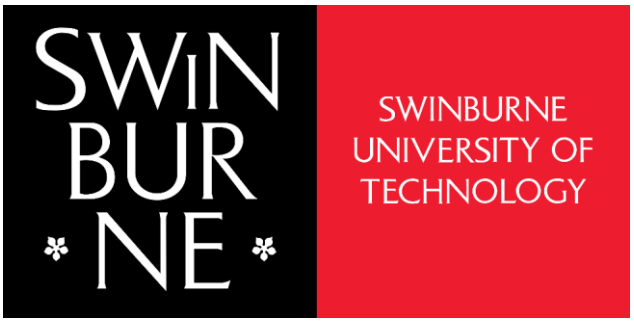

Full Stack Predicting Housing Market Web Development

Conducted by

Minh Cao – 104179289
Sean Smith – 103703153

Tutor: Ningran Li

November 3, 2024

1. Introduction

The Australian housing market is constantly fluctuating, making it challenging to predict property prices. Property owners could benefit greatly from an easy-to-use tool to estimate the value of their homes. To address this need, we have developed a machine-learning algorithm that predicts property prices based on key features, such as the number of bedrooms, bathrooms, and other attributes. To make this tool accessible, we've created a website for "Vanga RealEstate," a mock real estate platform designed to help users estimate the value of their homes. Through this platform, homeowners can gain insights into their property's potential value with ease.

2. System Architecture

The project's system architecture is designed to deliver real-time forecasting of the housing market, insight into factors influencing a house price, and historical median housing price records by month each year. The architecture comprises three primary layers: the Client Layer, the Server Layer, and the External API Layer. These layers collaborate to retrieve, process, and deliver housing price data to end users responsively and intuitively.

Link to system design: https://drive.google.com/file/d/1u2bjx4e-NYoClVcLVB5Vzt7VW8_R_tz1/view?usp=sharing

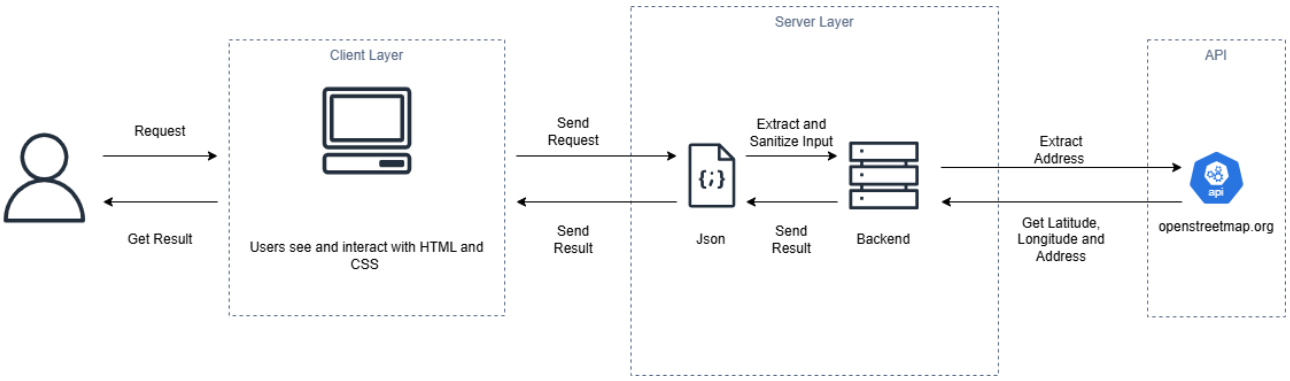

Figure 1: The system architecture diagram

2.1. Client Layer (Front-End)

The front-end consists of 3 pages; The Home, About, and Prediction Page. The website is designed with six components (Button, Footer, Logo, Navbar, SearchForm, and Testimonial Card). Each page has been designed using the six above components with additional CSS (e.g., for images) and HTML. Within the file structure, CSS styling is provided for each component.

Lastly, the assets folder contains images required for rendering image elements on the webpage. The code provides relative paths to these directories.

