House Price Prediction Project Report

1. Introduction

The House Price Prediction project aims to develop a machine learning model that accurately predicts house prices based on various features such as locality, number of bathrooms, furnishing status, transaction type, and more. The dataset used consists of real estate data from Delhi, India. This report covers the data preprocessing, feature engineering, model selection, training, evaluation, and deployment.

2. Problem Statement

Predicting house prices is a critical aspect of the real estate market. Buyers and sellers need reliable price estimations to make informed decisions. The objective of this project is to build a predictive model using machine learning techniques to estimate house prices based on given attributes.

3. Data Collection and Preprocessing

3.1 Dataset Overview

The dataset used for this project contains the following features:

- Locality (Categorical)
- Type (Builder Floor, Apartment, etc.)
- Transaction (Resale or New_Property)
- Furnishing (Furnished, Semi-Furnished, Unfurnished)
- Number of Bathrooms (Numeric)
- Per Square Foot Price (Numeric)
- **Total Price (Target Variable)**

3.2 Handling Missing Values

- Dropped the 'Status' column as it was not contributing to predictions.
- Replaced missing values in 'Bathroom' and 'Per_Sqft' columns with their mean values.
- Set missing values in 'Parking' to 0 (indicating no parking available).
- Replaced missing values in 'Furnishing' with 'Unfurnished' and in 'Type' with 'Apartment'.

```
df.isnull().sum()
```

	0
Area	0
BHK	0
Bathroom	2
Furnishing	5
Locality	0
Parking	33
Price	0
Status	0
Transaction	0
Type	5
Per_Sqft	241

dtype: int64

(A table showing missing values before and after handling them)

3.3 Handling Duplicates

- Identified and removed duplicate rows from the dataset.

3.4 Feature Engineering

- **One-Hot Encoding:**
    - Converted 'Type' and 'Transaction' into binary columns using one-hot encoding.

      'Type':

      - Builder_Floor = 1
      - Apartment = 0

      'Transaction':-

      - Resale = 1
      - New_Property = 0

- **Label Encoding:**
    - Encoded 'Furnishing' into numeric values

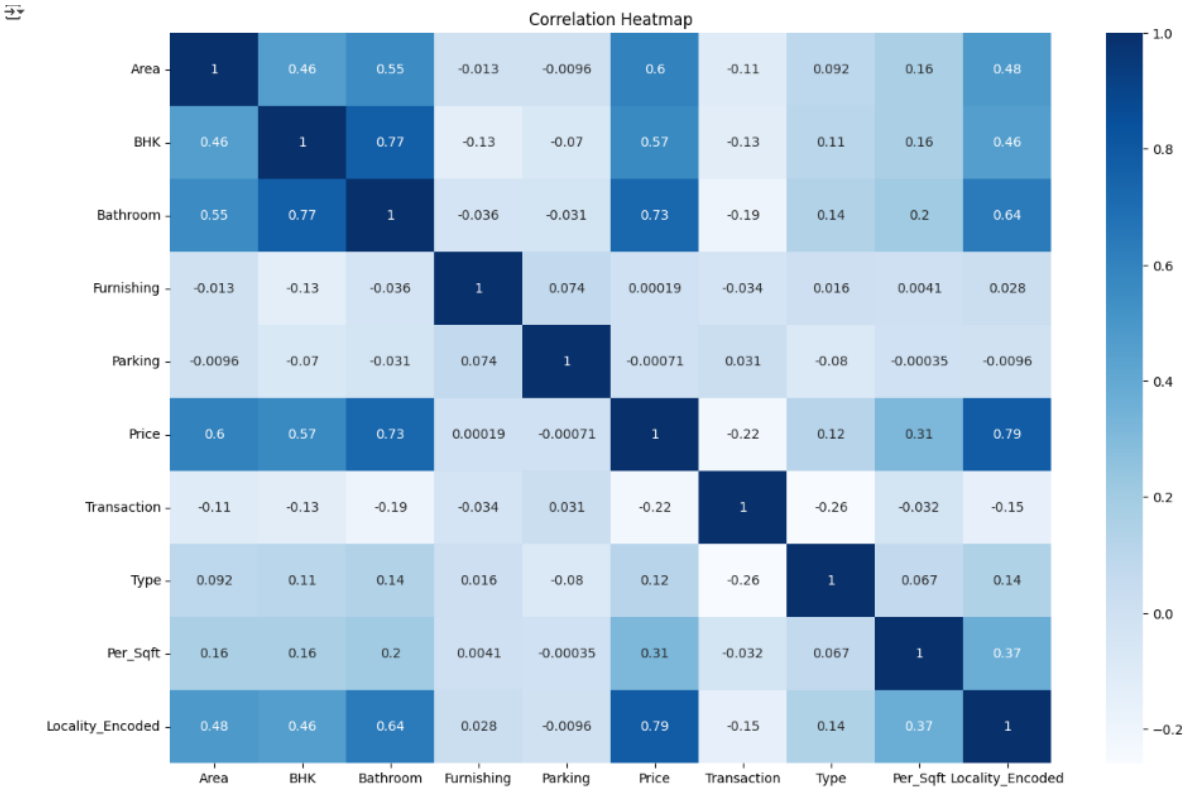
      (0: Semi-Furnished, 1: Unfurnished, 2: Furnished).
- **Locality Processing:**
    - Cleaned locality names using regex.
    - Computed and assigned the mean price per locality to a new column 'Locality_Encoded'.

