## REPORT TEMPLATE – House Price Prediction

### Title:

**End-to-End Predictive Modeling for House Price Forecasting**

---

### Prepared By:

*Anshika Dwivedi*
**Role:** ML Intern
**Submission Date:** [10-Nov-2025]

---

## 1. Problem Statement

The objective of this project is to develop a machine learning model capable of predicting **house prices** based on historical and auxiliary data.
The model should not only achieve accurate numerical predictions but also provide **insights into which factors most influence the outcome**, making it both predictive and interpretable.

**Example:**

- *House Price Prediction:* Estimate property prices based on features like area, location, rooms, and amenities.

---

## 2. Dataset Description

The dataset was obtained from **Kaggle**.
It contains multi-dimensional information about houses along with temporal and regional features.

**Example (House Price):**

| Feature | Description |
|---|---|
| LotArea | Total area of the property |
| OverallQual | Overall material and finish quality |

| Feature | Description |
|---|---|
| Date House was Sold | Year when the house was built |
| SalePrice | Actual selling price (target variable) |
| Total land size | Lot Area (in Sqft) |

---

### 3. Data Preprocessing

The following preprocessing steps were performed:

1. **Handled missing values:**
   - Replaced missing numerical values with mean/median.
   - Encoded categorical variables using LabelEncoder
2. **Outlier detection:**
   - Used box plots and IQR method to identify and treat outliers.
3. **Scaling:**
   - Applied StandardScaler to normalize features (especially for regression models).
4. **Feature selection:**
   - Removed irrelevant columns (e.g., IDs, non-informative data).

**Code snippet example:**

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()

scaled_features = scaler.fit_transform(X)
```

## 4. Feature Engineering

Feature engineering was a crucial step to improve model accuracy and capture temporal patterns.

**Examples:**

- 'Property_Quality_Index‘ : flat area (in sqft)'] *'overall grade' * 'condition_numeric' / ['age of house (in years)'] + 1
- 'Structure_Index‘ : ['no of bedrooms'] + ['no of bathrooms'] + ['no of floors'] + ['basement area (in sqft)'] / 1000)]

External indicators (like Facilities near house and crime_rate in that particular pincode).

## 5. Model Development

Several regression models were trained and compared:

| Model | Description | R² Score | RMSE | MAE |
|---|---|---|---|---|
| Linear Regression | Baseline model | 0.79 | 112553.41 | 83873.55 |
| Lasso Regression | L1 regularization to remove noise | 0.79 | 112553.39 | 83873.57 |
| Ridge Regression | L2 regularization to stabilize coefficients | 0.79 | 11252.78 | 83875.08 |
| GradientBoostingRegressor | Tree Based Model | 0.88 | 84974.73 | 59549.74 |
| XGBOOST Regressor | Tree Based Model | 0.88 | 84226.46 | 59355.94 |

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso

from xgboost import XGBRegressor

from sklearn.ensemble import GradientBoostingRegressor


models = {     "Linear Regression": LinearRegression(),     "Ridge Regression": Ridge(alpha=1.0),
"Lasso Regression": Lasso(alpha=0.1)
"Gradient Boosting Regressor": GradientBoostingRegressor(n_estimators=300),"XGBoost
Regressor": XGBRegressor(n_estimators=300 )

}

for name, model in models.items(): model.fit(X_train, y_train)

    preds = model.predict(X_test)

    print(name, mean_absolute_error(y_test, preds))
```

## 6. Evaluation Metrics

Performance was evaluated using:

- **RMSE (Root Mean Squared Error)**
- **MAE (Mean Absolute Error)**
- **R² Score (Goodness of Fit)**

All five regression models were evaluated using **RMSE**, **MAE**, and **R² Score** to measure their prediction accuracy and error magnitude.
Linear, Ridge, and Lasso models performed similarly due to the linear nature of their learning, achieving an R² of around **0.79**.
Tree-based models—**Gradient Boosting** and **XGBoost**—showed significantly better performance, with XGBoost achieving the **highest accuracy** (R² ≈ **0.883**) and the **lowest errors** (RMSE ≈ **85k**, MAE ≈ **60k**).
Based on these evaluation metrics, **XGBoost is the most reliable and accurate model**, capable of capturing complex, non-linear relationships in house pricing data.

