CSE 4/587 Data Intensive Computing

Project Phase 3

Title: Property Sales: Melbourne City

Team Details:

Harshavardhan Sivadanam -hsivadan (50478880)
Nikhil Kumar vadigala -nvadigal (50485135)
Sai Sree Depa- saisreed (50478013)

Problem Statement

The main objective of this project is to develop predictive models for forecasting real estate prices in Melbourne City accurately. The objective is to create models that can predict property sale prices by considering a wide range of factors, including the number of rooms, property type, location (suburb, postcode, latitude, longitude), building characteristics (bedroom count, bathroom count, car parking, land size, building area), year of construction, and other relevant attributes available in the dataset. This project should deliver practical insights for people and parties interested in the Melbourne housing market by forecasting property values. Making decisions on the purchase, sale, or investment of real estate in the city can be aided by these forecasts. The project is to help different people involved in the real estate market. For those selling homes, they want to know the best price possible by understanding what affects property prices. If you're looking to buy a house, this project provides detailed information to help you make smart choices. Investors always want to make money, and this project helps them find places for wise investments. Plus, it makes the whole market more transparent and helps everyone make better decisions.

Contribution to the Problem Domain:

This project has the potential to significantly contribute to the domain in the following ways:

This project can offer data-driven insights into the property market by evaluating a large dataset. Different people like homebuyers, sellers, and investors can follow these insights. This promotes price transparency and aids in decision-making for the general public. By locating places with rapid increases in property value, the project can help real estate investors make wise location decisions. Understanding the problems with housing affordability can help decision-makers understand the necessity of housing policies and efforts.

Data Sources

We obtained the dataset "Property Sales of Melbourne City" from Kaggle, which can be accessed using the following link: [Property sales: Melbourne City on Kaggle.]()

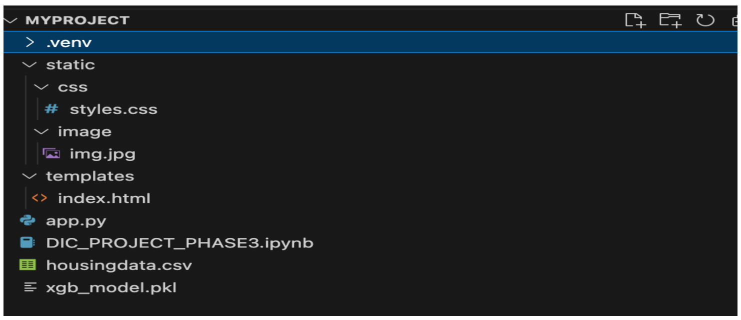
Working Instructions

1. Data Preprocessing and Model Development (IPYNB File)

- Open the IPython Notebook file (DIC_PROJECT_PHASE3.ipynb) containing data preprocessing and model development.
- Run each cell sequentially to execute data preprocessing steps (cleaning, feature engineering, etc.) and train the models.
- Ensure all necessary libraries are installed within the Python environment.

2. Flask Application (app.py)

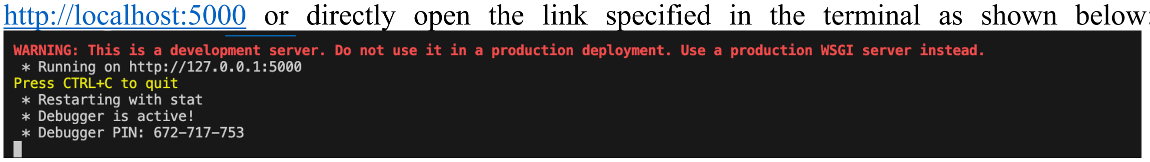
- Navigate to the Flask application directory.
- Make sure all files are in their respective directory. The directory structure is as follows:



- Ensure Flask (and other required libraries) is installed.
- To install Flask, run the command 'pip install Flask'.
- Run the Flask application using the command: ‘python app.py’