2.2. Server Layer (Back-End)

The server layer plays a crucial role in handling the logic of the website and processing hub for the application which is built with Python and FastAPI. The server receives the route request from the client layer, then those requests are sanitized to extract the main information. After that, the server communicates with the external API to get the relevant information to transmit the data into the AI model to get the prediction.

3. Front – End Implementation

3.1. User Input And Validation

When users enter the website, the first section they are presented with is the hero content (home page). The website aims to provide key information about the business Vanga Realestate, whilst providing users with a quick and easily accessible prediction prediction page. To achieve this, navigation has been distributed between the hero content and the navigation page. Lastly, the prediction page provides a form for users to enter details about their house. The form validates the user's input by checking for incorrect data types. Inputs expecting integers (e.g.,, Number of Bedrooms) do not accept strings. They also do not accept negative values. The input form also notifies users, when they have not entered details. After submission, input fields that have not been entered are highlighted in red. Additional text also appears at the bottom of the form after submission (“Please enter all required fields”), to notify users when a field has not been entered. This is an extra message that has been provided to users to help them identify errors in their input. After successfully submitting the form, if the address can’t be found on the map, the user will be navigated to the error page and display an error message with the button to back to the search form

3.2. Data Flow

In the client layer, data is sent to the back end for prediction through an HTTP POST request using Axios. When a user submits property details (such as address, house type, and number of rooms), the information is compiled into an object called formData. This formData is then sent to the back end by Axios through an API call to http://localhost:8000/predict. The back end processes the request and responds with a JSON object containing the predicted price, distance

to the city center, geographic coordinates, historical median prices, nearby properties with similar prices, and feature importance values (SHAP values) indicating how each input factor contributes to the price prediction. After receiving this data, the front end uses React Router's navigate function to transition to the /predict page, passing along the prediction data in the state. This allows the /predict page to display the prediction results, ensuring a seamless flow from data submission to result visualization.

3.3. Design

The website is designed to be responsive across multiple screen sizes, including mobile devices. To enhance readability, a max-width attribute is utilized to prevent content from becoming too wide. Consistent spacing is achieved through padding and gap attributes defined in rem units. Media queries adjust the layout from rows to columns based on screen size, following breakpoints outlined in Bootstrap documentation. For smaller screens, content is aligned vertically. Additionally, the dimensions of containers are defined using percentage values or VH units, allowing them to adapt relative to their content.

4. Back – End Implementation

4.1. FastAPI Server Setup

The back end of this application is implemented using FastAPI, a modern, fast (high-performance) web framework for building APIs with Python (Chen, J. 2023). This FastAPI server handles requests from the client, engages with a machine learning model for generating predictions, and connects with external services when necessary. Additionally, the server includes CORS middleware to allow cross-origin requests originating from the front-end.

a. First Setup

FastAPI is initialized to create an API server instance and the CORS middleware is added to enable cross-origin requests from the client layer (front-end)

b. Initialize ML Model

Initialize the instance from the MLmodel class and the instance consists of 2 key functions predict() and recommend_nearby_houses().

- The predict function generates the price prediction of the property based on the user’s input. Furthermore, the function also returns the distance of the property to

CBD, the property coordinates, center coordinates, median price, and Shap values.

- The recommended nearby house function returns a list of recommended properties where each object represents a recommended property with details like Longitude, Latitude, Price, and Address

c. Request Body Schema

The PredictionRequest schema, created using Pydantic, validates incoming requests to confirm they include essential fields like address, house type, bathrooms, bedrooms, carpark, building area, and land size.

4.2. API Endpoint Documents

API documentation offers developers precise guidance on utilizing an API, outlining the available endpoints, as well as the formats for requests and responses, authentication methods, and error management.

a. FastAPI

1. Root Endpoint

Endpoint: '/'

Method: 'GET'

Description: This endpoint ensures that the API server is running

```
{
  "message": "Welcome to the House Price Prediction API"
}
```

Figure 2: JSON format of root endpoint

2. House Price Prediction Endpoint

Endpoint: '/predict'

Method: 'POST'

Description: This transmits the house property from the user's form to the backend and they will be sanitized to go to the ML model

JSON Request:

