COURSE NAME – AML 3104 NEURAL NETWORK AND DEEP LEARNING

PROJECT NAME – BANGALORE HOUSE PRICE PREDICTION MODEL

SUBMITTED BY:

AAKASH PATEL - C0895931

ISHIKA SUKHIJA - C0895302

NIPUN GULATI - C0896042

SHREYA DAWAR - C0895966

PROJECT ASSIGNED ROLES FOR EACH TEAM MEMBER

Dataset Selection – All Team Members

Data Cleaning and Exploration - Aakash Patel and Shreya Dawar

Data Visualization – Ishika Sukhija and Aakash Patel

Feature Engineering – Nipun Gulati

Model Building – Nipun Gulati and Aakash

Model Evaluation – Shreya Dawar

Creating the Model's Pickle File – Shreya Dawar

Streamlit Application – Ishika Sukhija

Deployment: Nipun Gulati and Ishika Sukhija

TABLE OF CONTENT

1. Introduction

Predicting housing prices is a complex task due to the multitude of factors that can influence real estate values. The objective of this project is to develop a robust machine learning model that can accurately predict housing prices in Bengaluru, India. This involves several key steps: data selection, cleaning, exploration, feature engineering, model selection, hyperparameter tuning, evaluation, and deployment. By following a structured approach, we aim to build a reliable predictive model and deploy it as a web application.

2. Dataset Selection

Dataset Overview

The dataset used for this project is "Bengaluru House Data," which contains various features such as location, size, total square feet, number of bathrooms, and balconies, price. This dataset provides a comprehensive overview of the housing market in Bengaluru and is suitable for regression tasks to predict housing prices.

Justification for Dataset Choice

The dataset was chosen due to its relevance and detailed information, which allows for thorough analysis and model training. The inclusion of multiple features enables the exploration of complex relationships and interactions that can impact housing prices.

Loading the Data:

- df = pd.read_csv("Bengaluru_House_Data.csv"): This line reads the CSV file named Bengaluru_House_Data.csv into a Pandas DataFrame df, which will hold the data for further processing and analysis.
- df.head(): This line displays the first five rows of the DataFrame df to give an initial overview of the data structure and contents.

Inspecting Data Structure:

- df.info(): This line provides a concise summary of the DataFrame df, including the number of non-null entries, data types of each column, and memory usage. This is helpful for understanding the overall structure and identifying potential data quality issues.

Location Value Counts:

- df['location'].value_counts(): This line calculates and displays the count of unique values in the 'location' column of the DataFrame df. This is useful for identifying the most common locations in the dataset.

3.Data Cleaning

Data cleaning is a crucial step to ensure the quality and reliability of the data before analysis and model building.

Dropping Unnecessary Columns:

- data = data.drop(columns=['society', 'availability'], axis=1): Removes the 'society' and 'availability' columns from the DataFrame as they are deemed unnecessary for the analysis.
- data.head(1): Displays the first row of the modified DataFrame to verify the changes.

Converting Square Feet to Numerical Values:

- Defines a function convert_sqft_to_num that converts a range or single value in the 'total_sqft' column to a numerical value.
- data.total_sqft = data.total_sqft.apply(convert_sqft_to_num): Applies the conversion function to the 'total_sqft' column.
- data = data[data.total_sqft.notnull()]: Removes rows with null values in the 'total_sqft' column.
- data.head(): Displays the first few rows of the cleaned DataFrame.

Checking for Missing Values:

- data.isnull().sum(): Outputs the number of missing values in each column, identifying columns that may require further cleaning.

Handling Missing Values:

- data = data.dropna(): Removes all rows with any missing values, ensuring a clean dataset for analysis.

Unique Area Types:

- data.area_type.unique(): Lists the unique values in the 'area_type' column, providing an overview of the different types of areas present in the dataset.

Location Counts:

- location_count = data.groupby('location').size().sort_values(ascending=False): Groups the data by location and counts the number of entries for each location, then sorts the counts in descending order.
- location_count: Displays the sorted counts of each location.

Grouping Less Common Locations:

- data.location = data.location.apply(lambda x: 'other' if x in location_less_10 else x): Replaces less common locations with the label 'other' to reduce the number of unique locations.
- len(data.location.unique()): Outputs the new count of unique locations after the replacement.

Converting Size to Numerical Values:

- data['size'] = data['size'].apply(lambda x: int(x.split(' ')[0])): Extracts the numerical part of the 'size' column and converts it to an integer.
- data['size'].unique(): Lists the unique values in the converted 'size' column to verify the transformation.
- data.head(1): Displays the first row of the modified DataFrame to check the changes.

Handling Outliers

Outliers can significantly affect model performance. They were identified using statistical methods such as standard deviation and were removed to reduce their impact.