	Locality	Cleaned_Locality
0	Rohini Sector 25	Rohini Sector 25
1	J R Designers Floors, Rohini Sector 24	Rohini Sector 24
2	Citizen Apartment, Rohini Sector 13	Citizen Apartment
3	Rohini Sector 24	Rohini Sector 24
4	Rohini Sector 24 carpet area 650 sqft status R...	Rohini Sector 24

(data in tabular form of house prices across different localities)

4. Exploratory Data Analysis (EDA)

- Visualized correlations between features using a heatmap.
- Checked price distributions
- Skipping outliers as each house has its own speciality so making changes according to outlier may misinterpret the data.
- Identified the most influential features using mutual information scores.



(Correlation heatmap)

5. <u>Model Selection and Training</u>

5.1 Machine Learning Models Used

- **Linear Regression**

- **Random Forest Regressor**
- **XGBoost Regressor**

```
# Define the Models

model_selection = {

'Linear Regression': LinearRegression(),

'Random Forest': RandomForestRegressor(),

'XGBoost': xgb.XGBRegressor()

}
```

5.2 Splitting Data

- The dataset was split into 80% training and 20% testing.

```
# Split the data into feature and target

X = df.drop(columns=['Price'])

y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

5.3 Hyperparameter Tuning

- **Used GridSearchCV to optimize the Random Forest model with the following parameter grid:**

```
param_grid = {

    "n_estimators": [50, 100, 200],    # Reduced range to avoid overfitting

    "max_depth": [3, 5, 10, None],     # Added smaller depth to prevent
overfitting

    "min_samples_split": [2, 5, 10],  # Keeping key values

    "min_samples_leaf": [1, 2, 4, 6], # Added slightly higher value

}
```

(Insert Image: A line graph showing model performance at different hyperparameters)

6. Model Evaluation

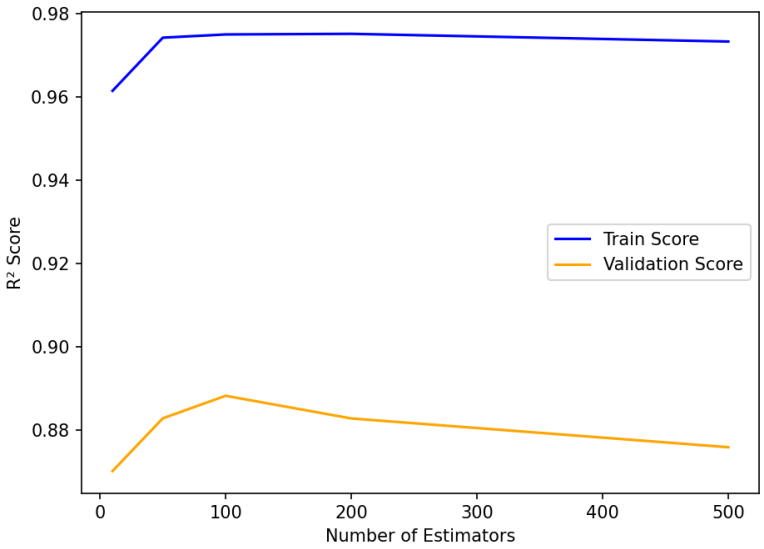
6.1 Performance Metrics

Model	Mean Absolute Error	Root Mean Squared Error	R^2 Score
Linear Regression	7.377646e+06	1.791716e+07	0.651876
Random Forest	5.039305e+06	1.070421e+07	0.875748
XGBoost	4.566850e+06	8.193241e+06	0.927204

(MAE, RMSE, and R^2 for different models)

6.2 Learning Curve

- **A learning curve was plotted to detect overfitting or underfitting.**



(Learning curve showing train vs. validation scores)

7. Model Deployment

7.1 Flask API

- **Developed a Flask API for real-time predictions.**
- **The API receives JSON input, preprocesses the data, and returns a predicted house price.**

7.2 Key Functions in API

```python
# Flask App
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        input_data = request.get_json()  # Get input data
        input_arr = preprocess_input(input_data)  # Preprocess input
        prediction = model.predict(input_arr)  # Make Prediction
        return jsonify({"Predicted House Price": round(float(prediction[0]), 2)})
    except Exception as e:
        return jsonify({"Error": str(e)})

if __name__ == '__main__':
    app.run(debug=True)
```

7.3 Common Errors and Fixes

- **Addressed an error:** 'RandomForestRegressor' object has no attribute 'transform'.
- **Solution:** Removed unnecessary transformation steps in Flask API.

8. Conclusion

- **A XGBoost model was selected as the best model.**
- **The model achieved high accuracy with an R^2 score of 92.72%.**
- **The deployment using Flask API allows real-time house price predictions.**

9. Future Improvements

- **Incorporating more features such as crime rate, school ratings, and amenities.**
- **Improving performance by tuning advanced models like LightGBM.**
- **Deploying the model as a web application for broader accessibility.**

10. References

- **Dataset Source: Delhi House Data from Kaggel**
- **Scikit-learn Documentation: https://scikit-learn.org**
- **Flask Documentation: https://flask.palletsprojects.com**