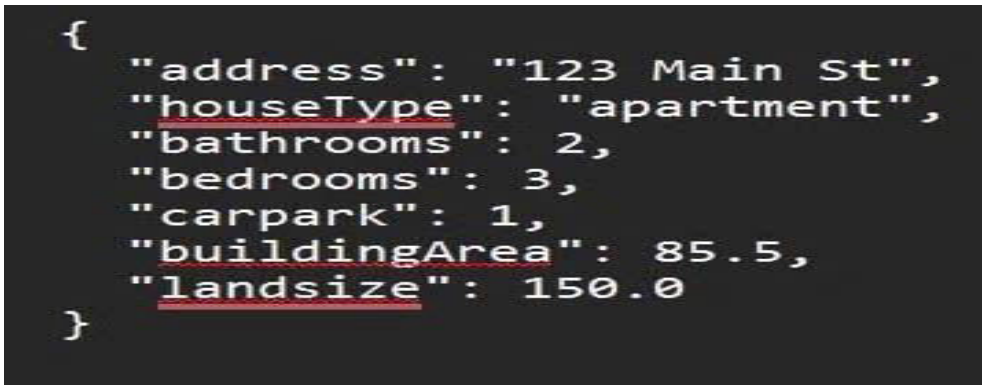

Figure 3: JSON request format

JSON Response:

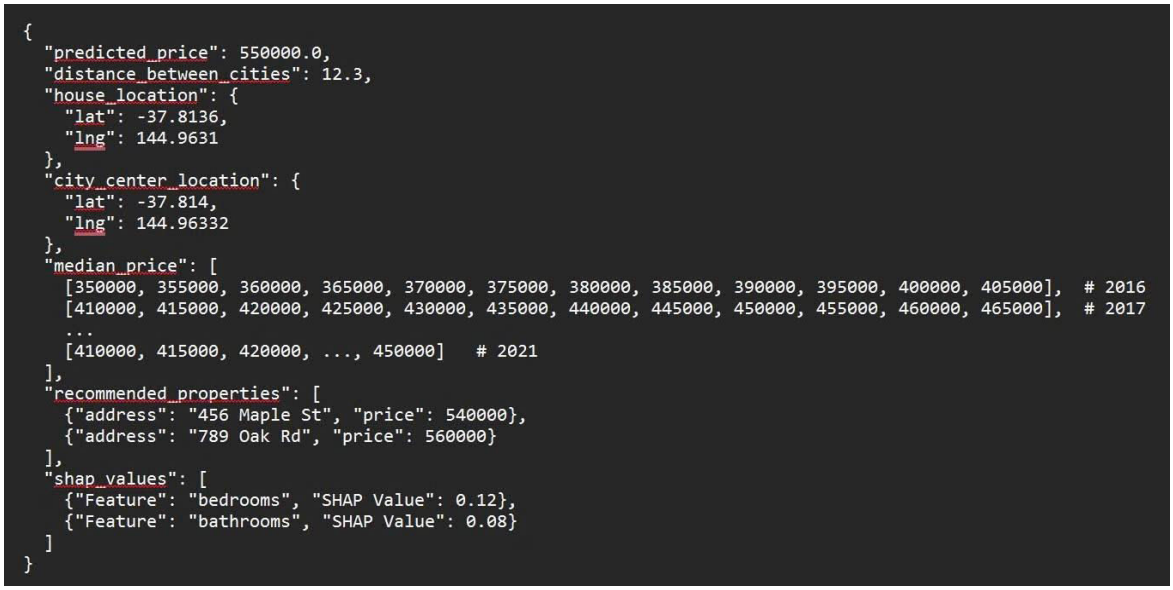

Figure 4: JSON response format

3. Error Handling

Endpoint: '/error'

Description: This navigates the user to the error page and displays the error on the screen

```
{
  "detail": "No results found for the given address."
}


{
  "detail": "Error: Request failed with status code 500."
}
```

Figure 5: JSON format of error message

b. OpenStreetMap API

Description: Leveraging OpenStreetMap API to get all relevant details about the address for machine learning model

Base URL:
'https://nominatim.openstreetmap.org/search.php?q={address}&format=jsonv2'
'https://nominatim.openstreetmap.org/reverse?lat={lat}&lon={lon}&format=jsonv2'