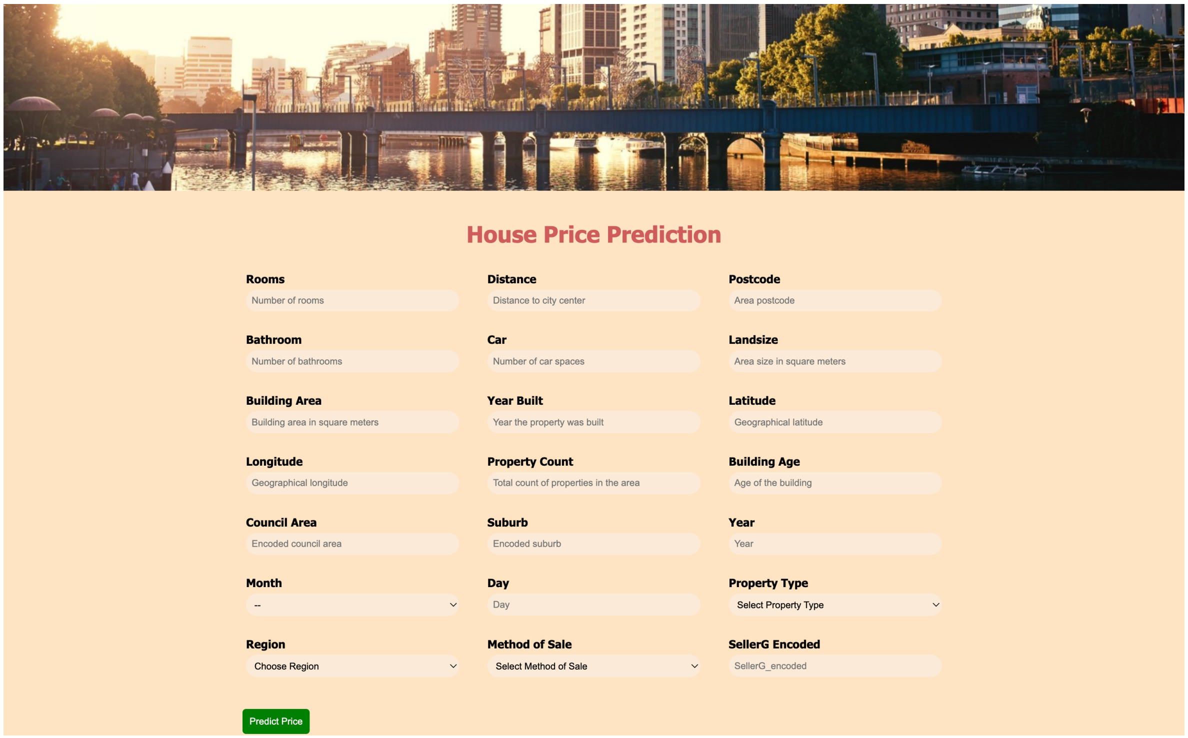
3. User Interface (HTML, CSS)

- To open the web browser and navigate to the associated web browser to locally run on http://localhost:5000 or directly open the link specified in the terminal as shown below:



- Once the page is opened, interact with the user interface to input property details for price prediction.
- Enter the details, and click on the button predicted price, to view the prediction and also to visualize the graphs for the input data and predicted price.

The web page is as follows:



Utilization of Models from Phase 2

Model Selection:

The project evaluated several regression models including DecisionTreeRegressor, LinearRegression, GradientBoostingRegressor, AdaBoostRegressor, AdaBoostRegressor with Decision Trees, RandomForestRegressor, and XGBRegressor.

Based on performance metrics, the XGBoost Regressor was chosen as the top-performing model due to its 81.12% prediction score and the lowest Mean Absolute Error of 135,981.67.

	Algorithm	Prediction_score	Mean_Absolute_Error
0	DecisionTreeRegressor	60.869469	193390.613999
1	LinearRegression	63.608127	205623.054656
2	GradientBoostingRegressor	76.484865	154690.035853
3	AdaBoostRegressor	35.829307	304152.236669
4	AdaBoostRegressor With DT	78.340502	142282.921219
5	RandomForestRegressor	79.160705	141453.231109
6	XGBRegressor	81.117168	135981.667465

Tuning and Parameters

Parameters for each model were optimized through grid search and cross-validation to enhance their predictive capabilities. Specific parameters like learning rate, max depth, and number of estimators were fine-tuned to achieve optimal performance.

