Predicting House Prices Using Image Detection
Milestone 2

Kaitlyn Chen, Ryan Guo, Jonathan Wang
GenAI Gurus

Data preparation

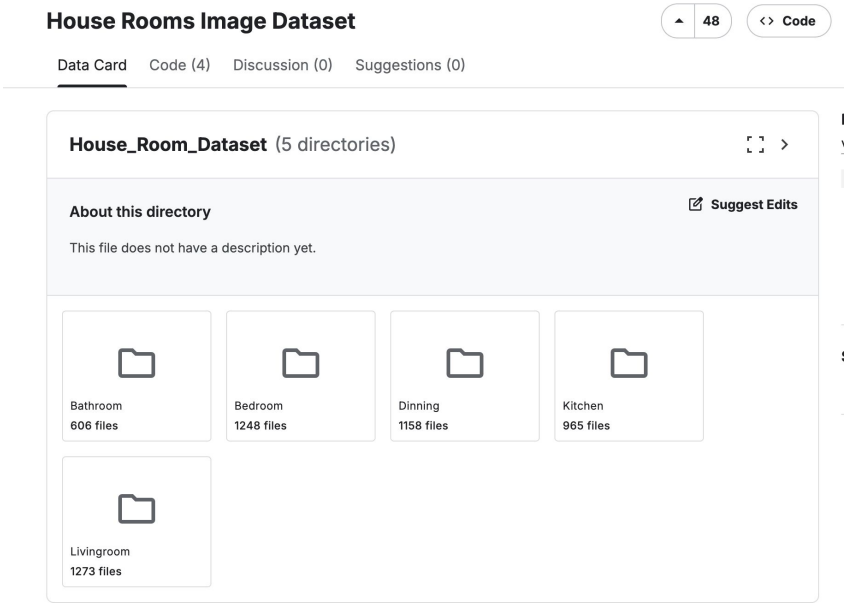
Robin Reni Rooms Images Kaggle dataset (No price)

```python
# Load images from dataset subfolder
X_images, Y = load_images_from_folder(dataset_subfolder, num_samples=5250)
print(f"Loaded {X_images.shape[0]} images from the dataset.")

# Split into train and test sets
X_train_images, X_test_images, Y_train, Y_test = train_test_split(
    X_images, Y, test_size=0.2, random_state=42
)

# Save the preprocessed data
os.makedirs("preprocessed_data", exist_ok=True)
np.save("preprocessed_data/X_train_images.npy", X_train_images)
np.save("preprocessed_data/X_test_images.npy", X_test_images)
np.save("preprocessed_data/Y_train.npy", Y_train)
np.save("preprocessed_data/Y_test.npy", Y_test)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/robinreni/house-rooms-image-dataset/
Extracting /content/datasets/House_Room_Dataset/house-rooms-image-dataset.zip to /content/d
Dataset extracted to /content/datasets/House_Room_Dataset
Contents of dataset folder: ['house-rooms-image-dataset.zip', 'House_Room_Dataset']
Contents of dataset subfolder: ['Kitchen', 'Livingroom', 'Dinning', 'Bedroom', 'Bathroom']
Class names: ['Bathroom', 'Bedroom', 'Dinning', 'Kitchen', 'Livingroom']
Loading images from class: Bathroom
100%|██████████| 606/606 [00:00<00:00, 1007.63it/s]
Loading images from class: Bedroom
100%|██████████| 1248/1248 [00:01<00:00, 939.66it/s]
Loading images from class: Dinning
100%|██████████| 1158/1158 [00:01<00:00, 874.77it/s]
Loading images from class: Kitchen
100%|██████████| 965/965 [00:01<00:00, 946.74it/s]
Loading images from class: Livingroom
100%|██████████| 1273/1273 [00:01<00:00, 920.32it/s]
Loaded 5250 images from the dataset.
```



Data preparation

Mikhail Ma House Rooms & Streets Kaggle dataset (No Price)

