

Real Estate Automated Valuation Model

Jeremy Wizenfeld, Aaron Poulad, and Yair Levie



Problem:

BUSINESS | EARNINGS

Zillow's Shuttered Home-Flipping Business Lost \$881 Million in 2021

Real-estate company says in a letter to shareholders that it is targeting revenue of \$5 billion by 2025

By *Will Parker* [Follow](#)

Updated Feb. 10, 2022 6:24 pm ET

- Given the rapid pace of real estate markets, manually evaluating numerous properties can be time-consuming and inefficient.
- iBuying- a company makes an offer to buy a home without going out to visit it
- Successful in generic houses of similar quality and size, but less effective for high-end or low-quality homes where construction quality significantly impacts pricing.
- [Zillow's Shuttered Home-Flipping Business Lost \\$881 Million in 2021](#)

Solution:

- Examine property images to refine price prediction
 - Allows us to account for building **quality**, not seen in standard property data
- Convolutional Neural Network (**CNN**) approach
 - Use a CNN to categorize room types and score key **visual features** such as property condition, style, and quality from specific rooms
- Score refinement
 - Consolidate multiple image scores into an **overall property quality score**, and combine with structured data to attain a more accurate property value estimate

Project Timeline

Q1:

**August 28th-
September 25th**

1. Researched CNNs
2. Researched Regression Models
3. Devised a plan for data collection

Q2:

**September 26th-
October 14th**

1. Deployed Label System on GCP to Collect and Clean Data
2. Developed basic Linear Regression models

Q3:

**October 15th-
November 20th**

1. Created CNN for image classification and scoring
2. Linear Regression implemented the image scores
3. Pipeline to pull from Zillow API

Q4:

**November 21st-
December 11th**

1. Merged all components and fixed bugs
2. Built a UI to input address and quickly receive prediction

Data Collection: Labeling and Scoring

Keyboard Shortcuts

Common Areas

B - Bathroom
K - Kitchen
L - Living Room
D - Dining Room
R - Bedroom

Extra Rooms

O - Office
C - Closet
A - Attic
G - Garage
M - Basement
P - Pantry
U - Utility/Laundry

Connecting Spaces

H - Hallway
S - Stairs

Outdoor

E - Exterior
T - Patio
Y - Balcony
Q - Deck

X - Other

Space - Skip Image
Tab - Switch to Labeling

Image Labeling System - COM4930

Skip Image



Use keyboard shortcuts or select a label and submit.

Bathroom

Submit Label

Switch to Scoring

Property Scoring Guidelines

Score: 8-9 - Luxury

- Top-tier, fancy finishes
- Perfect condition
- High-tech, modern everything

Score: 6-7 - Above Average

- Good quality
- Nice finishes
- No major issues

Score: 4-5 - Average

- Basic features
- Okay condition
- Nothing special

Score: 2-3 - Below Average

- Outdated
- Needs repairs
- Worn out

Score: 1 - Poor

- Major repairs needed
- Lots of damage
- Old and broken

Keyboard Shortcuts:

- 1-9: Quick score
- Space: Skip Image
- N: Not sure for scoring

Property Quality Rating

Skip Image



Rate the property quality from 1 to 9



CNN

ResNet18-Based Room Classifier (11,181,642 parameters)

- **Task:** Multi-class classification (predict room type among 10 categories).
- **Architecture:**
 - Pretrained ResNet18
 - Final fully connected (FC) layer replaced with `nn.Linear(in_features, num_classes=10)`.
 - Retains strong feature-extraction capacity from ImageNet pretraining.
- **Loss Function:**
 - `nn.CrossEntropyLoss` for multi-class classification.
- **Optimizer & Scheduler:**
 - Adam optimizer with a learning rate of 0.001.
 - `ReduceLROnPlateau` scheduler to adjust learning rate when loss plateaus.

MobileNetV2-Based Score Regressor (2,305,921 parameters)

- **Task:** Regression (predict a continuous “score”).
- **Architecture:**
 - Pretrained MobileNetV2 (smaller version of ImageNet, training was taking too long).
 - Final classifier replaced with a small FC “head”:
 - `nn.Dropout(0.2) > nn.Linear(...)`
`> nn.ReLU()`
`nn.Dropout(0.2) > nn.Linear(..., 1)`
 - Lightweight and efficient, suitable for large-scale or MPS (Metal Performance Shaders) usage.
- **Loss Function:**
 - `nn.MSELoss` for regression tasks.
- **Optimizer:**
 - Adam optimizer with a learning rate of 0.001.

Label CNN Demo

Real-Estate-AVM

test_image.py M X

pic2_17_N_Parkview_Ave.jpg

pic17_17_N_Parkview_Ave.jpg

pic21_17_N_Parkview_Ave.jpg

pic47_17_N_Parkview_Ave.jpg

OPEN EDITORS

test_image.py utils M

pic2_17_N_Parkview_Ave.jpg...

pic17_17_N_Parkview_Ave.jp...

pic21_17_N_Parkview_Ave.jp...

pic47_17_N_Parkview_Ave.jp...

pic51_17_N_Parkview_Ave.jp...

