input_data[0, 1] * 100) # Simple dummy calculation

        # Format the output
        output_text = f"Predicted Price: ${prediction:.2f}"
        if suggestion_applied:
           output_text += "\n(Suggestion: Increased number of rooms to improve forecast)"

        return output_text

    # --- Define Gradio Interface Inputs ---
    # Replace these with the appropriate input components for your features.
    # The order of these components should match the order of features expected by your predict_house_price function.
    input_components = [
        gr.Number(label="Number of Rooms", value=2),  # Example: Initial value
        gr.Number(label="Square Footage", value=1500),
        gr.Radio(label="Location Type", choices=["Suburban", "Urban", "Rural"], value="Suburban")
    ]

    # --- Define Gradio Interface Output ---
    output_component = gr.Textbox(label="Predicted Price")

    # --- Create and Launch the Gradio Interface ---
    interface = gr.Interface(fn=predict_house_price,
                            inputs=input_components,
                            outputs=output_component,
                            title="House Price Forecasting with Smart Suggestions",
                            description="Enter house features to get a price forecast. Smart suggestions may be applied to improve accuracy.")

    # Launch the interface
    interface.launch()
```

```
Requirement already satisfied: gradio==3.50.0 in /usr/local/lib/python3.11/dist-packages (3.50.0)
Requirement already satisfied: aiofiles<24.0,>=22.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (23.2.1)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (5.5.0)
Requirement already satisfied: fastapi in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.115.12)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.5.0)
Requirement already satisfied: gradio-client==0.6.1 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.6.1)
Requirement already satisfied: httpx in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.31.1)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (6.5.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (3.1.6)
Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (2.1.5)
Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (3.10.0)
Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (1.26.4)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (2.2.2)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (10.4.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (2.11.4)
Requirement already satisfied: pydub in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.25.1)
Requirement already satisfied: python-multipart in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (6.0.2)
Requirement already satisfied: requests~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (2.32.3)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (2.10.0)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (4.13.2)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (0.34.2)
Requirement already satisfied: websockets<12.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.50.0) (11.0.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==0.6.1->gradio==3.50.0) (2025.3.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-packages (from altair<6.0,>=4.2.0->gradio==3.50.0) (4.23.0)
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from altair<6.0,>=4.2.0->gradio==3.50.0) (1.38.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.50.0) (3.18.0)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.50.0) (4.67.1)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.50.0) (1.1.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib~=3.0->gradio==3.50.0) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio==3.50.0) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio==3.50.0) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio==3.50.0) (
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio==3.50.0) (2
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio==3.50.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests~=2.0->gradio==3.50.0) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests~=2.0->gradio==3.50.0) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests~=2.0->gradio==3.50.0) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests~=2.0->gradio==3.50.0) (2025.4.26)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.11/dist-packages (from uvicorn>=0.14.0->gradio==3.50.0) (8.1.8)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.11/dist-packages (from uvicorn>=0.14.0->gradio==3.50.0) (0.16.0)
Requirement already satisfied: starlette<0.47.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from fastapi->gradio==3.50.0) (0.46.2)
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.50.0) (4.9.0)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.50.0) (1.0.9)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50.0) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50.0) (202
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50.0) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50.0) (0.24.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib~=3.0->gradio==3.50.0) (1.17.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio->httpx->gradio==3.50.0) (1.3.1)
Loading dummy model...
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
IMPORTANT: You are using gradio version 3.50.0, however version 4.44.1 is available, please upgrade.
--------
Running on public URL: https://c909ca5d700767e589.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```



## House Price Forecasting with Smart Suggestions

Enter house features to get a price forecast. Smart suggestions may be applied to improve accuracy.