- 5000 image limit for RAM usage

```
# Load images, set limit to limit RAM usage
X_images, Y = load_images_from_folder(dataset_subfolder, street_data_limit=5000)
print(f"Loaded {X_images.shape[0]} images from the dataset.")

# Split into train and test sets
X_train_images, X_test_images, Y_train, Y_test = train_test_split(
    X_images, Y, test_size=0.2, random_state=42
)

# Save the preprocessed data
os.makedirs("preprocessed_data", exist_ok=True)
np.save("preprocessed_data/X_train_images.npy", X_train_images)
np.save("preprocessed_data/X_test_images.npy", X_test_images)
np.save("preprocessed_data/Y_train.npy", Y_train)
np.save("preprocessed_data/Y_test.npy", Y_test)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/mikhailma/house-rooms-stre
Extracting /content/datasets/House_Rooms_Streets_Dataset/house-rooms-streets-imag
Dataset extracted to /content/datasets/House_Rooms_Streets_Dataset
Contents of dataset folder: ['kaggle_room_street_data', 'house-rooms-streets-imag
Contents of dataset subfolder: ['street_data', 'house_data']
Class names: ['house_data', 'street_data']
Loading images from class: house_data
100%|██████████| 5249/5249 [00:04<00:00, 1120.94it/s]
Loading images from class: street_data
 25%|██▍       | 5000/19658 [00:03<00:10, 1351.97it/s]
Reached street_data limit of 5000 images.
Loaded 10249 images from the dataset.
```

GitHub dataset (No price)

- Previous solution contains 2140 images, 4 images for each house
- images for bedroom, bathroom, kitchen and a frontal image of the house.

```
            # Resize and normalize image
            img = cv2.resize(img, IMAGE_SIZE)
            img = img_to_array(img) / 255.0  # Normalize pixel values to [0, 1]
            images.append(img)
            filenames.append(filename)
        else:
            print(f"Skipping non-image file: {filename}")

    return np.array(images), filenames

# Load all images
X_images, image_filenames = load_images_from_folder(drive_path)

# Save data
output_path = os.path.join(drive_path, "X_images.npy")
np.save(output_path, X_images)

# Print confirmation
print(f"Images processed: {len(X_images)}")
print(f"Processed images saved to: {output_path}")
```

Baseline Model

Implement a simple convolutional neural network (CNN) as a baseline for comparison

- **Input Layer**: Images of houses resized to a consistent input shape(128x128)
- **Conv Layer 1:** Convolution + MaxPooling (32 filters, 3x3 kernel, 2x2 pool)
- **Conv Layer 2:** Convolution + MaxPooling (64 filters, 3x3 kernel, 2x2 pool)
- **Flatten Layer:** Flatten the output of convolutional layers
- **Dense Layer 1:** Fully connected layer (e.g., 128 units, ReLU activation)
- **Output Layer:** Single node with no activation function (for regression). Outputs a single value using linear activation.
- Or use ResNet-50

ResNet-50

- 50 layers deep
- Residual connections (skip connections) to ease training of deep networks
- Building block: Bottleneck architecture (Conv1x1 → Conv3x3 → Conv1x1)
- Performs well on image classification tasks (e.g., ImageNet)
- Residual learning: Learns the difference (residual) from input to output
- Used in many real-world applications: object detection, segmentation, etc.

Model training

- Transfer learning from base model
- Use information like # of rooms, size, etc for multimodal deep learning architecture
- Augment data (flip image, zoom, random rotation)
- Train the model using model.fit() with callbacks like EarlyStopping and ModelCheckpoint
- Include visualization of training curves (loss vs. epochs) for analysis
- Final output layer should be a single unit with no activation function

Evaluation on test data

- Load the best-performing model weights (e.g., from ModelCheckpoint).
- Split dataset into training, validation, and testing sets
- Evaluate the model on a separate test set not used during training/validation
- Report final evaluation metrics such as MSE, MAE, and R^2 score
- Visualize predicted vs. actual house prices
- Error analysis (e.g., inspecting cases with large prediction errors)
- Compare to baseline model

Questions?

(Feedback)

Option 1: Run pretraining model on data (resnet-50) and then add a classifier at the end

Option 2: Predict prices pm their pwm and then compare to the actual prices

Option 3: get pictures that r similar? And then compare their prices if theyre similar or smth like this idk

Train a model based on the dataset containing the prediction sales price, use the convolution layer from this model and compare layers for a separate model training the data without images.