### 7. Feature Importance and Explainability

To interpret model decisions, **SHAP** was used. import

```
shap

explainer = shap.Explainer(model, X_test)

shap_values = explainer(X_test)

shap.summary_plot(shap_values, X_test)
```

Insights:

- For house prices, **rolling_price5_zip**, **Area of the house**, and **Living area after renovation** were most important.

---

### 8. Results and Insights

- The **XGBoost model** gave the best results due to its ability to capture non-linear relationships.
- Regularized models like **Ridge** and **Lasso** reduced overfitting.
- Feature importance analysis showed [rolling_price5_zip, Area of the house, and Living area after renovation] as the most impactful.

**Visualization Example:**

```
plt.figure(figsize=(10,5))
plt.plot(y_test.values, label="Actual")
plt.plot(y_pred, label="Predicted")

plt.legend()
plt.title("Actual vs Predicted Prices")
plt.show()
```

---

### 9. Limitations

1. Model doesn’t account for sudden market or economic changes.
2. Limited dataset size reduces generalization.

3. External data integration (crime_rate) can improve accuracy.

---

### 10. Future Scope

- Integrate **real-time data APIs** for house price updates.
- Deploy as a **web dashboard** using Flask or Streamlit.

---

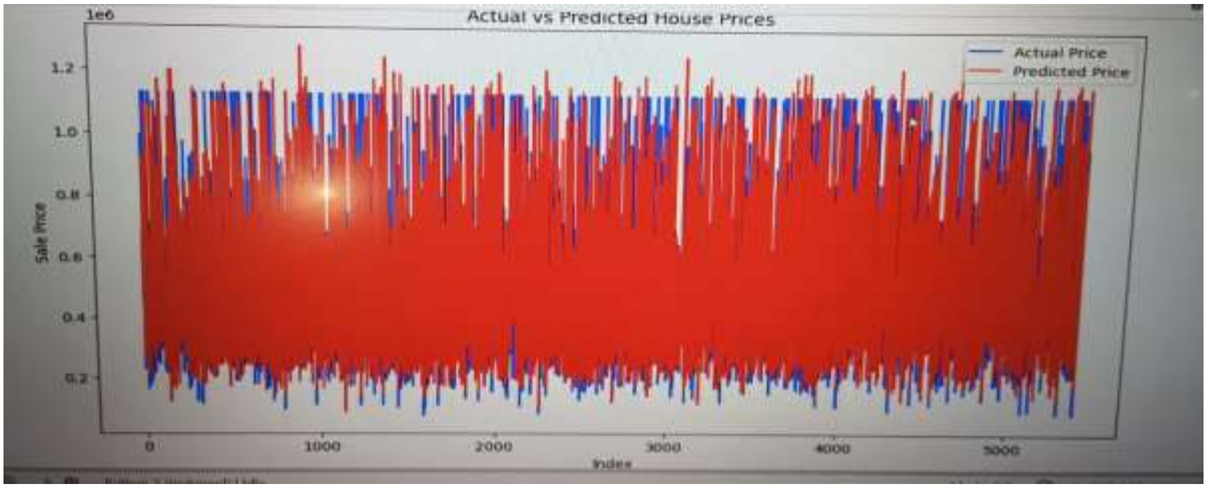
### 11. Conclusion

This project demonstrates an end-to-end machine learning workflow — from data collection and preprocessing to feature engineering, model building, and explainability.

The model achieved strong performance and provided interpretable insights into the factors influencing price movements.

Future work will focus on improving temporal modeling and integrating richer contextual data sources.

## Actual Vs predicted House Price Visualization


Actual vs Predicted House Prices
1e6
Actual Price
Predicted Price
Sale Price
Index