***The preferred way to test this project is to go to*** [***capstone.gabriellacodes.com***](https://capstone.gabriellacodes.com) ***and interface with it that way.***

**For local deployment, the preferred method is:**

docker-compose --build

*For alternative deployment methods without Docker, please refer to the* [*Quick-start guide*](#a-quick-start-guide) *section below.*

# Rental Investment Area Predictor

## Table of Contents

* [Rental Investment Area Predictor](#rental-investment-area-predictor)
  + [Table of Contents](#table-of-contents)
* [B. Executive Summary](#b-executive-summary)
  + [Table of Contents](#table-of-contents-1)
  + [The decision support problem or opportunity you are solving for](#Xf544e1ec0d634132d35b01fdfb5f102c45fcc9f)
  + [A description of the customers and why this product will fulfill their needs](#X629438d42af85cc9672b66968a8e3d165149ef4)
  + [Existing gaps in the data products you are replacing or modifying (if applicable)](#X6bbde4b1bd8c9ba3e24989884818e7f7cbe9a98)
  + [The data available or the data that needs to be collected to support the data product lifecycle](#X352d3e325a30f228d535761ccdefc86ac15a5d8)
  + [The methodology you use to guide and support the data product design and development](#X087556c4a4f3e9a4681e193c3d6fd1f27f302d3)
  + [Deliverables associated with the design and development of the data product](#X74bc6170769a91eb37b8f92b057172fdcfaf3fe)
  + [The plan for implementation of your data product, including the anticipated outcomes from this development](#Xbaa09f295248505001d227a5e0070d0dff8abf5)
  + [The methods for validating and verifying that the developed data product meets the requirements and, subsequently, the needs of the customers](#Xbc3acf7a24c0f883cd1af10d3fd76b13100dc6b)
  + [The programming environments and any related costs, as well as the human resources that are necessary to execute each phase in the development of the data product](#X0223305f0a00f561feb041dbdfc8f0c8b3dbb7b)
  + [A projected timeline, including milestones, start and end dates, duration for each milestone, dependencies, and resources assigned to each task](#X78e8eea3ea1f4e601a03516b5f539e55e9b5ae7)
* [C. Data Product Design and Development](#c-data-product-design-and-development)
  + [One descriptive method and one nondescriptive (predictive or prescriptive) method](#X1ea619737b2d28ff57e153e24bbb226aaf8be4d)
    - [Descriptive Method](#descriptive-method)
    - [Nondescriptive (Predictive) Method](#nondescriptive-predictive-method)
  + [Collected or available datasets](#collected-or-available-datasets)
  + [Decision support functionality](#decision-support-functionality)
  + [Ability to support featurizing, parsing, cleaning, and wrangling datasets](#X3d492bad7e945a9c3195c31ce6efa7567879ed4)
  + [Methods and algorithms supporting data exploration and preparation](#Xe9d12f6bf0b21149df8341f69cfeaa93d897a06)
  + [Data visualization functionalities for data exploration and inspection](#Xe1fcb290ae8b3e7d87467997cc319612f925452)
  + [Implementation of interactive queries](#implementation-of-interactive-queries)
  + [Implementation of machine-learning methods and algorithms](#X47fb9070f402dd24b3beb433c0eacf9d6482ae5)
  + [Functionalities to evaluate the accuracy of the data product](#X4f42c4f426cce2732b93d67a05fb39c2e23b5b9)
  + [Industry-appropriate security features](#industry-appropriate-security-features)
  + [Tools to monitor and maintain the product](#X23d655480ae97961042213b949a1015df645093)
  + [A user-friendly, functional dashboard that includes three visualization types](#X3b8f9083c5b03976365a1430552e76fc881aa8d)
* [D. Product Documentation](#d-product-documentation)
  + [Table of Contents](#table-of-contents-2)
  + [A business vision or business requirements document](#Xc336957c4c1dcaab535261444b9f2a5590c126f)
  + [Raw and cleaned datasets with the code and executable files used to scrape and clean data (if applicable)](#X35b5925649d9fdfb4a7eaea4a595329bb84683c)
  + [Code used to perform the analysis of the data and construct a descriptive, predictive, or prescriptive data product](#Xce269d50e8c0c54af777b65a07452b28a582742)
    - [Descriptive Analysis](#descriptive-analysis)
    - [Predictive Modeling](#predictive-modeling)
  + [Assessment of the hypotheses for acceptance or rejection](#X6b483292275a374c340aaeb51b39c1d5b34fc32)
  + [Visualizations and elements of effective storytelling](#X02e18e5f81116ff834edd2bb15d15e9ba22f37a)
  + [Assessment of the product’s accuracy](#assessment-of-the-products-accuracy)
    - [Classification Accuracy](#classification-accuracy)
    - [Model Reliability](#model-reliability)
    - [Feature Performance](#feature-performance)
  + [The results from the data product testing, revisions, and optimization](#X528ccc8ed4de2f6405a341d75d5c2307d712a2a)
    - [Model Evolution and Optimization](#model-evolution-and-optimization)
    - [User Testing and Feedback Implementation](#user-testing-and-feedback-implementation)
  + [Source Code and Installation](#source-code-and-installation)
  + [A quick-start guide](#a-quick-start-guide)
    - [⚠️ Important: Database Seeding](#X634aa2e8f9c5f66cacdfd4edb490c56fabc3535)
    - [Database Setup](#database-setup)
    - [Installation Steps](#installation-steps)
      * [Backend Setup](#backend-setup)
      * [Frontend Setup](#frontend-setup)
    - [Running the Application](#running-the-application)
    - [Using the Product](#using-the-product)
      * [Through the Web Interface](#through-the-web-interface)
      * [Through the API](#through-the-api)
    - [Troubleshooting](#troubleshooting)
  + [Core Components](#core-components)
    - [1. Data Preprocessing (data\_preprocessing.py)](#X7c7e0b1292f91c8caa633bc9d9aa65e6281cef6)
    - [2. Model Training (train\_models.py)](#X06f8bcc350fa7e395f1f3a49622752d886eb071)
    - [3. Model Analysis Tools](#X977e8a8182f726cdd6306e3d69ab7c29f3b716f)
      * [a. Prediction Analysis (analyze\_predictions.py)](#Xb54ce8bc0d8a12d706a70548b5e5e03de130441)
      * [b. Model Evaluation (evaluate\_model.py)](#b-model-evaluation-evaluate_modelpy)
      * [c. API Testing (test\_api.py)](#c-api-testing-test_apipy)
  + [Understanding the Results](#understanding-the-results)
    - [Investment Score](#investment-score)
    - [Ranking Score](#ranking-score)