Method: 'GET'

Success Response: 200 OK

Authentication: using personal email to verify

```
headers = {
    "User-Agent": "MyGeocodingApp/1.0 (caominh418@gmail.com)"
}
```

Figure 6: Headers to authenticate and use the OpenStreetMap API

Response Structure: JSON

```
{
  "place_id": 50875452,
  "osm_type": "way",
  "osm_id": 738167610,
  "lat": "-33.3922702",
  "lon": "151.473351",
  "category": "highway",
  "type": "tertiary",
  "place_rank": 26,
  "importance": 0.10000999999999993,
  "addresstype": "road",
  "name": "Cresthaven Avenue",
  "display_name": "Cresthaven Avenue, Bateau Bay, Central Coast Council, New South Wales, 2261, Australia",
  "boundingbox": [
    "-33.3923132",
    "-33.3922445",
    "151.4731593",
    "151.4734656"
  ]
}
```


Figure 7: JSON format of OpenStreetMap API

5. AI Model Integration

The AI models are essential for forecasting housing prices and recommending nearby properties within a similar price range. This application relies on two main models: XGBoost and DBScan, chosen for their high accuracy and performance demonstrated in Assignment 2 (MSE: 0.006, R^2: 85%, silhouette score: 0.84). Trained on a dataset with 19 features, these models require additional data from an external API to generate predictions. The process begins by sanitizing the input to remove any spaces or commas. The cleaned address is then sent to the OpenStreetMap API to retrieve the latitude, longitude, and display name of the address. All essential features are gathered and fed into the XGBoost model to make a prediction. Additionally, the input features are analyzed using the feature_importance() function, which leverages Shap values to determine each feature's contribution to the predicted price. Finally, the latitude, longitude, and predicted price from XGBoost are passed to the DBScan model to identify properties within a similar price range and location. This combined approach enables both accurate forecasting and relevant property recommendations.

6. Conclusion

Our solution aimed to investigate and analyse housing prices, as well as the market performance of different types of houses. To achieve this aim, we decided to develop an application that could be used by property owners in the real world, to predict the cost of their house whilst providing users with additional insights. We included attributes such as the distance to the city center to help users understand how this factor affects housing costs, featuring a prominent graphical display for visibility. Additionally, a pie chart illustrating Feature Importance for Price (SHAP) clarifies the factors influencing house prices, potentially impacting property owners' and buyers' preferences.

After reviewing the data, property owners may feel justified in raising their asking prices, while buyers might consider alternatives like units or apartments over townhouses. However, we acknowledge areas for improvement. The pie chart may be confusing for users unfamiliar with machine learning terminology, suggesting a need for clearer titles or more accessible visualizations. The front-end design could be enhanced, with larger navigation options more centered and a consistent layout across pages to avoid user confusion. Lastly, the visual appeal of the About and Prediction pages could be improved for a more attractive overall experience.

Additional Functionality

We propose adding functionality that allows users to predict their house's cost relative to other properties in the same area, enhancing prediction accuracy. Additionally, incorporating an investment score feature for potential property owners and investors would help assess the viability of investments based on trends like house type and land size. This would enable users to make more informed investment decisions.

References

Chen, J. (2023). Application Technology of Atmospheric Dispersion Models Based on FastAPI+Vue. *Academic Journal of Engineering and Technology Science*, 6(11). doi:https://doi.org/10.25236/ajets.2023.061118.

Font Awesome 5 (2024). *Font Awesome 5*. [online] Fontawesome.com. Available at: https://fontawesome.com/.

Google (2019). *Google Fonts*. [online] Google Fonts. Available at: https://fonts.google.com/.

Mark Otto, Jacob Thornton, and Bootstrap (2024). *Breakpoints*. [online] getbootstrap.com. Available at: https://getbootstrap.com/docs/5.0/layout/breakpoints/.

React, K. (2024). *Now Ui Kit React by Creative Tim*. [online] Creative Tim. Available at: https://demos.creative-tim.com/now-ui-kit-react/?_ga=2.56028715.1816365111.1730340274-1329238446.1729926708#/index [Accessed 3 Nov. 2024].