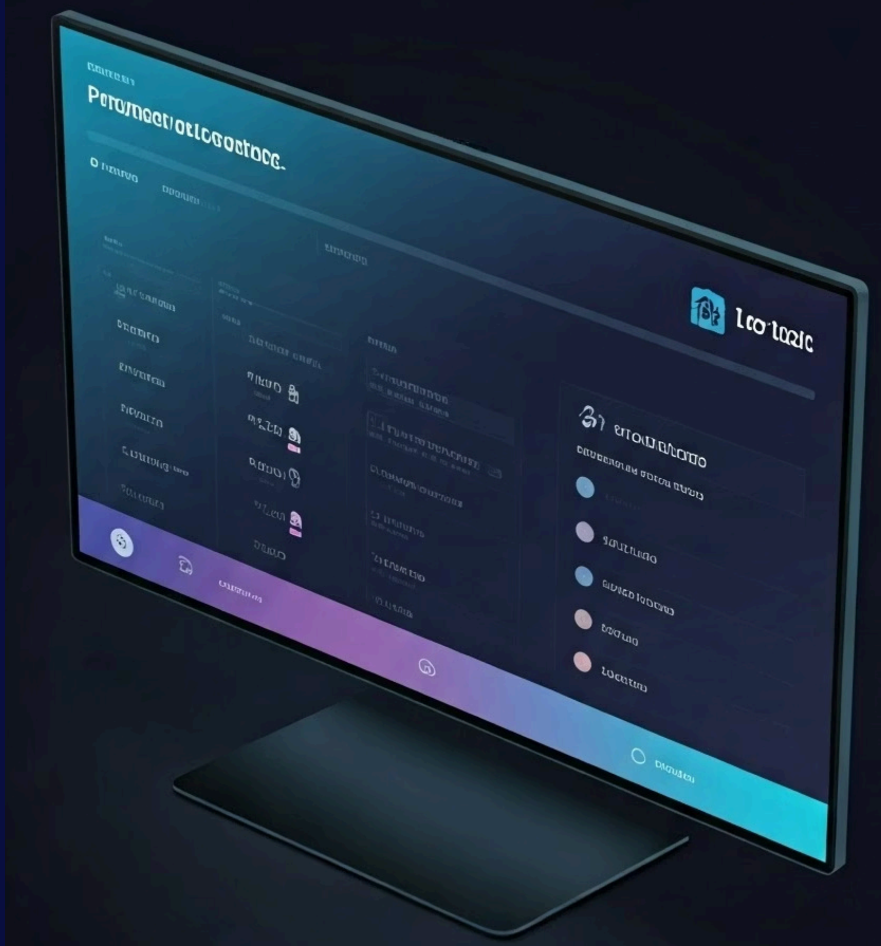
PropComp AI: Intelligent Real Estate Appraisal System

Revolutionizing property appraisal through machine learning and explainable AI. Developed by Misbah Ahmed Nauman, University of Alberta Computer Engineering student.

by Misbah Ahmed Nauman



Real Estate Appraisal Challenges

Manual Comparable Selection

Professionals rely on subjective "comps" without systematic automation or ranking models.

Lack of Transparency

No human-readable reasoning behind property selection decisions for stakeholders.

Local Market Gaps

Generic approaches fail to capture neighborhood-specific similarity patterns and preferences.

```
📁 property-recommendation-system/
│
├── 📁 data/
│   ├── raw/              # Original appraisal data
│   ├── cleaned/          # Standardized data
│   ├── engineered/       # Feature-enhanced data
│   ├── training/         # ML-ready data
│   ├── geocoded-data/    # Location coordinates
│   └── README.md         # Data documentation
│
├── 📁 frontend/          # Streamlit web interface
│   ├── app.py            # Main application
│   └── feedback/         # User feedback storage
│
├── 📁 models/            # Trained ML models
├── 📁 scripts/           # Python processing scripts
├── 📁 outputs/           # Generated explanations
│
├── .gitignore
├── requirements.txt
└── README.md
```

PropComp AI Solution Architecture

1

Data Processing Pipeline

Cleans and geo-processes raw property datasets with automated geocoding capabilities.

2

XGBoost Ranking Model

Trains pairwise ranking model on labeled comparable property pairs for accuracy.

3

Intelligent Explanations

GPT-generated reasoning clarifies selection criteria for each recommended comparable property.

4

Interactive Feedback

Planned refinement loop enables continuous model improvement through user input.