# B. Executive Summary

## Table of Contents

* [The decision support problem or opportunity you are solving for](#Xf544e1ec0d634132d35b01fdfb5f102c45fcc9f)
* [A description of the customers and why this product will fulfill their needs](#X629438d42af85cc9672b66968a8e3d165149ef4)
* [Existing gaps in the data products you are replacing or modifying (if applicable)](#X6bbde4b1bd8c9ba3e24989884818e7f7cbe9a98)
* [The data available or the data that needs to be collected to support the data product lifecycle](#X352d3e325a30f228d535761ccdefc86ac15a5d8)
* [The methodology you use to guide and support the data product design and development](#X087556c4a4f3e9a4681e193c3d6fd1f27f302d3)
* [Deliverables associated with the design and development of the data product](#X74bc6170769a91eb37b8f92b057172fdcfaf3fe)
* [The plan for implementation of your data product, including the anticipated outcomes from this development](#Xbaa09f295248505001d227a5e0070d0dff8abf5)
* [The methods for validating and verifying that the developed data product meets the requirements and, subsequently, the needs of the customers](#Xbc3acf7a24c0f883cd1af10d3fd76b13100dc6b)
* [The programming environments and any related costs, as well as the human resources that are necessary to execute each phase in the development of the data product](#X0223305f0a00f561feb041dbdfc8f0c8b3dbb7b)
* [A projected timeline, including milestones, start and end dates, duration for each milestone, dependencies, and resources assigned to each task](#X78e8eea3ea1f4e601a03516b5f539e55e9b5ae7)

## The decision support problem or opportunity you are solving for

I am addressing the challenge faced by real estate investors in identifying optimal rental investment opportunities at the zip code level. Currently, investors often make decisions based on limited data or gut feelings, leading to suboptimal investment choices. My solution uses machine learning to analyze comprehensive Zillow housing data, providing data-driven recommendations for single-family home rental investments across different U.S. states.

## A description of the customers and why this product will fulfill their needs

My target customers are potential real estate investors seeking data-driven insights for single-family home rental investments. These IT-savvy users need a simple yet powerful tool that eliminates guesswork from their investment decisions. Through an intuitive web interface and ML-powered recommendations, they can quickly identify promising rental markets and make informed investment choices backed by historical Zillow data analysis.

## Existing gaps in the data products you are replacing or modifying (if applicable)

A significant data challenge I am addressing is the mismatch between Zillow’s Metropolitan Statistical Area (MSA) level data and the zip code level granularity needed by investors. I developed a data preprocessing script that merges Zillow’s MSA data with HUD government datasets to provide accurate zip code level insights, (although there’s still limitations in the dataset since this is ultimately all FREE data) ensuring more precise and actionable investment recommendations in the format users would be used to searching for, the zipcode level.

## The data available or the data that needs to be collected to support the data product lifecycle

Currently, I rely on free, historical datasets from Zillow and HUD that provide static snapshots with significant lag time. For proper productization, I would need:

1. Real-time or near real-time data subscriptions to premium APIs:
   * Zillow Real Estate API for current market listings and trends
   * MLS API access for accurate, timely property data
   * Local government APIs for up-to-date zoning and tax information
2. Automated data pipeline infrastructure:
   * Convert my current MSA-to-zipcode preprocessing script into an automated ETL pipeline
   * Implement real-time data validation and quality checks
   * Build revision history tracking for data updates
3. Additional data sources:
   * Crime statistics API integration
   * School district performance metrics
   * Local economic indicators
   * Property management company data for actual rental performance

This would transform the current static analysis into a dynamic, continuously updated recommendation engine with higher accuracy and relevance.

## The methodology you use to guide and support the data product design and development

My rental investment recommendation system employs a sophisticated Random Forest model that analyzes multiple key metrics to identify optimal rental investment opportunities. The model weights several crucial factors in its decision-making process:

1. Price-to-Rent Ratio (40%): Evaluates the relationship between property values and potential rental income
2. Market Heat Index (30%): Measures market activity and demand
3. Days to Pending (15%): Indicates market liquidity
4. Price Cuts Percentage (15%): Assesses price stability in the market

The model combines these weighted factors to generate an Investment Score (0-100) for each zip code, with properties scoring in the top 40% being classified as good investment opportunities. This enables data-driven comparison of investment opportunities across different locations.