Visualizing Outliers with Boxplots:

- import seaborn as sns: Imports the Seaborn library for creating statistical visualizations.
- numeric_cols = ['total_sqft', 'bath', 'balcony', 'price']: Defines a list of numerical columns to be visualized.
- plt.figure(figsize=(15, 10)): Sets the size of the plot.
- Creates a series of boxplots for each numerical column, identifying potential outliers.
- plt.tight_layout(): Adjusts the layout for better visualization.
- plt.show(): Displays the boxplots.

Removing Outliers Based on Standard Deviation:

- Calculates the upper and lower limits for the 'price' and 'total_sqft' columns based on one standard deviation from the mean.
- Filters the DataFrame to retain only the rows within these limits, removing outliers.

4. Data Exploration

Data exploration helps in understanding the underlying patterns and distributions in the data.

Statistical Analysis

Basic statistical measures such as mean, median, standard deviation, and range provide insights into the central tendency and variability of the data.

More explanation given in the python file.

5. Visualizations

Various plots were generated to visualize the distribution of features and their relationships.

Visualisation given in the python file mentioned in the end.

# Price Distribution

Understanding the distribution of housing prices helps in identifying skewness and potential outliers.

# Correlation Heatmap

Correlation heatmaps visualize the relationships between numerical features, helping to identify potential predictors of housing prices.

5. Feature Engineering

Feature engineering involves creating new features and transforming existing ones to improve model performance.

Creating New Features

New features were created to enhance the model's performance. For example, price per square foot was calculated as a new feature.

Handling Categorical Variables

Categorical variables were handled using encoding techniques such as one-hot encoding to convert them into a numerical format suitable for machine learning models.

6. Model Selection

Several machine learning models were experimented with to identify the best-performing model for predicting housing prices.

Splitting the Data into Training and Testing Sets:

- X = data.drop(columns=['price']): Defines the feature matrix X by dropping the 'price' column from the DataFrame.
- y = data.price: Defines the target vector y as the 'price' column.
- Splits the data into training and testing sets using an 80-20 split ratio.

Preparing for Model Training:

- Imports necessary libraries for encoding, model training, and evaluation.
- Defines a column transformer column_trans to apply one-hot encoding to categorical columns ('area_type' and 'location'), while passing through the remaining columns unchanged.

8. Model Evaluation

Models were evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Evaluation Metrics

- **MAE** measures the average magnitude of errors in predictions, without considering their direction.

- **MSE** measures the average squared difference between predicted and actual values, penalizing larger errors more heavily.

- **RMSE** is the square root of MSE, providing an error metric in the same units as the target variable.

Training and Evaluating Models:

- Defined a dictionary of models to be evaluated.
- For each model, created a pipeline including the column transformer and the model, then trains the model on the training data.
- Evaluated the model's performance on the test data using the R-squared metric and prints the score.

7. Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing model performance. Techniques such as GridSearchCV and RandomizedSearchCV were employed to find the optimal hyperparameters for each model.

Linear Regression

Linear Regression is a simple yet powerful model for regression tasks. It assumes a linear relationship between input features and the target variable and serves as a baseline model due to its simplicity and interpretability.

Decision Trees

Decision Trees are non-linear models that can capture complex relationships between features. They split the data into subsets based on feature values, making them powerful for modeling interactions.

XGBoost

XGBoost is an advanced gradient boosting technique known for its high performance and efficiency. It builds an ensemble of trees sequentially, where each tree corrects the errors of the previous one.

Stacked Model Training and Evaluation:

- Predicts the test data using individual models (Linear Regression, Decision Tree, Random Forest).
- Stacks the predictions to create meta-features.
- Trains a meta-model (XGBoost) on the meta-features.
- Evaluates the stacked model using the R-squared metric and prints the score.

Visualizing Stacked Model Predictions:

- Creates a scatter plot comparing the actual values to the predicted values from the stacked model.
- Adds a reference line indicating the ideal fit.
- Customizes the plot with titles, labels, legend, and grid, and displays the plot.

Training the Pipeline with Stacked Model:

- Creates a pipeline including the column transformer and the stacked meta-model.
- Trains the pipeline on the training data.

Making Predictions:

- Uses the trained pipeline to predict the price for a new data instance.
- pipe.predict(pd.DataFrame([...])): Passes a new data instance to the pipeline for prediction and outputs the predicted price.

9. Saving the Models

The trained models were saved using the pickle library for easy loading and deployment. This ensures that the models can be reused without retraining.

Saving the Model:

- Imports the pickle library for serialization.
- Saves the trained meta-model to a file named 'model.pkl' using pickle, allowing for later reuse.

10. Streamlit Application

A Streamlit application was created to provide a user interface for making predictions. The application allows users to input relevant features and receive predictions based on the trained models.

LINK FOR THE STREAMLIT APPLICATION IS AS FOLLOWS:

11. Conclusion

This project demonstrated the end-to-end process of predicting housing prices in Bengaluru using various machine learning models. The steps included data cleaning, exploration, feature engineering, model selection and tuning, and deployment of a Streamlit application for predictions. The XGBoost model showed superior performance compared to Linear Regression, highlighting the importance of trying different models and tuning their hyperparameters.