DEBUG CONSOLE

REAL-ESTATE-AVM

data

raw

docs

image_dataset

logs

secrets

templates

index.html

utils

__pycache__

__init__.py

bing_download.py

kaggle_to_csv.py

quality_score_for_csv.py

test_image.py M

venv

bin

include

lib

share

pyvenv.cfg

env

.gitignore

kaggle_labels.csv

main.py

quality_score.py

README.md

requirements.txt

OUTLINE

TIMELINE

```
1 import torch
2 from torchvision import transforms
3 from PIL import Image
4 import logging
5 import sys
6 from room_labeler_cnn import RoomClassifier, room_label_map # Import your model and label map
7
8 # Set up logging
9 logging.basicConfig(level=logging.INFO)
10 logger = logging.getLogger(__name__)
11
12
13 # Define a function to preprocess the input image
14 def preprocess_image(image_path: str):
15     transform = transforms.Compose(
16         [
17             transforms.Resize((224, 224)),
18             transforms.ToTensor(),
19             transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
20         ]
21     )
22
23     try:
24         image = Image.open(image_path).convert("RGB") # Open and convert to RGB
25         return transform(image).unsqueeze(0) # Add batch dimension
26     except Exception as e:
27         logger.error(f"Error processing image {image_path}: {e}")
28         sys.exit(1)
29
30
31 # Load the model
32 def load_model(model_path: str, num_classes: int, device):
33     try:
34         model = RoomClassifier(num_classes=num_classes)
35         model.load_state_dict(torch.load(model_path, map_location=device, weights_only=True))
```

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS

(venv) jeremywizenfeld@dyn-010-144-014-057 ~/Desktop/Real-Estate-AVM main ±

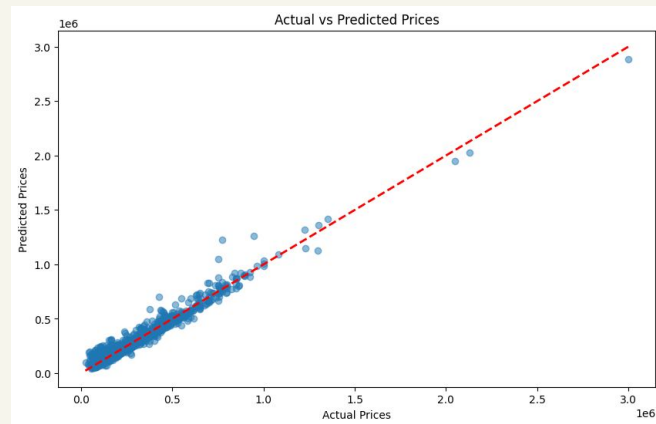
Python

zsh

Ln 65, Col 1 Spaces: 4 UTF-8 LF Python 3.12.4 (venv: venv) Go Live Prettier

Price Prediction Model

- **Linear Regression**
 - Finds linear relationships between features and price
- **Decision Tree**
 - Partitions data into regions based on feature values and averages price within those regions
- **Random Forest**
 - Creates multiple decision trees and averages them, being less sensitive to overfitting
- **Gradient Boosting (XGBoost)**
 - Error correction on top of random forest, and can handle complex relationships and interactions in the data.



Results

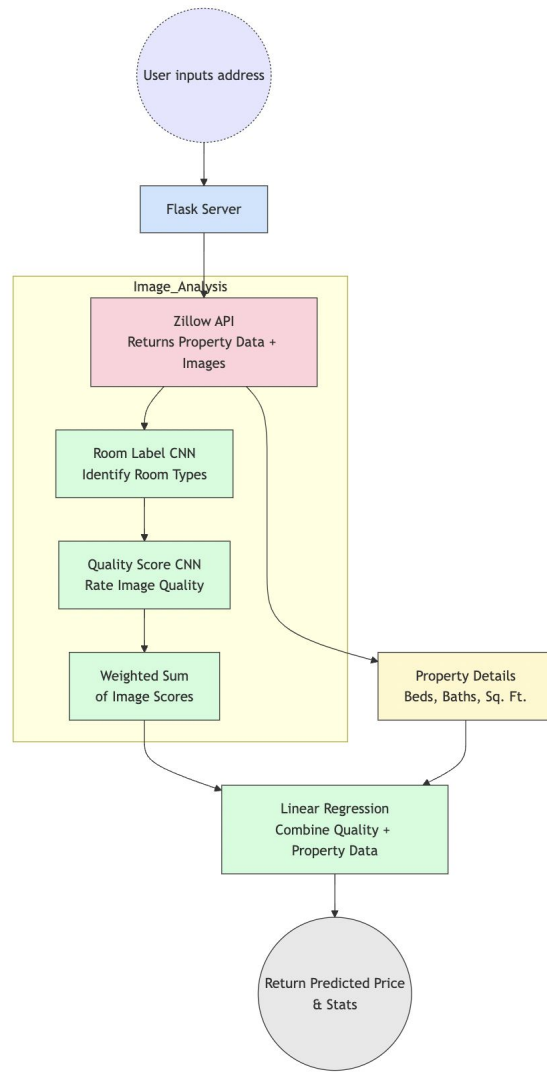
Data incorporated	MAE	RMSE	SMAPE	R ²
Only elementary data	\$50,931	\$82,732	18.73%	86.50%
With Zestimate	\$30,183	\$52,946	12.71%	94.47%
With image scores	\$29,647	\$52,171	12.74%	94.63%