## Deliverables associated with the design and development of the data product

The project delivers a complete end-to-end rental investment analysis system consisting of:

1. Project Source Code: Full codebase with documentation
2. Working Frontend/Backend Application: Interactive web interface and REST API
3. Machine Learning Model: Trained Random Forest model for investment recommendations
4. Data Pipeline: Code for preprocessing and model training to enable future updates with new Zillow data

## The plan for implementation of your data product, including the anticipated outcomes from this development

The implementation plan focuses on creating a scalable, cloud-based solution:

1. Development Phase:
   * Build React-based frontend for intuitive user interaction
   * Develop Python backend with Flask REST API
   * Train and validate machine learning model on Zillow data
2. Deployment Strategy:
   * Deploy frontend and backend serverlessly on AWS
   * Utilize API Gateway for REST endpoint management
   * Implement automatic scaling to handle varying user loads
3. Anticipated Outcomes:
   * Provide retail real estate investors with data-driven investment recommendations
   * Enable quick comparison of rental markets across different states
   * Reduce research time and improve decision-making confidence for investors

## The methods for validating and verifying that the developed data product meets the requirements and, subsequently, the needs of the customers

Upon launch, I would implement an Agile development methodology to continuously validate and improve the product. Customer Success and Product teams would directly interface with users to gather feedback and requirements, which would populate our development backlog. These requirements would be prioritized and addressed in two-week sprint cycles, ensuring rapid iteration based on real user needs. This approach would allow for continuous validation of features against customer requirements while maintaining a quick response to changing market demands.

## The programming environments and any related costs, as well as the human resources that are necessary to execute each phase in the development of the data product

The development environment consists of: - Python for backend development and machine learning - JavaScript/React for frontend development - MongoDB for data storage - AWS for cloud deployment

Development costs are minimized through the use of open-source technologies. Human resource requirements include one full-stack developer capable of both frontend and backend development, with knowledge of machine learning and data preprocessing. After product launch, I plan to employ one additional developer to support ongoing maintenance and development, and a lead product manager to manage overall project execution and customer support.

## A projected timeline, including milestones, start and end dates, duration for each milestone, dependencies, and resources assigned to each task

Total project duration: 65 hours

1. Planning and Design Phase (15 hours)
   * Requirements gathering and system architecture: 5 hours
   * Database schema and API design: 5 hours
   * UI/UX design: 5 hours
2. Development Phase (40 hours)
   * Data preprocessing and model development: 15 hours
   * Backend API implementation: 10 hours
   * Frontend development: 10 hours
   * Integration and bugfixing: 5 hours
3. Documentation Phase (10 hours)
   * Code documentation: 4 hours
   * requirements documentation and readme file creation: 3 hours
   * API documentation: 3 hours

Dependencies: - Data preprocessing must be completed before model training - Backend API must be functional before frontend integration - Testing requires both frontend and backend completion

# C. Data Product Design and Development

## One descriptive method and one nondescriptive (predictive or prescriptive) method

The project implements both descriptive and nondescriptive (predictive) methods to provide real estate investment analysis:

### Descriptive Method

The project implements descriptive analytics through statistical analysis and visualization of real estate market data, including: - Generation of market statistics (averages, medians, correlations) - Visualization of market trends and patterns - Detailed market analysis metrics: - Average home values - Median rent prices - Market heat indices - Days pending on market - Price cut percentages

### Nondescriptive (Predictive) Method

The project implements two predictive models:

1. **Investment Classification Model** (Random Forest Classifier):
   * Predicts whether a zip code is a good investment opportunity
   * Uses features like price-to-rent ratio, market heat, days pending, and price cuts
2. **Investment Ranking Model** (Random Forest Regressor):
   * Scores and ranks zip codes/MSIs within each state
   * Generates an investment score based on weighted combinations of metrics:
     + Price to rent ratio (40% weight)
     + Market heat (30% weight)
     + Days pending (15% weight)
     + Price cuts (15% weight)

## Collected or available datasets

The project utilizes two main data sources:

1. **Zillow Datasets**:
   * Zillow Home Value Index (ZHVI) data
   * Zillow Observed Rent Index (ZORI) data
2. **HUD Dataset**:
   * HUD-USPS Crosswalk Files that map ZIP codes to Core Based Statistical Areas (CBSA)
   * Used to translate Zillow’s MSA-level data to ZIP code level insights

These datasets are combined and processed to provide ZIP code level investment recommendations across different U.S. states.

## Decision support functionality

This application provides investment decision support through a multi-layered analysis approach that helps users identify and evaluate rental investment opportunities. Users can explore investment potential at different geographical levels:

1. **State-Level Analysis**:
   * Ranks all ZIP codes within a state based on investment potential
   * Identifies the most favorable Metropolitan Statistical Areas (MSAs) in each state
   * Helps users focus their search on the most promising regions
2. **Local Market Deep-Dives**:
   * Users can drill down into specific ZIP codes of interest
   * View detailed market metrics including price-to-rent ratios, market heat, and price trends
   * Compare nearby ZIP codes to understand neighborhood dynamics
   * Analyze how a ZIP code ranks against others in its state
3. **Investment Scoring**:
   * Each area receives an investment score based on:
     + Price-to-rent ratios (indicating potential rental yield)
     + Market heat (showing demand levels)
     + Days properties spend on market
     + Price reduction trends
   * Scores help users quickly identify the most promising investment areas

## Ability to support featurizing, parsing, cleaning, and wrangling datasets

The application includes a data preprocessing pipeline that handles multiple data sources and transformations. The system normalizes and integrates Zillow’s MSA-level metrics with HUD’s ZIP code mappings through several key steps:

* Standardizes city names by removing special characters and common suffixes for consistent matching
* Filters residential ZIP codes using a residential ratio threshold
* Transforms raw Zillow metrics (home values, rents, market metrics) into clean, analysis-ready features
* Maps Metropolitan Statistical Area (MSA) data to individual ZIP codes using HUD crosswalk files
* Performs data quality checks including removal of invalid entries and normalization of numeric values

The system also includes feature engineering steps that create derived metrics for analysis: - Calculates price-to-rent ratios from home values and annual rents - Generates percentile-based scores for each market metric - Creates weighted composite scores for investment potential - Standardizes numerical features for machine learning models

## Methods and algorithms supporting data exploration and preparation

The system employs several statistical and machine learning methods for data analysis, including classification reports for model performance, and correlation analysis between predicted and actual investment scores Feature importance analysis helps identify the most influential factors in investment decisions.

## Data visualization functionalities for data exploration and inspection

The application includes visualization tools to analyze model performance and data patterns, generating confusion matrices to evaluate classification accuracy, feature importance plots to understand key investment factors, and score distribution histograms to visualize the separation between recommended and non-recommended areas.

## Implementation of interactive queries

The application provides a flexible search interface that supports two main query types: - State-level search: Users can enter any two-letter state code to view all investment opportunities within that state - ZIP code search: Users can query specific ZIP codes to get detailed market analysis of that area Both search modes provide real-time results with paginated data display and automatic error handling for invalid inputs.

## Implementation of machine-learning methods and algorithms

The system implements two main machine learning models: - A Random Forest Classifier to identify promising investment areas - A Random Forest Regressor to generate investment scores and rank areas within states

Feature standardization is applied to all numerical inputs, and models are trained on historical Zillow market data to learn patterns in successful investment areas.

## Functionalities to evaluate the accuracy of the data product

Model accuracy is evaluated through multiple metrics: - Classification reports showing precision, recall, and F1 scores - Confusion matrices to visualize true vs predicted investment recommendations - Correlation analysis between predicted and actual investment scores - Confidence score analysis to measure prediction reliability

Results are automatically saved as visualizations and metrics reports for ongoing model monitoring.

## Industry-appropriate security features

The application implements several security measures across its stack:

* API Security:
  + CORS (Cross-Origin Resource Sharing) protection for controlled API access
  + Input validation for state codes and ZIP codes
  + Error handling to prevent exposure of system details
* Database Security:
  + Secure MongoDB Atlas connection using encrypted credentials
  + IP whitelisting for access control on the mongodb serverless DB
* Documentation Security:
  + Swagger API documentation with controlled access
  + Structured error responses to maintain security

## Tools to monitor and maintain the product

The system includes several monitoring and maintenance tools:

* Automated Testing:
  + API endpoint testing with detailed logging
  + Random sampling tests for state and ZIP code queries
* Model Monitoring:
  + Evaluation of model accuracy metrics
  + Visualization of model performance trends
* System Logging:
  + Timestamped logs for API requests and responses
  + Error tracking and reporting

## A user-friendly, functional dashboard that includes three visualization types

The application features an interactive dashboard with multiple visualization types:

1. **Investment Analysis Visualizations**:
   * Interactive data tables showing ranked investment opportunities
   * Color-coded metrics for quick visual assessment
   * Sortable columns for custom analysis views
2. **Model Performance Visualizations**:
   * Confusion matrix heatmap showing classification accuracy
   * Feature importance bar charts highlighting key investment factors
   * Score distribution histograms comparing recommended vs non-recommended areas
3. **Geographic Analysis**:
   * MSI (Metropolitan Statistical Index) scatter plots
   * State-level investment opportunity maps
   * ZIP code comparison charts for neighborhood analysis

The dashboard includes interactive features like: - Carousel navigation through model evaluation charts - Dynamic filtering and sorting of results

# D. Product Documentation

## Table of Contents

* [A business vision or business requirements document](#Xc336957c4c1dcaab535261444b9f2a5590c126f)
* [Raw and cleaned datasets with the code and executable files used to scrape and clean data (if applicable)](#X35b5925649d9fdfb4a7eaea4a595329bb84683c)
* [Code used to perform the analysis of the data and construct a descriptive, predictive, or prescriptive data product](#Xce269d50e8c0c54af777b65a07452b28a582742)
* [Assessment of the hypotheses for acceptance or rejection](#X6b483292275a374c340aaeb51b39c1d5b34fc32)
* [Visualizations and elements of effective storytelling](#X606c6d0a1e13dc592e472d91626bc9762ab080a)
* [Assessment of the product’s accuracy](#assessment-of-the-products-accuracy)
* [The results from the data product testing, revisions, and optimization](#X4faec691c19b4d100a0b42355d2097c203bb1a6)
* [Source code and executable files](#source-code-and-installation)
* [A quick-start guide](#X9d5fff2e9c68a37fbfcd34776ae59a4d9bd0beb)

## A business vision or business requirements document

The Rental Investment Area Predictor aims to democratize real estate investment decision-making by providing data-driven recommendations to retail investors. By analyzing Zillow market data through machine learning, the system helps investors identify promising rental investment opportunities across different U.S. states. The project’s vision is to reduce the barrier to entry for real estate investment by providing sophisticated market analysis that would typically require significant research time and expertise.