Integration

The XGBoost Regressor is integrated into the final system, enabling users to input property details and obtain accurate price predictions.

Web Application Overview

IPYNB File:

This code snippet demonstrates the process of saving the trained XGBoost model to a pickle file (xgb_model.pkl). This saved model is then used within the Flask application for predictions.

```
import pickle
#Choosing XGboost Model because of its high accuracy
# Saving the XGBoost model to a file using pickle
with open('xgb_model.pkl', 'wb') as file:
    pickle.dump(XGBR, file)
```

Flask Backend (app.py):

This Python file handles model loading and prediction using Flask, allowing interaction with the trained model through API endpoints. The below figure shows Flask initialization and model loading.

```
app = Flask(__name__)
CORS(app)

#considering highest accuracy gained model
with open('xgb_model.pkl', 'rb') as file:
    xgb_model = pickle.load(file)
```

HTML Frontend (index.html): Connection with the Server

Contains a form (predictionForm) with a 'Predict Price' button that triggers a POST request to the Flask backend (/predict route) on submission. Handles form submission using fetch to send data to the Flask server and receives the predicted price as JSON.

```
document.getElementById('predictionForm').addEventListener('submit', function(event) {
    event.preventDefault();

    var form = this;
    var formData = new FormData(form);

    // Send the form data to your Flask server
    fetch('http://127.0.0.1:5000/predict', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
    // Display the prediction result
        var predictedPrice = data.prediction[0];
```

Functionality:

Data Submission and Prediction: JavaScript functions capture the form data on submission and send it to the Flask backend (/predict route) via a POST request. The backend processes the received data using the XGBoost model and returns the predicted price as JSON. The predicted price is dynamically displayed on the interface.

Chart Display (On Prediction): If the prediction is successful, bar charts representing the input values and the predicted price are dynamically generated and displayed using Chart.js. These charts provide a visual representation of the input parameters and the predicted house price for better understanding and analysis.

User Interface:

The user interface is an HTML form operating within a Flask-powered web application. Flask is a Python-based web framework used to develop web applications. It offers a robust set of tools and libraries for building web applications using Python. This setup enables users to input property details and obtain predicted house prices.

Input Fields: Each property detail (Rooms, Distance, Postcode, etc.) has a corresponding input field. Users can enter numerical data such as number of rooms, distance, Building area size, etc., into these fields.

Dropdown Menus:

- Month: Users select the required month from this dropdown.
- Property Type: Users select the property type (House, Townhouse, Unit/Apartment) from a dropdown.
- Region: A dropdown allows users to choose the property's region from various options.
- Method of Sale: Users select the method of sale (Property Sold, Sold After Auction, Property Sold Prior, Vendor Bid) from this dropdown.