Zestimate accuracy	\$34,910	\$61,141	14.32%	93.28%
--------------------	----------	----------	--------	--------

Data used for all stages: bathrooms, bedrooms, living area, lot size, tax assessed value, yearBuilt
Trained on 5441 properties; tested on 1361.

Putting It All Together

1. **User Input:** The user enters a property address.
2. **Data Retrieval:** A Flask server queries the Zillow API to fetch property details (e.g., beds, baths, square footage) and corresponding image URLs.
3. **Room Classification:** Each image is passed through a *Room Label CNN* to identify the type of room (kitchen, bedroom, etc.).
4. **Image Quality Scoring:** The images are then fed into a *Quality Score CNN*, which assigns a quality rating based on visual and aesthetic criteria.
5. **House Quality Score:** The individual image scores are combined (weighted sum) into an overall “house quality” metric.
6. **Price Prediction:** A *Linear Regression* model uses both the property’s features (e.g., size, location) and the house quality score to estimate the final home value.
7. **Results:** The predicted price, along with relevant statistics, is sent back to the user interface.



Demo

The image shows a code editor with a dark theme. On the left is a sidebar with a file explorer showing a project structure for 'REAL-ESTATE-AVM'. The main editor displays Python code for a Flask application. The code includes a route for the index page, a POST route for fetching property data, and a cache mechanism. A tooltip is visible over the `validate_address` function call. At the bottom, a terminal window shows the command prompt for a virtual environment.

File Explorer (Left Sidebar):

- REAL-ESTATE-AVM
 - __pycache__
 - pytest_cache
 - .vscode
 - cnn_models
 - lm_with_images.sav
 - room_classifier.pth
 - room_scorer.pth
 - data
 - docs
 - image_dataset
 - logs
 - secrets
 - templates
 - index.html
 - utils
 - __pycache__
 - __init__.py
 - bing_download.py
 - kaggle_to_csv.py
 - quality_score_for_csv.py
 - test_image.py
 - venv
 - .env
 - .gitignore
 - kaggle_labels.csv
 - main.py
 - quality_score.py
 - README.md
 - requirements.txt
 - room_labeled_properties.csv
 - room_labeler_cnn.py
 - room_scoring_cnn.py
 - scores_labeled_properties.csv

Main Editor (Python Code):

```
91 @app.route("/")
92 def index():
93     logger.info("Serving index page")
94     return render_template("index.html")
95
96
97 @app.route("/get-property-data", methods=["POST"])
98 def fetch_property_data():
99     with RequestTimer("/get-property-data"):
100         try:
101             logger.info("Received property data request")
102             data = request.get_json()
103
104             if not data:
105                 logger.warning("No JSON data provided in request")
106                 return jsonify({"error": "No JSON data provided"}), 400
107
108             address = data.get("address")
109             logger.info(f"Processing request for address {address}")
110
111             is_valid, error_message = validate_address(address)
112             if not is_valid:
113                 return jsonify({"error": error_message}), 400
114
115             if address in property_data_cache:
116                 logger.info(f"Cache hit for address: {address}")
117                 return jsonify(
118                     {
119                         "message": "Data retrieved from cache",
120                         "property_data": property_data_cache[address],
121                         "status": "success",
122                     }
123                 )
124
```

Tooltip: (variable) address: Any

Terminal (Bottom):

```
(venv) jeremywizenfeld@Jeremys-MacBook-Pro-2 ~/Desktop/Real-Estate-AVM main ±
```

Status Bar (Bottom): Ln 213, Col 46 Spaces: 4 UTF-8 LF Python 3.12.4 ("venv": venv) @ Go Live Prettier

Future Improvements

Crowdsourced Quality Scores

Gather user-based ratings from many more properties to refine the **Quality Score**. A larger dataset would improve both accuracy and confidence in the predicted price.

Geolocation & Comps

Incorporate geospatial data to identify recently sold homes nearby. **Comparable Sales (Comps)** data can be automatically fed into the model to better reflect local market conditions.

Economic Indicators

Pull in **macroeconomic data** (e.g., interest rates, GDP trends) so the model can adjust dynamically for broader market shifts—a key lesson from Zillow's 2021 iBuying challenges.

Model Fine-Tuning

Experiment with different **weights for room scores** and **hyperparameters** (e.g., learning rates, number of layers) to further optimize accuracy and reliability.

Technologies