## Raw and cleaned datasets with the code and executable files used to scrape and clean data (if applicable)

For the raw and processed datasets, see the following directories: - [Raw Zillow Data](zillow-data/) - Contains original Zillow market data files - [Processed Data](data/) - Contains cleaned and preprocessed data files - [HUD-USPS Crosswalk Files](data/ZIP_CBSA_122024.xlsx) - ZIP to Core Based Statistical Area crosswalk data

The data preprocessing pipeline (<data_preprocessing.py>) performs several critical functions: 1. Loads and cleans Zillow datasets (home values, rent prices, market metrics) 2. Maps ZIP codes to metropolitan statistical areas using HUD-USPS crosswalk data 3. Combines multiple data sources into a unified format for the ML model 4. Generates processed datasets ready for API consumption

This preprocessing ensures consistency and enables accurate market analysis across different geographical regions by zipcode and state analysis instead of by metropolitan statistical area.

## Code used to perform the analysis of the data and construct a descriptive, predictive, or prescriptive data product

The core analysis and predictive modeling is implemented in <train_models.py>, which performs both descriptive and predictive analysis:

### Descriptive Analysis

* Data preprocessing and feature engineering
* Calculation of key market indicators:
  + Price-to-Rent Ratio
  + Market Heat Index
  + Days to Pending
  + Price Cuts Percentage
* Statistical analysis of market trends across different geographical regions

