**Madhav Institute of Technology and Science, Gwalior**

(Deemed to be University)

NAAC Accredited with A++ Grade

**Centre for Artificial Intelligence**



## Skill Based Mini Project Of

**Data Mining & Warehousing**

**(280601)**

**SUBMITTED BY**

Anand Bhadoriya (0901AM211010)

Aryan Singh (0901AM211015)

Neetesh Jatav (0901AM211034)

6th semester

Artificial Intelligence & Machine Learning (AIML)

JANUARY – JUNE, 2024

**SUBMITTED TO**

Dr. Shubha Mishra

Prof. Gaurisha Sisodiya

**ACKNOWLEDGEMENT**

I would like to express my greatest appreciation to all the individuals who have helped and supported me throughout this lab file. I am thankful to whole Centre for Artificial Intelligence department for their ongoing support during the experiments, from initial advice and provision of contact in the first stages through ongoing advice and encouragement, which led to the finals report of this lab file.

A special acknowledgement goes to my colleagues who help me in completing the file and by exchanging interesting ideas to deal with problems and sharing the experience.

I wish to thank our professor **Dr. Shubha Mishra** as well for her undivided support and interests which inspired me and encouraged me to go my own way without whom I would be unable to complete my project.

At the end, I want to thank my friends who displayed appreciation to my work and motivated me to continue my work.

Anand Bhadoriya (0901AM211010)

Aryan Singh (0901AM211015)

Neetesh Jatav (0901AM211034)

**MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR**

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

***NAAC Accredited with A++ Grade***

**CERTIFICATE**

This is certified that **Anand Bhadoriya (0901AM211010), Aryan Singh (0901AM211015) & Neetesh Jatav (0901AM211034)** has submitted the project report titled **House Price Prediction** under the mentorship of **Dr. Shubha Mishra & Prof. Gaurisha Sisodiya** in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning** from Madhav Institute of Technology and Science, Gwalior.

**Dr. Shubha Mishra Prof. Gaurisha Sisodiya**

Faculty Mentor Faculty Mentor

Centre for Artificial Intelligence Centre for Artificial Intelligence

**Table of Content**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Type of Project** | **Page No.** | **Date** | **Signature** |
| **1.** | Micro Project |  |  |  |
| **2.** | Macro Project |  |  |  |
| **3.** | Mini Project |  |  |  |

# Micro Project

## Micro Project

**Aim:** This project aims to preprocess and visualize a dataset of real estate properties in Mumbai, India, specifically focusing on apartments. The goal is to prepare the data for further analysis, such as building a model to predict property prices.

**Theory :**

**Data Preprocessing:**

1. **Data Loading and Inspection:**
   * The code imports necessary libraries: numpy (numerical computations), pandas (data manipulation), and matplotlib.pyplot (visualization).
   * It reads the CSV data into a DataFrame named data using pd.read\_csv.
   * The code then explores the data using data.shape (dimensions), data.info() (data types and missing values), and data.isna().sum() (total missing values in each column).
2. **Feature Selection and Cleaning:**
   * The code drops irrelevant or potentially redundant features:
     + Unnamed: 0 (likely an index column)
     + Amenities like Gymnasium, Lift Available, etc. (may be relevant depending on the specific analysis)
   * It checks for missing values using data.isna().sum(). While the ratings suggest potential handling of missing values (e.g., imputation), the provided code doesn't explicitly address this step.
3. **Feature Engineering:**
   * The code calculates a new feature, price\_per\_sq\_feet, by dividing Price by Area. This creates a more interpretable measure of price.
4. **Outlier Removal (Location-Specific):**
   * The code identifies locations with fewer than 5 properties (location\_count\_less\_than\_5).
   * It replaces location names in those groups with "other" to reduce the number of unique locations (potentially simplifying encoding later).
   * It defines a function remove\_outlier\_sqfeet to detect outliers in price\_per\_sq\_feet for each location group. Outliers are defined as values outside one standard deviation from the mean.
   * The function creates a new DataFrame data\_output containing only non-outliers from each location group.
   * The original data DataFrame is updated with data\_output.
   * The ratings suggest considering alternative outlier detection methods (e.g., IQR-based), which might be more robust for skewed distributions.
5. **Data Cleaning and Saving:**
   * The code drops the price\_per\_sq\_feet column as it was likely intended for temporary use during outlier detection.
   * It saves the cleaned data to a new CSV file named "Cleaned\_data.csv" using data.to\_csv.