Number of Rooms: 2

Square Footage: 1500

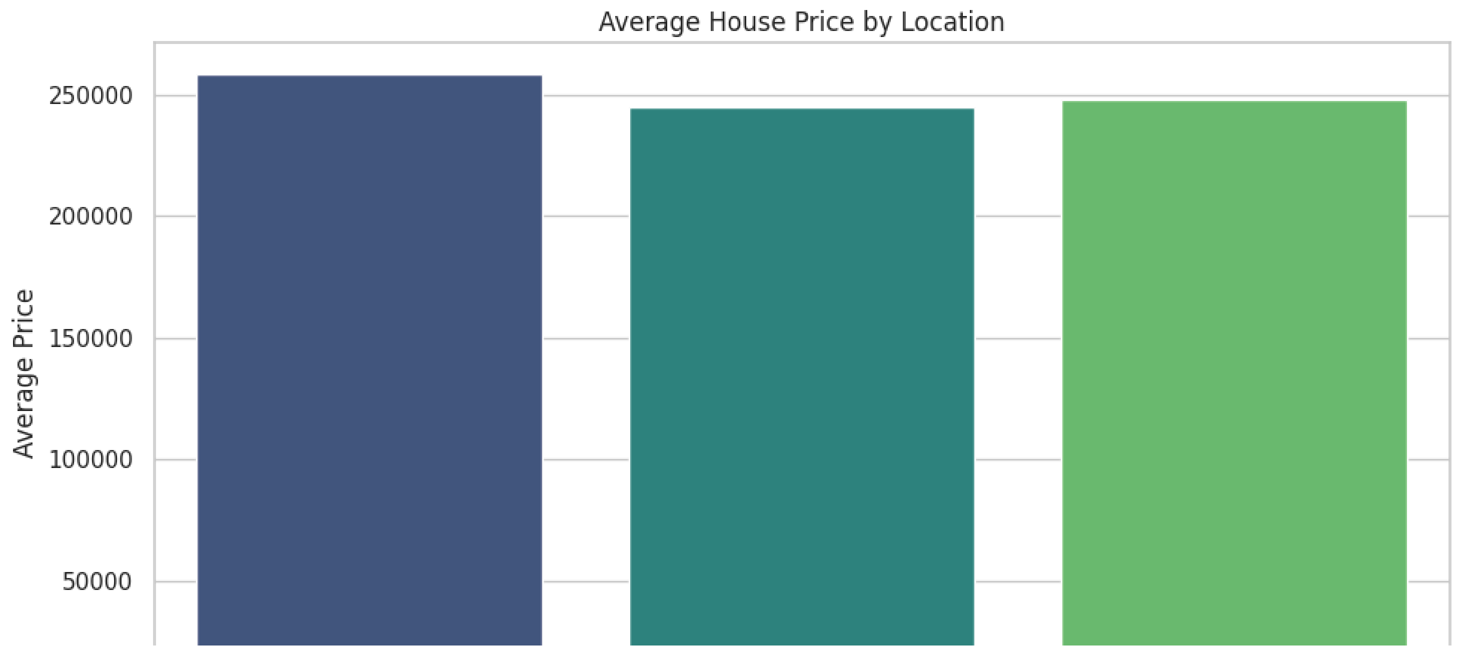
Location Type: Suburban Urban Rural

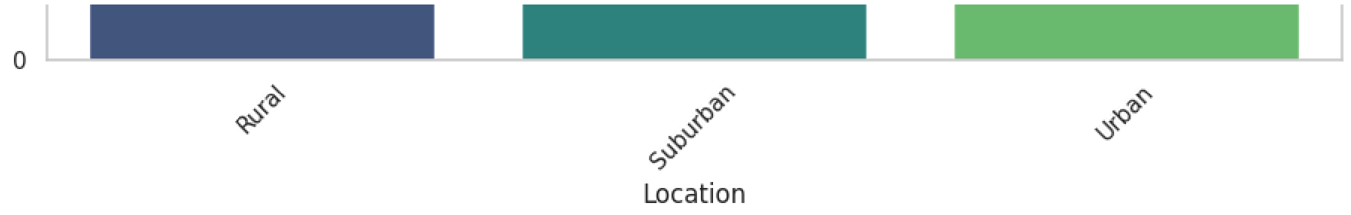
Predicted Price:

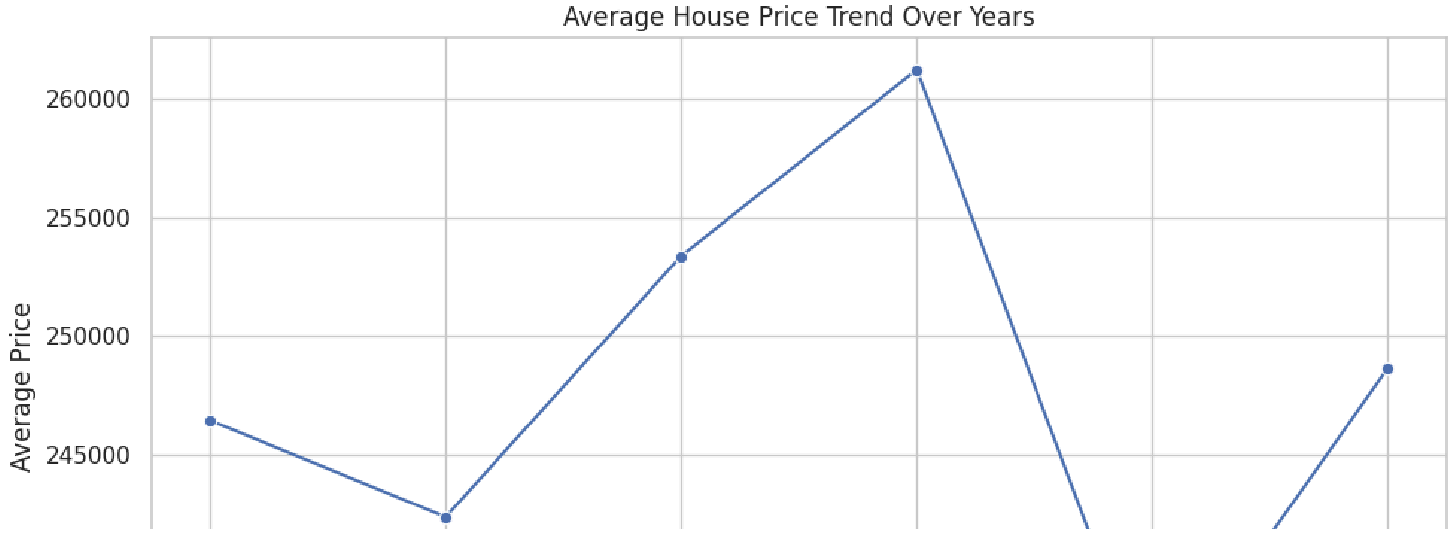
```
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load the dataset
file_path = 'large_house_price_forecasting_dataset.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Initial Data:")
print(data.head())

# Data Cleaning
# Fill missing values
data['Size_sqft'] = data['Size_sqft'].fillna(data['Size_sqft'].mean())
data['Bedrooms'] = data['Bedrooms'].fillna(data['Bedrooms'].mode()[0])
data['Bathrooms'] = data['Bathrooms'].fillna(data['Bathrooms'].mode()[0])
data['Garage'] = data['Garage'].fillna('No')  # Assuming 'No' if missing
data['Price'] = data['Price'].fillna(data['Price'].mean())

# Convert categorical variables to numerical
data['Garage'] = data['Garage'].map({'Yes': 1, 'No': 0})

# Feature Selection
features = ['Location', 'Size_sqft', 'Bedrooms', 'Bathrooms', 'Garage', 'Year_Built']
X = pd.get_dummies(data[features], drop_first=True)  # One-hot encoding for categorical variables
y = data['Price']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Model Evaluation
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error: {rmse:.2f}")

# Feature Importance
importances = model.feature_importances
```

```
    importances = model.feature_importances_
    feature_importance = pd.Series(importances, index=X.columns).sort_values(ascending=False)
    print("\nFeature Importance:")
    print(feature_importance)

    # Save the model (optional)
    joblib.dump(model, 'house_price_model.pkl')

    # Set the aesthetics for the plots
    sns.set(style="whitegrid")

    # Bar Graph: Average Price by Location
    plt.figure(figsize=(10, 6))
    avg_price_by_location = data.groupby('Location')['Price'].mean().reset_index()
    sns.barplot(x='Location', y='Price', data=avg_price_by_location, palette='viridis')
    plt.title('Average House Price by Location')
    plt.xlabel('Location')
    plt.ylabel('Average Price')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Line Graph: Price Trend Over Years
    plt.figure(figsize=(10, 6))
    price_trend = data.groupby('Year_Built')['Price'].mean().reset_index()
    sns.lineplot(x='Year_Built', y='Price', data=price_trend, marker='o')
    plt.title('Average House Price Trend Over Years')
    plt.xlabel('Year Built')
    plt.ylabel('Average Price')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

```
Initial Data:
   House_ID  Location  Size_sqft  Bedrooms  Bathrooms  Year_Built Garage  \
0         1     Urban        NaN       2.0          1      2015.0     No
1         2     Rural     2000.0       NaN          1      2000.0    NaN
2         3     Rural        NaN       2.0          2      2020.0    Yes
3         4  Suburban     1500.0       3.0          3      2005.0     No
4         5  Suburban     1800.0       1.0          1      2005.0    Yes

      Price
0  400000.0
1  250000.0
2  100000.0
3       NaN
4  300000.0
Root Mean Squared Error: 105028.04

Feature Importance:
Year_Built           0.245902
Size_sqft            0.243393
Bedrooms             0.183612
Bathrooms            0.128606
Garage               0.081404
Location_Suburban    0.059130
Location_Urban       0.057953
dtype: float64
<ipython-input-8-a510747b30be>:64: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Location', y='Price', data=avg_price_by_location, palette='viridis')
```




0
Rural
Suburban
Urban
Location


Average House Price Trend Over Years
260000
255000
250000
245000
Average Price