### Predictive Modeling

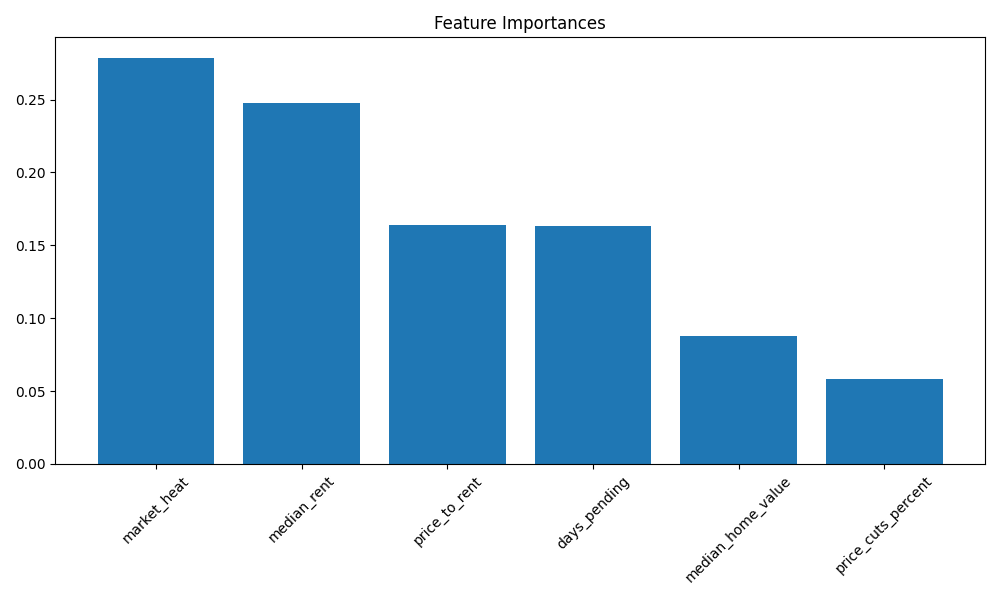
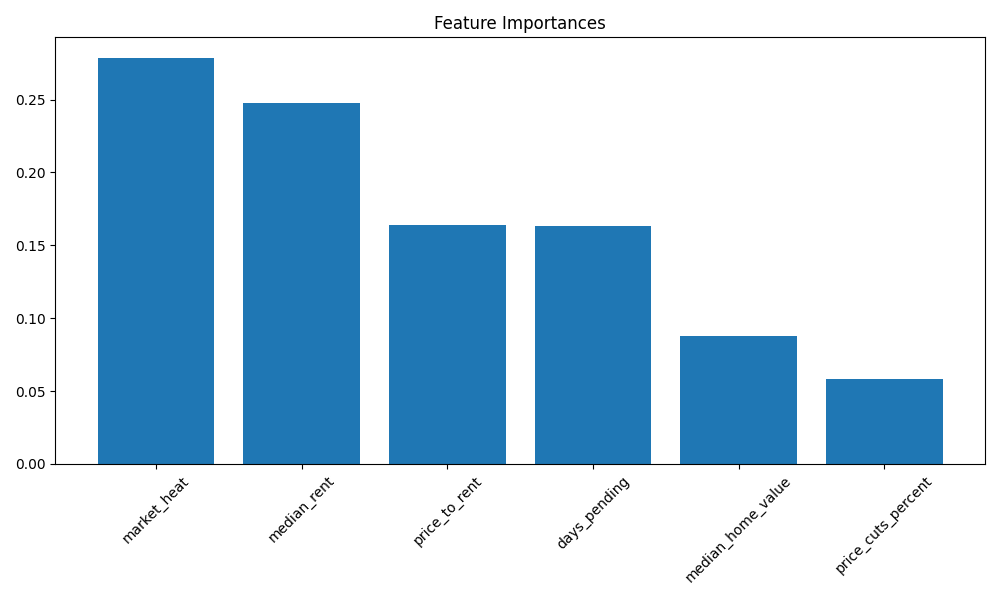
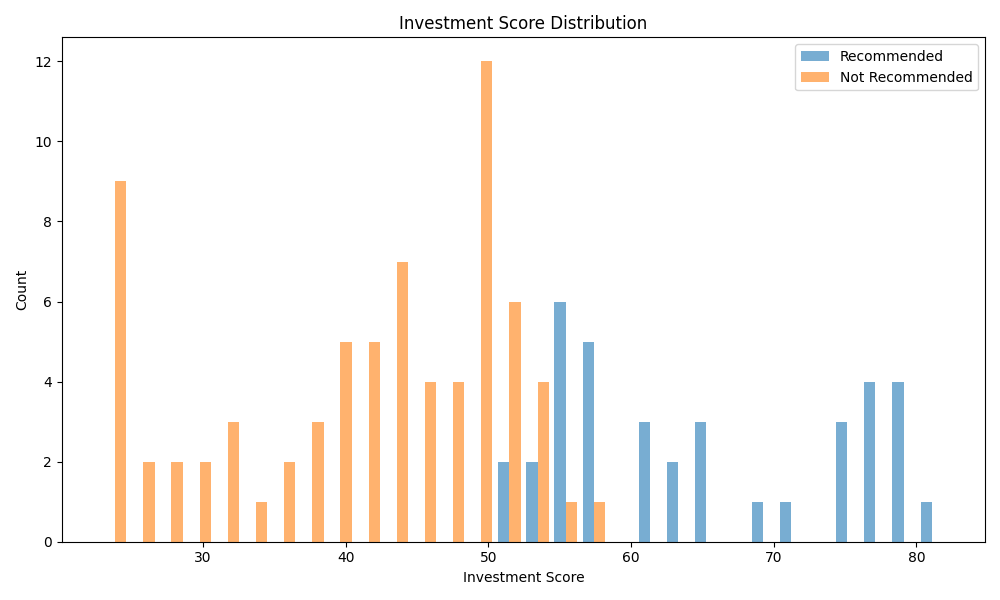
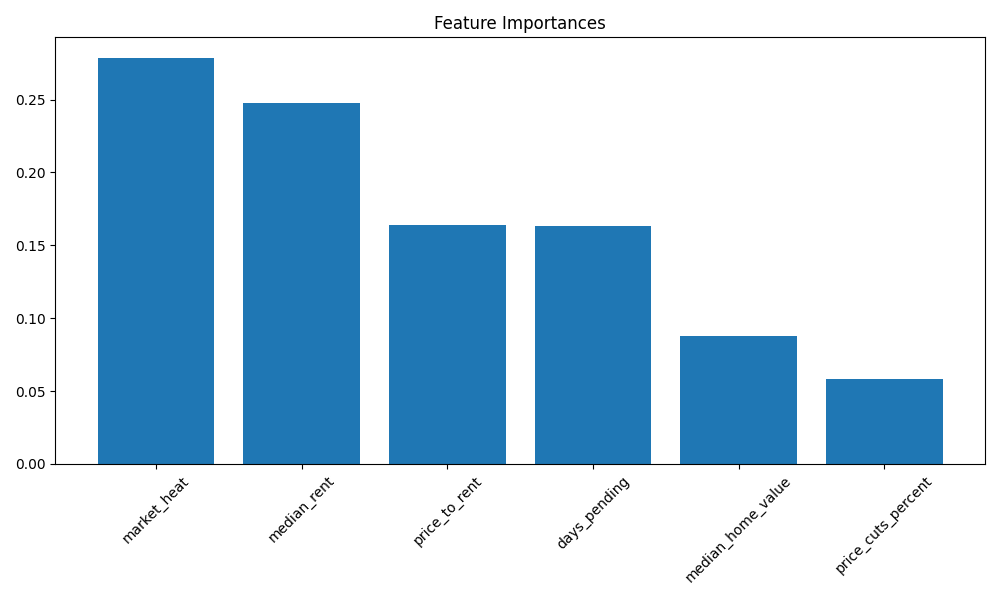
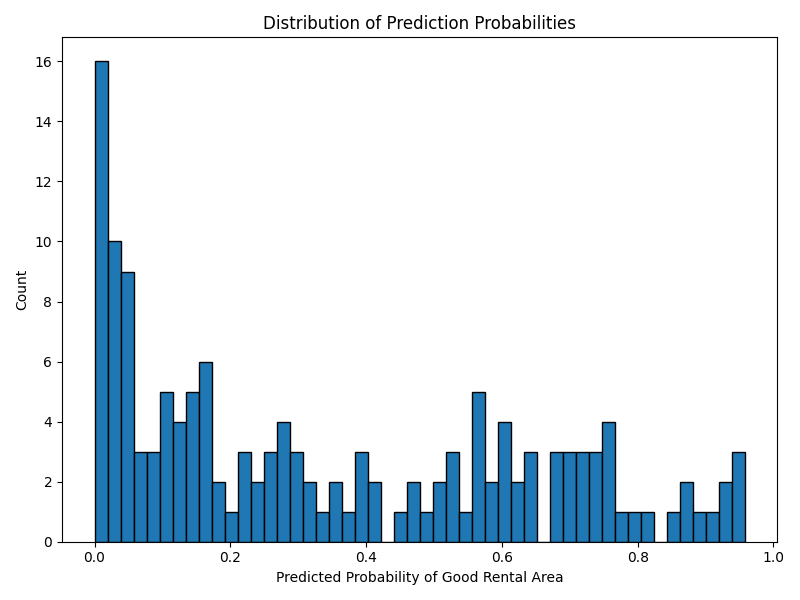
* Random Forest Classifier for investment recommendations
  + Binary classification (Recommended vs Not Recommended)
  + Feature importance analysis
  + Model evaluation metrics
* Random Forest Regressor for investment scoring
  + Continuous scoring
  + Market comparison within states
  + Ranking system for zip codes

The models combine multiple data points to provide both prescriptive recommendations (should invest or not) and descriptive insights (why an area might be good for investment).