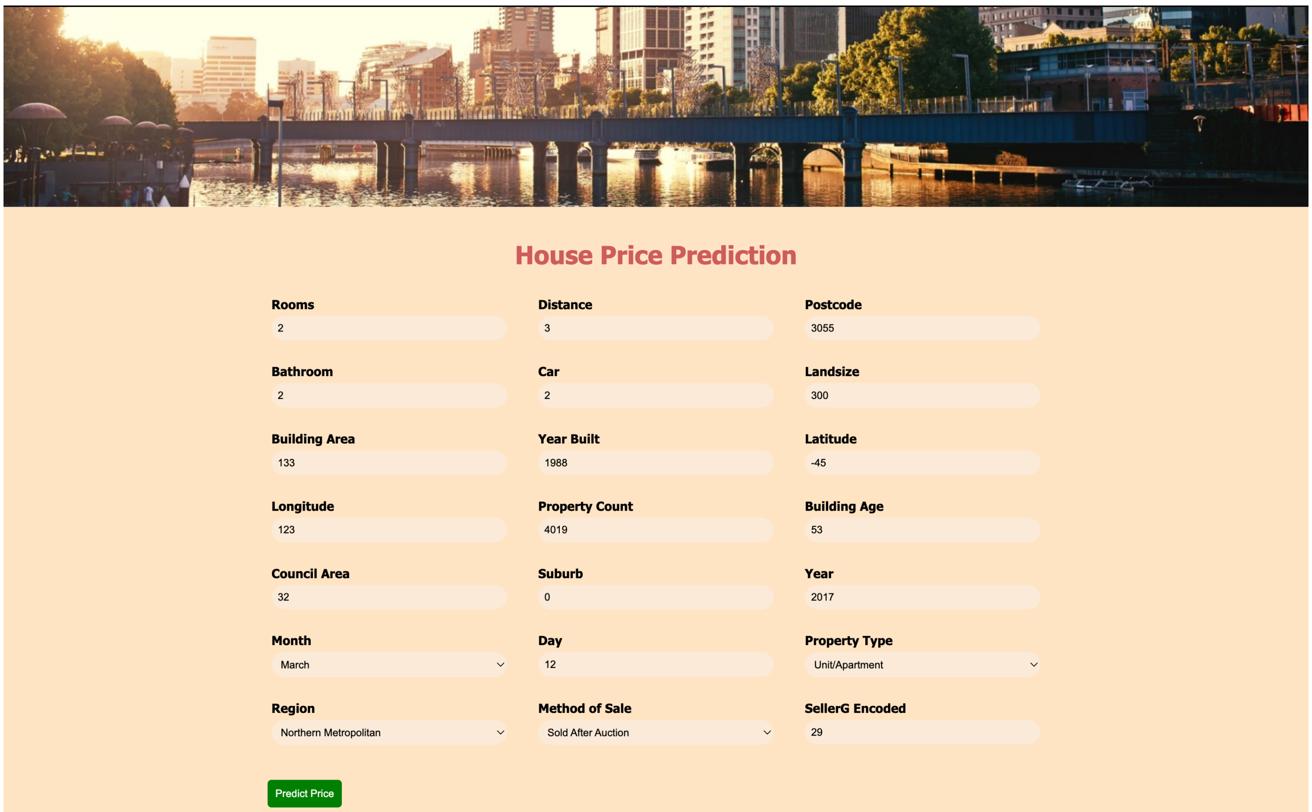
Form Submission Button: Clicking the "Predict Price" button submits the form data for prediction.

Prediction Result Display: The predicted price, received from the backend, is displayed after the prediction button is clicked.