Technical Implementation Stack

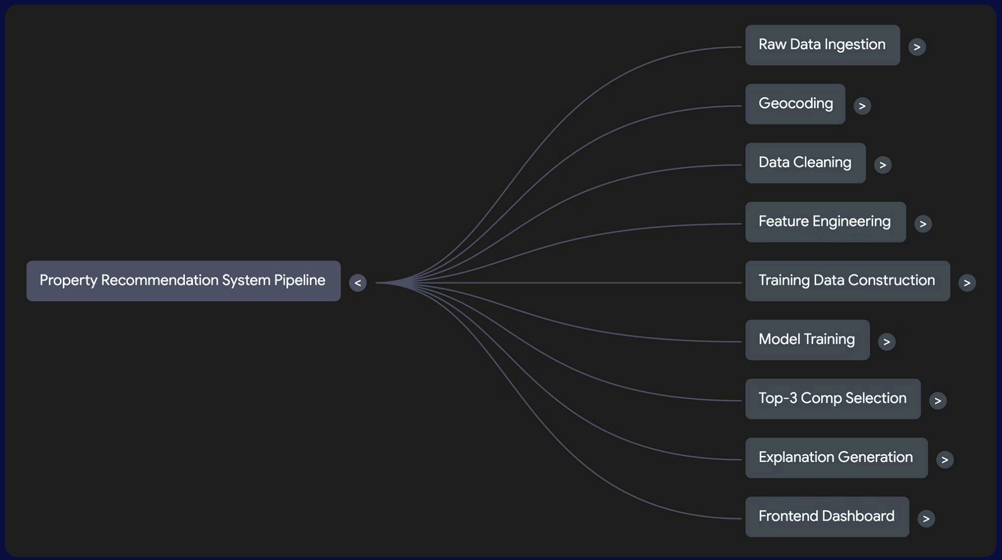
Core Components

- **Data Ingestion & Cleaning**
  – Load raw appraisal JSON/CSV, dedupe & normalize fields
- **Geocoding & Spatial Enrichment**
  – Turn addresses into lat/long, compute neighbourhood proximities
- **Feature Engineering & Similarity Metrics**
  – Compute comparable-property features (GLA, lot size, beds/baths, age, type)
- **XGBoost Pairwise Ranker**
  – Trained in "rank:pairwise" mode to score and order candidates
- **OpenAI Completion API**
  – Turn model outputs into human-readable explanations via GPT
- **Streamlit Dashboard & Feedback Loop**
  – Interactive UI for reviewing comps, collecting Yes/No feedback, and triggering retraining

Data Pipeline

Sequential processing transforms raw appraisal data into structured insights. The pipeline includes geocoding, data cleaning, feature engineering, model training, and explanation generation.

It culminates in ranked property comps, each paired with transparent, AI-generated justifications to support valuation decisions.



Performance Metrics and Demo Results

96.21%

Precision@Top-3

Industry-leading accuracy in comparable property selection

254

Correct Predictions

Out of 264 total comparable property recommendations

3

Best Comparables

Ranked recommendations with GPT-powered explanations

📊 **Model Evaluation Summary**

Evaluation Metric: Top-3 Precision (per appraisal)

Test Data: Held-out appraisal groups

Ranking Model: XGBoost with `rank:pairwise` objective

Metric	Value
Total Top-3 Predictions	264
Correctly Identified Comps	254
False Positives	10
Top-3 Precision	96.21%

False Positives (Predicted Top-3 but Not True Comps):

Order ID	Candidate Address
4758672	177 Bramco Lane
4759024	242 Coville Circle NE
4760630	240 Nelson Street
4761446	871 Crestwood Ave
4762330	239 Kinniburgh Loop
4765948	240 Nelson Street
4765948	16 James St
4772152	36 Hidden Spring Place NW
4775292	6555 Third Line
4776066	161 Everoak Circle SW

```python
import xgboost as xgb
import openai

# --- Feature Engineering ---
def score_comp(s, c):
    return (
        abs(s["gla"] - c["gla"]) / s["gla"] * 0.3 +
        abs(s["lot"] - c["lot"]) / s["lot"] * 0.2 +
        abs(s["bed"] - c["bed"]) * 0.2 +
        abs(s["bath"] - c["bath"]) * 0.2 +
        abs(s["year"] - c["year"]) * 0.1
    )

# --- Model Training ---
def train_model(X, y, group):
    dtrain = xgb.DMatrix(X, label=y)
    dtrain.set_group(group)
    model = xgb.train({"objective": "rank:pairwise"}, dtrain, 50)
    model.save_model("xgb_rank_model.json")

# --- Explanation Generation ---
def explain(subject, comp):
    prompt = f"""Subject: {subject}, Comp: {comp}.
    Why is this comp a good match?"""
    return openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )["choices"][0]["message"]["content"]
```

Engineering Deep Dive

Model Training (train_model.py)

Implements pairwise XGBoost ranking with careful train/test splits to prevent overfitting. Saves optimized model to JSON format.

Feature Engineering (features.py)

Scaled distance computation using weighted similarity metrics. Core scoring based on GLA, lot size, bedrooms, bathrooms, property age.

Explanation Generation

GPT prompting with subject-comparable property differences. Ensures every recommendation includes transparent, defensible reasoning for stakeholders.

Future Development Roadmap

Enhanced Analytics

Integrate price prediction algorithms and SHAP value visualizations for comprehensive model interpretability.

Platform Expansion

Build user authentication system with personalized dashboards and historical comparable property tracking.

AI Model Comparison

Evaluate Claude and Gemini alternatives for explanation generation and multi-page administrative metrics interface.