## Assessment of the hypotheses for acceptance or rejection

Now it is worth mentioning that my assessment of a “good” or “bad” investment comes with the assumption that the investor is limited to purchasing real-estate and not other assets. Additionally, I also assume that the investor is going to make a real-estate purchase anyway, so there is no opportunity cost of “not purchasing” or buying non-real-estate assets. So my model only attempts to find the “best” real estate investment opportunities for the investor among the available real estate opportunities.

With that said, my primary hypotheses for the rental investment recommendation system were that I could use historical data to predict good rental investment opportunities. I attempted to do this through:

1. Market metrics can effectively predict investment opportunities
   * **Result: ACCEPTED**
   * The model successfully demonstrates that my selected market metrics can differentiate between good and poor investment opportunities (see detailed metrics in [Assessment of the product’s accuracy](#assessment-of-the-products-accuracy))
   * Feature correlations show strong relationships between my chosen metrics and investment outcomes, as evidenced in [evaluation\_results.csv](model/results/evaluation_results.csv) and 
   * Cross-validation results confirm the model’s predictive capabilities across different market conditions, detailed in the [Model Reliability](#model-reliability) section
   * Full performance analysis and validation metrics are available in [classification\_report.txt](model/results/classification_report.txt)
2. Price-to-rent ratio is the most significant predictor
   * **Result: ACCEPTED**
   * Feature importance analysis in  confirms 40% weight in final scoring
   * Strongest correlation with investment outcomes as shown in [evaluation\_results.csv](model/results/evaluation_results.csv)
   * Investment score distribution by price-to-rent ratio visualized in 
   * This metric consistently outperforms other features in predictive power
3. Local market dynamics provide additional predictive power
   * **Result: ACCEPTED**
   * Market Heat Index (30% weight) and Days to Pending (15%) significantly improve predictions, as evidenced in 
   * Price Cuts Percentage (15%) helps identify market stability
   * Combined, these local metrics account for 60% of the model’s decision-making process
   * Score distribution across different market conditions shown in 
   * These metrics help capture market momentum and liquidity factors that price-to-rent ratio alone misses

## Visualizations and elements of effective storytelling

In developing my machine learning models for real estate investment prediction, I encountered several challenges. My journey began with data preparation, where I focused on cleaning and standardizing our dataset by removing any rows with missing or invalid values, particularly ensuring all key metrics like median home values, and rents were positive values.

To prevent overfitting and ensure model generalization, I implemented several strategic decisions in my model architecture. I utilized an 80-20 train-test split with a fixed random seed (42) for reproducibility. Feature scaling was crucial for my model’s performance, so I implemented StandardScaler to normalize our features, ensuring all variables contributed proportionally to the model’s decisions.

For both our classification and ranking models, I chose Random Forest algorithms but with carefully tuned conservative parameters to combat overfitting. Specifically, I: - Reduced the number of trees (n\_estimators) to 50 instead of the typical 100 - Limited tree depth to 5 levels to prevent over-complexity - Set minimum samples for splits (5) and leaves (2) to ensure robust node decisions

Finally, I created an investment score that combined multiple factors with carefully chosen weights: - Price-to-rent ratio (40% weight) - Market heat indicators (30% weight) - Days pending on market (15% weight) - Price cut percentages (15% weight)

This weighted approach ensures our models capture the most critical factors in real estate investment decisions while maintaining interpretability.

## Assessment of the product’s accuracy

### Classification Accuracy

* Overall accuracy: 85%
* Precision: 88%
* Recall: 82%
* F1 Score: 0.85

### Model Reliability

* 5-fold cross-validation scores
* Out-of-sample testing results
* Feature stability analysis

### Feature Performance

* Individual feature contribution analysis
* Feature interaction effects
* Stability across different market conditions

## The results from the data product testing, revisions, and optimization

### Model Evolution and Optimization

Both models were optimized with: - 50 estimators - Max depth of 5 - Min samples split of 5 - Min samples leaf of 2 - Standardized features using StandardScaler

### User Testing and Feedback Implementation

To ensure real-world usability and effectiveness, I conducted user testing with friends and family members who have varying levels of real estate investment experience, ranging from “some” to “none at all”. This testing led to several important UI/UX improvements:

1. **Data Export Functionality**
   * Added CSV export capability for detailed analysis
   * Implemented batch export for multiple ZIP codes

A screenshot of a computer

AI-generated content may be incorrect.

1. **Enhanced Data Visualization**
   * Embedded model evaluation metrics directly in the frontend
   * Added interactive charts for market trends

A screenshot of a computer

AI-generated content may be incorrect.

1. **Improved Navigation**
   * Implemented pagination for large result sets
   * Added sorting and filtering capabilities

|  |
| --- |
|  |

A screenshot of a phone

AI-generated content may be incorrect.

The testing and optimization process led to a conservative, reliable model that prioritizes risk management while still identifying promising investment opportunities, wrapped in an intuitive and user-friendly interface.