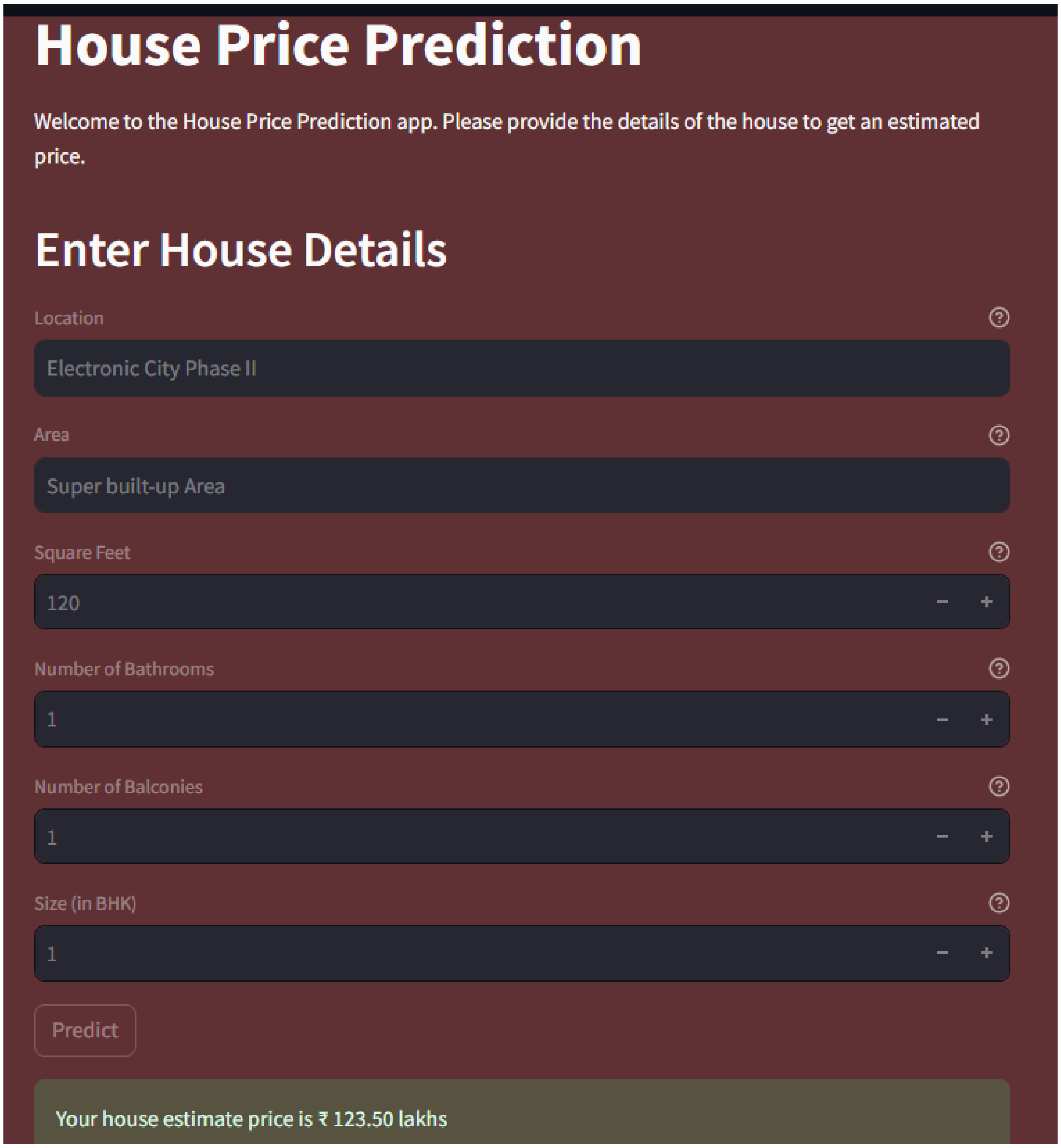
This report provides a comprehensive overview of the housing price prediction project, detailing each step and the rationale behind it. It includes theoretical explanations and justifications for the chosen methods and techniques, ensuring a thorough understanding of the project workflow.

PYTHON FILE FOR THE WHOLE PROJECT IS:

Bengaluru_House_Pric
e_Prediction_Grp7.pdf

streamlit.pdf

Screenshot of UI:



House Price Prediction

Welcome to the House Price Prediction app. Please provide the details of the house to get an estimated price.

Enter House Details

Location: Electronic City Phase II

Area: Super built-up Area

Square Feet: 120

Number of Bathrooms: 1

Number of Balconies: 1

Size (in BHK): 1

Your house estimate price is ₹ 123.50 lakhs

url: https://mr-thispc.herokuapp.com/