**Data Visualization (Exploratory Data Analysis - EDA):**

1. **Price Distribution Histogram:**
   * The code uses plt.hist to create a histogram of the Price distribution, providing insights into the price range and potential skewness.
   * It sets labels and a title for clarity.
2. **Price vs. Area Scatter Plot:**
   * The code creates a scatter plot using plt.scatter to visualize the relationship between Price and Area. This can help identify potential trends or correlations.
   * Labels and a title are added for better understanding.
3. **Price by Number of Bedrooms Box Plot:**
   * The code imports seaborn for advanced visualization.
   * It selects numeric columns using data.select\_dtypes(include=[np.number]).
   * It calculates the correlation matrix (corr) to understand relationships between numeric features.
   * A correlation heatmap is created using sns.heatmap to visually represent these correlations. The heatmap helps in identifying potentially redundant or highly correlated features (a step not explicitly shown in the provided code but could be explored further).

# Macro Project

## Macro Project

**Aim:**

The aim of this section of the code is to build and implement a **multiple linear regression** model to predict apartment prices in Mumbai, India, based on the pre-processed data.

**Theory:**

Multiple linear regression is a statistical technique that models the relationship between a dependent variable (target variable) and two or more independent variables (features). In this case, the dependent variable is the selling price of the apartment (Price), and the independent variables are the various features of the apartments in the dataset (e.g., Area, No. of Bedrooms, Location).

**The underlying theory assumes a linear relationship between the features and the price.** The model learns a set of coefficients that, when combined linearly with the features of a new apartment, can predict its approximate selling price.

Here's a breakdown of the steps involved in the code:

1. **Feature Selection and Preparation:**
   * The code separates the features (independent variables) stored in x from the target variable (dependent variable), y (price).
   * It splits the data into training and testing sets using train\_test\_split. The training set is used to build the model, while the testing set is used to evaluate its performance on unseen data.
2. **Model Building:**
   * The code defines a pipeline using make\_pipeline. This pipeline combines multiple steps:
     + OneHotEncoder: Handles categorical features like Location by converting them into numerical representations suitable for the model.
     + StandardScaler: Standardizes numerical features (e.g., Area, No. of Bedrooms) by scaling them to have a mean of 0 and a standard deviation of 1. This can improve the model's convergence and stability.
     + LinearRegression: Implements the multiple linear regression model to learn the relationship between features and price.
3. **Model Training:**
   * The pipeline (pipe) is fit on the training data (x\_train, y\_train) using pipe.fit. This trains the model to learn the coefficients of the linear equation that best predicts price based on the features.
4. **Model Evaluation:**
   * The trained model is used to predict prices on the testing data (x\_test) using pipe.predict.
   * The r2\_score function evaluates the model's performance by calculating the R-squared value. R-squared represents the proportion of variance in the target variable (price) explained by the model. A higher R-squared value indicates a better fit.
5. **Model Prediction:**
   * The code demonstrates how to use the trained model for prediction on new, unseen data (new\_data). The predicted price is obtained using pipe.predict.

# Mini Project

## Mini Project

* **Aim –** The project aims to implementation in web application for the House Price Prediction.
* **Theory –**

1. **Machine Learning Model:**

**Regression Analysis:** House price prediction is a regression problem where the aim is to predict a continuous output (price). The machine learning model uses regression techniques to analyze and infer the relationship between input features (like area, number

of bedrooms, location, etc.) and the output price.

**Model Selection:** The choice of using a machine learning framework such as sklearn is justified by their flexibility in handling various layers, and large datasets, which are typical in real estate price prediction.

1. **Flask Web Framework:**

**Flask Overview:** Flask is a micro web framework for Python, chosen for its simplicity and efficiency in building web applications. It is particularly suitable for small to medium applications and prototyping.

**Routing and View Functions:** Flask uses decorators to link functions to URLs, making it straightforward to create web pages that interact with the user. This feature is essential for taking user inputs and displaying predictions.

1. **User Interface with HTML:**

**HTML Forms:** HTML forms are used to collect user inputs, which are crucial for making predictions. The ease of integrating HTML with Python/Flask and updating it dynamically makes it an excellent choice for front-end development.

**Responsiveness and Accessibility:** The design considerations ensure that the web application is user-friendly and accessible across different devices and browsers. This inclusiveness enhances user interaction and satisfaction.

1. **Data Handling and Security:**

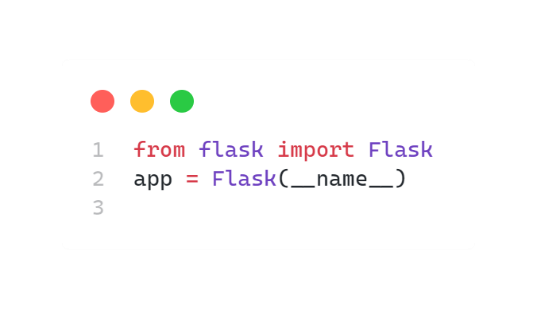
**Data Validation:** Before making predictions, it is critical to validate and preprocess user inputs to avoid errors and improve prediction accuracy. This involves checking for outliers, handling missing values, and ensuring data types are correct.

**Security Measures:** The application must protect sensitive data and prevent common vulnerabilities such as SQL injection and cross-site scripting (XSS). Techniques include validating user inputs, using HTTPS, and setting secure HTTP headers.