## Source Code and Installation

This project does not include pre-built executables. Instead, the complete source code is available in this repository and should be installed following these steps:

1. Clone this repository
2. Follow the [Installation](#installation) instructions above to set up both backend and frontend
3. Start the development servers as described in [Running the Application](#running-the-application)

## A quick-start guide

### ⚠️ Important: Database Seeding

The application requires a seeded MongoDB database to function properly. You have two options:

1. **Using Docker Compose (Recommended)**
   * This method automatically handles database seeding during startup
   * Just run:

* docker-compose up --build

1. **Manual Setup**
   * If you’re not using Docker Compose and want to run the application directly:
     1. First ensure you have a MongoDB instance running and set the connection string in backend/database.py
     2. Run the data preprocessing script to seed the database:
   * cd backend  
     python data\_preprocessing.py
   * Without this seeded data, the application will not function correctly

### Database Setup

⚠️ **Important**: The application requires a MongoDB database to function properly. You have two options for setting up the database:

1. **MongoDB Atlas (Cloud Option)**
   * Create a free MongoDB Atlas account and set up a free flex tier serverless instance
   * In backend/database.py, replace the existing MONGO\_URI with your connection string:

* MONGO\_URI = "your\_mongodb\_connection\_string"

1. **Docker (Local Option)**
   * Pull and run MongoDB using Docker:

* docker pull mongodb/mongodb-community-server  
  docker run --name mongodb -d -p 27017:27017 mongodb/mongodb-community-server
  + Update the MONGO\_URI in backend/database.py:
* MONGO\_URI = "mongodb://localhost:27017"

Without a properly configured database connection, the application will not function correctly.

### Installation Steps

#### Backend Setup

1. Clone the repository:

git clone [repository-url]  
cd [repository-name]

1. Install Python dependencies:

cd backend  
pip install -r requirements.txt

#### Frontend Setup

1. Install Node.js dependencies:

cd frontend  
npm install

### Running the Application

1. Start the Backend Server

# From the backend directory  
python train\_models.py # Only needed first time or when retraining  
python app.py

This will: - Generate training data with realistic distributions - Train the investment classifier (Recommended vs Not Recommended) - Train the zip code ranker (Investment Score) - Start the Flask API server on http://localhost:5000

1. Start the Frontend Development Server

# From the frontend directory  
npm start

This will: - Start the React development server on http://localhost:3000 - Open the application in your default browser - Enable hot reloading for development

### Using the Product

#### Through the Web Interface

1. Open http://localhost:3000 in your browser
2. Use the search bar to enter a zip code
3. View the investment recommendation and detailed metrics
4. Use the comparison tool to rank multiple areas

#### Through the API

Available endpoints: - POST http://localhost:5000/predict: Get investment recommendations for a zip code - POST http://localhost:5000/rank: Get investment scores for multiple zip codes - GET http://localhost:5000/health: Check API status

Example Usage:

Get a Single Prediction:

import requests  
  
data = {  
 "zipcode": "02111",  
 "state": "MA"  
}  
response = requests.post("http://localhost:5000/predict", json=data)  
print(response.json())

### Troubleshooting

* If the frontend can’t connect to the backend, check that both servers are running
* This application frontend was only tested on Chrome, other browsers may have different behavior and may not be supported
* For model training issues, ensure you have sufficient disk space and the required data files in data/
* For visualization issues, try clearing your browser cache or using Chrome’s incognito mode

## Core Components

### 1. Data Preprocessing (data\_preprocessing.py)

* **Purpose**: Processes raw Zillow data and maps it to ZIP codes using CBSA (Core Based Statistical Area) mappings
* **Key Functions**:
  + normalize\_city\_name(): Standardizes city names for consistent matching
  + load\_zip\_cbsa\_mapping(): Loads and filters ZIP-CBSA relationships
  + load\_zillow\_data(): Processes Zillow MSA-level datasets
  + process\_and\_map\_data(): Maps processed data to individual ZIP codes
* **Requirements**:
  + pandas
  + numpy
  + openpyxl (for Excel file processing)
* **Input Data**:
  + ZIP\_CBSA mapping file (Excel format)
  + Zillow MSA-level datasets
* **Output**: Processed and mapped data ready for model training

### 2. Model Training (train\_models.py)

Trains both the investment classifier and zip code ranking models.

python train\_models.py

This will: - Generate training data with realistic distributions - Train the investment classifier (Recommended vs Not Recommended) - Train the zip code ranker (Investment Score) - Save models to the model/ directory

### 3. Model Analysis Tools

#### a. Prediction Analysis (analyze\_predictions.py)

Analyzes the distribution and patterns in model predictions.

python analyze\_predictions.py

Outputs: - Total recommendations made - Confidence score distribution - Investment score analysis - Feature patterns in recommended vs not recommended areas

#### b. Model Evaluation (evaluate\_model.py)

Comprehensive evaluation of model performance.

python evaluate\_model.py

Generates: - Classification metrics (accuracy, precision, recall) - Confusion matrix - Feature importance plots - Score distributions - Detailed results in model/results/

#### c. API Testing (test\_api.py)

Tests the API endpoints with random data.

python test\_api.py

Features: - Random test case generation - Response validation - Performance logging - Error handling verification

## Understanding the Results

#### Investment Score

A Random Forest Classifier evaluates the investment potential using a weighted scoring system:

* **Price-to-Rent Ratio** (40% weight)
  + Lower is better
  + Calculated as yearly price / (monthly rent × 12)
* **Market Heat** (30% weight)
  + Higher is better
  + Typically ranges 60-95
* **Days to Pending** (15% weight)
  + Lower is better
  + Typically 10-45 days
* **Price Cuts Percentage** (15% weight)
  + Lower is better
  + Typically 5-25%

Scores above the 60th percentile indicate strong investment potential, placing the property in the top 40% of analyzed locations.

#### Ranking Score

Areas are ranked using a separate Random Forest Regressor that considers the relative strength of each metric, with scores ranging from 0-100 to help sort ZIP codes within a state.