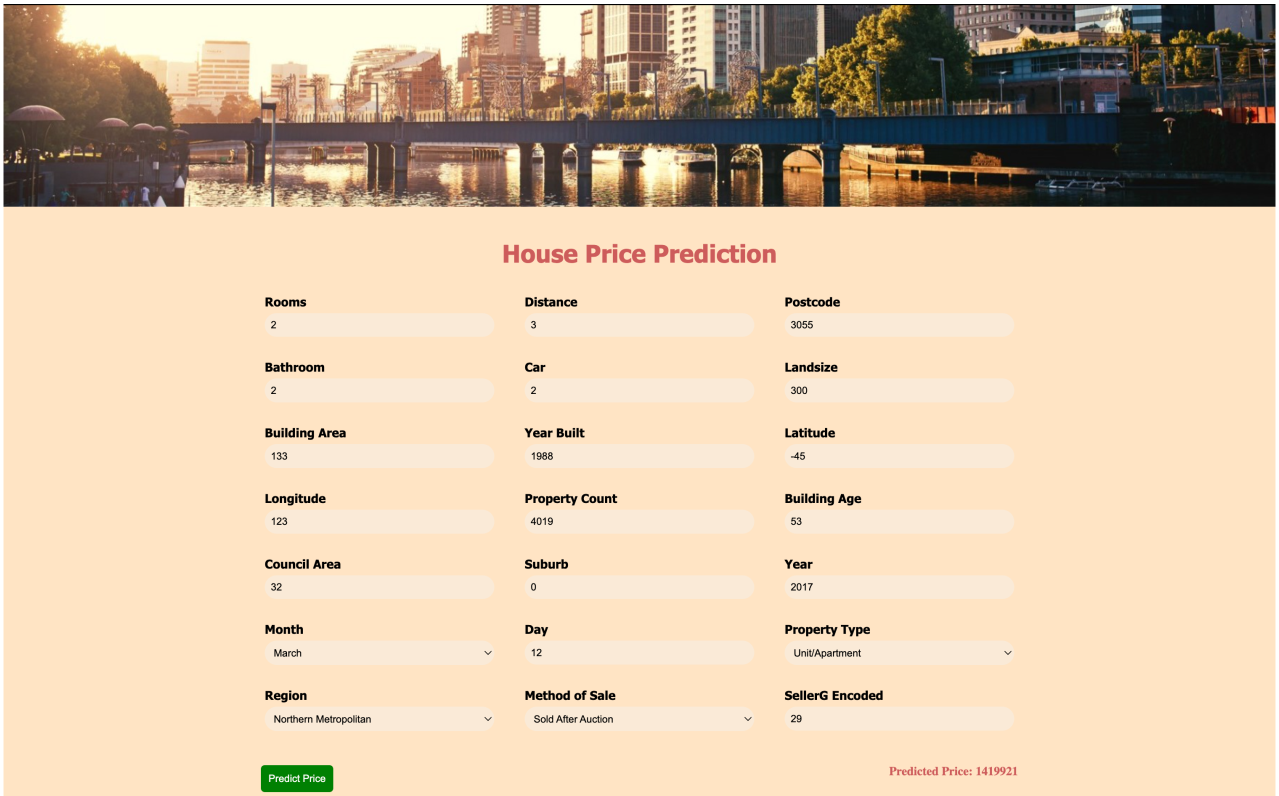
Graphs (Hidden by Default): Upon prediction, two bar charts appear displaying:

- Input Field Values: A visual representation of the user-entered property details.
- Predicted Prices: A chart showcasing the predicted house price derived from the inputs.

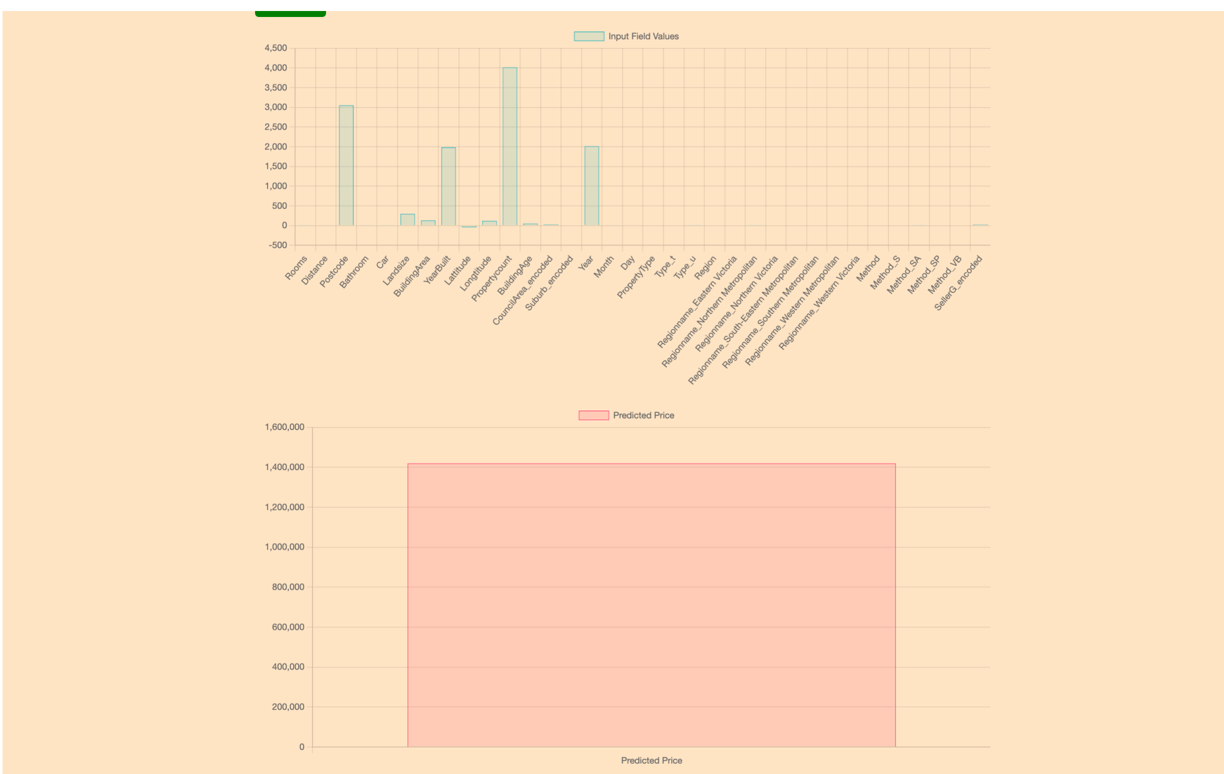
Now, when a user inputs the data into the respective fields, the web application will be successfully able to predict the output. Consider the sample input data entered as follows:



By clicking on Predict Price Button, the output is displayed as follows:



This predicted price serves as an estimated value based on the input factors provided by the user. It's not an absolute value but rather an estimation derived from the model's understanding of historical data patterns.



We can also view visualizations generated based on the input field values and the predicted price in the above screenshot. The bar charts depict the different parameters entered and the predicted house price.

Real-World Applicability

The developed solution enhances transparency in the real estate market, fostering better decision-making for stakeholders. It serves as a reliable tool for pricing estimations and offers a data-driven approach for property valuation.

Users:

- Users can learn about the critical factors influencing real estate prices in Melbourne City through the models' predictions.
- Understanding how various attributes impact property values can enhance users' knowledge of the housing market dynamics.

Problem-Solving Benefits:

- **Sellers**: Gain insights into factors affecting property prices to optimize their selling strategies.
- **Buyers**: Make informed decisions based on predicted property prices and associated factors.
- **Investors**: Identify potentially profitable real estate opportunities in the market.

Graphs Usage:
By using the graphs, users can manipulate or filter the input fields to observe changes in the visualizations and predict prices dynamically. This interactive feature enables users to explore 'what-if' scenarios. For instance, they can see how adding an extra room or changing the property's location impacts its estimated value. It assists in making informed decisions by understanding the sensitivity of different factors in price determination.

Project Extensions and Future Enhancement

- The project can be extended to include more diverse datasets or additional features for better predictive accuracy.
- The project can be added with ensemble methods or advanced techniques to further improve model performance.
- Expanding the project scope to cover other cities or regions, providing a broader perspective on real estate markets.
- Exploring the integration of real-time data and external factors (economic indicators, policy changes, etc.) for more dynamic predictions.
- Can introduce new functionality of user accounts and authentication mechanisms, allowing users to save searches, track favorite properties, or access personalized recommendations.

Conclusion

Our application stands as a user-friendly platform predicting house prices based on user-provided property details. The potential buyers, sellers, or real estate agents can input property specifics to obtain estimated price ranges. Users manipulate or filter input fields, observing real-time changes in visualizations and predicted prices. You can play around with the web app by changing details. Graphs show how different features affect the estimated price. It's like trying out different scenarios to see how they impact the price.

Our app gives you a starting point for understanding property values better. It's a tool to help buyers, sellers, and agents make smarter decisions about houses.

REFERENCES:

1. https://flask.palletsprojects.com/en/3.0.x/

2. https://pythonprogramminglanguage.com/python-flask-example/

3. https://www.geeksforgeeks.org/css-selectors-complete-reference/?ref=shm

4. https://www.tutorialspoint.com/javascript/javascript_overview.html