1. **Deployment Considerations:**

**Scalability:** Discuss how the Flask application can be scaled using additional tools like Gunicorn or Nginx, especially when deploying to a production environment.

**Performance Optimization:** Techniques such as caching, load balancing, and asynchronous processing are important to ensure the application handles multiple requests efficiently.

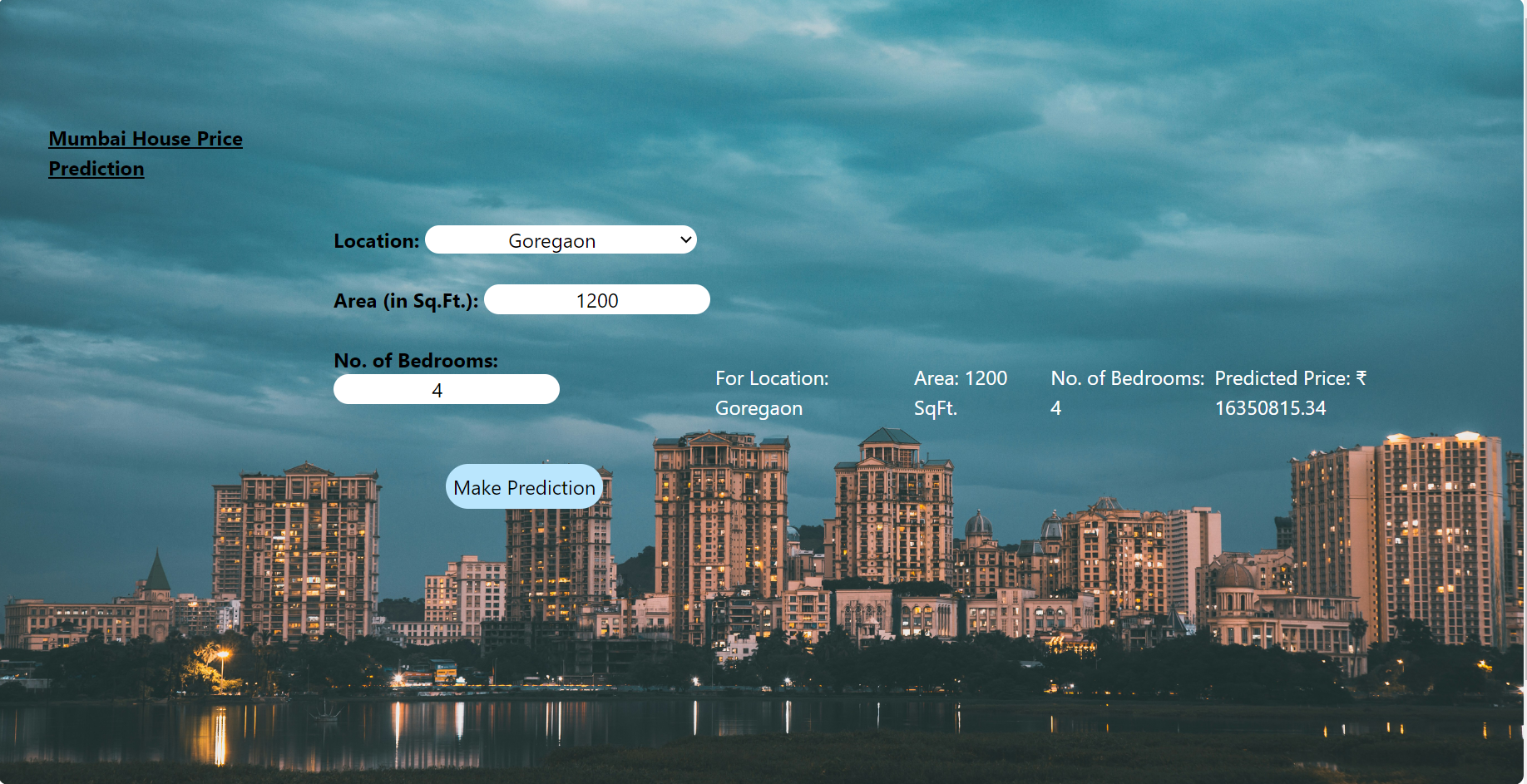
**Implementation**

**1. Setting up Flask**

**2. Backend**

**3. Frontend**

1. **Results**



**Example Use Case:**

An example use case in the theory section can describe a scenario where a user inputs several house features into the web form and receives the predicted price. The discussion can elaborate on how the machine learning model processes these inputs using the trained weights and biases to compute the output, demonstrating the practical application of theoretical concepts like neural networks and regression analysis.

**Conclusion:**

In conclusion, the theoretical foundation of the Flask web application for house price prediction hinges on understanding machine learning principles, web application architecture, user interface design, data security, and deployment strategies. This comprehensive understanding ensures that the implementation is robust, secure, and user-friendly, bridging the gap between complex data models and practical real-world applications.