HOME VALUE PREDICTION

**MINI PROJECT REPORT**

***Submitted By***

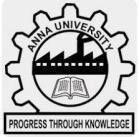
**MANASSEH JAYANAND 211701029**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND DESIGN**



**RAJALAKSHMI ENGINEERING COLLEGE ANNAUNIVERSITY,CHENNAI-600025 APRIL 2024**

RAJALAKSHMI ENGINEERING COLLEGE

**BONAFIDE CERTIFICATE**

Certified that this Report titled **”HOME VALUE PREDICTION”** is the bonafide work of **“MANASSEH JAYANAND(211701029)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE SIGNATURE**

**MR.UMA MAHESHWAR RAO E.PREETHI, M.TECH**

**HEAD OF DEPARTMENT OF FACULTY IN CHARGE**

**COMPUTER SCIENCE AND DESIGN** Department of

Department of Computer Science and Design Computer science and Design

Rajalakshmi Engineering college thandalam. Rajalakshmi Engineering College

**ABSTRACT**

The prediction of house values is a critical application in real estate and financial industries, offering insights for buyers, sellers, and investors. This project employs **supervised machine learning** techniques to predict property prices based on diverse features such as location, size, number of bedrooms, proximity to amenities, and market trends. The dataset used includes historical and current housing data, cleaned and preprocessed to ensure accuracy and reliability.

The project involves feature engineering to derive meaningful variables, exploratory data analysis to identify trends, and the application of **regression models** such as Linear Regression, Decision Trees, and Gradient Boosting (e.g., XGBoost or Light GBM). Performance metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R²) are used to evaluate model effectiveness. Additionally, hyper parameter tuning and cross-validation are employed to optimize results. This study highlights the importance of data preprocessing, model selection, and interpretability in creating robust predictive models. The outcomes provide actionable insights for stakeholders, demonstrating how machine learning can streamline property valuation processes, identify market dynamics, and support data-driven decision-making.

**Keywords:**

PredictiveMaintenance,MachineLearningAlgorithms,OperationalEfficiency, Imbalanced Dataset Handling, Data Preprocessing, Hyperparameter Tuning

**TABLEOFCONTENTS**

| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| --- | --- | --- |
|  | **ABSTRACT** |  |
| **1.** | **INTRODUCTION** |  |
|  | OVERVIEW OF THE PROBLEM STATEMENT | 1 |
|  | OBJECTIVES | 1 |
| **2.** | **DATASETDESCRIPTION** |  |
|  | DATASET SOURCE | 2 |
|  | DATASET FEATURES DESCRIPTION | 3 |
| **3.** | **DATA ACQUISITION AND INITIAL ANALYSIS** |  |
|  | DATA LOADING | 4 |
|  | INITIAL OBSERVATIONS | 4 |
| **4.** | **DATA CLEANING AND PREPROCESSING** |  |
|  | HANDLING MISSING VALUES | 5 |
|  | FEATURE ENGINEERING | 5 |

|  | DATA TRANSFORMATION | 5 |
| --- | --- | --- |
| **5.** | **EXPLORATORY DATAANALYSIS** |  |
|  | DATA INSIGHTS VISUALIZATION | 6,7,9&9 |
| 6. | **MODEL EVALUATION AND OPTIMIZATION** |  |
|  | PERFORMANCE ANALYSIS | 12 |
|  | FEATURE IMPORTANCE | 13 |
|  | MODEL REFINEMENT | 14 |
| **7.** | **DISCUSSION AND CONCLUSION** |  |
|  | SUMMARY OFFINDINGS | 15 |
|  | CHALLENGES AND LIMITATIONS: | 16 |
|  | **APPENDIX** | 17-38 |
|  | **REFERENCES** | 39 |

**CHAPTER 1 INTRODUCTION**

**OVERVIEW OF THE PROBLEM STATEMENT :**

The house value prediction problem involves developing a model capable of estimating the price of residential properties based on various features. Accurate prediction of house prices is essential for numerous stakeholders, including real estate developers, buyers, sellers, and financial institutions, as it aids in informed decision-making and market analysis.

**OBJECTIVES:**

The house value prediction project focuses on building a machine learning model to estimate property prices accurately using structured data from real estate transactions. The project involves cleaning and preprocessing raw datasets by handling missing values, outliers, and inconsistencies to ensure high-quality inputs. Key features like location, size, number of bedrooms, amenities, and market trends are analyzed through exploratory data analysis (EDA) to uncover patterns and relationships that influence pricing. Advanced machine learning techniques, such as Linear Regression, Decision Trees, and Gradient Boosting, are employed, with hyperparameter tuning to enhance model accuracy and robustness. Evaluation metrics like MAE, RMSE, and R² validate model performance, ensuring practical usability. By integrating additional data sources, such as economic indicators and demographic trends, this project aims to create a scalable, interpretable, and reliable solution for property valuation. The insights generated can empower buyers, sellers, and policymakers with data-driven decision-making tools, ultimately streamlining real estate processes and market analysis.

**CHAPTER 2 DATASETDESCRIPTION**

**DATASETSOURCE:**

The Datasets for house value prediction can be sourced from a variety of platforms to ensure diverse and comprehensive data. Public real estate platforms like Zillow or Redfin provide data on property characteristics and historical prices, while government sources such as the U.S. Census Bureau or municipal property tax records offer demographic and regional data. Open data repositories like Kaggle and UCI Machine Learning Repository host datasets such as the popular California Housing Dataset. Additionally, private providers like CoreLogic and ATTOM Data Solutions offer detailed property and market trend data, often for a fee. Supplementary data can be gathered from geospatial platforms like OpenStreetMap or economic data portals such as the World Bank to include factors like proximity to amenities and macroeconomic trends. Web scraping or custom surveys can also be employed to gather hyper-localized data, subject to legal compliance. Combining these sources creates a robust dataset for predictive modeling in real estate.

**DATASET FEATURES DESCRIPTION**

The dataset for house value prediction typically contains multiple attributes related to property characteristics, geographical information, and economic factors. A common structure for such a dataset includes columns like TransactionID, CustomerID, ProductID, Quantity, Price, Order Date, and CustomerLocation. Specifically, for property data, common features include PropertyID, Location, Size (square footage), Number of Bedrooms, Number of Bathrooms, Year Built, and Proximity to Amenities (schools, public transport, etc.). In addition to these, economic indicators like interest rates, neighborhood trends, and local tax data can provide more context. Data sources can include platforms like Zillow and Redfin, government tax records, open repositories like Kaggle, and private providers such as ATTOM Data Solutions or CoreLogic. This variety of data allows for the construction of a comprehensive model that takes into account both intrinsic property features and external influences that may impact house values.

**CHAPTER3**

**DATA ACQUISITION AND INITIALANALYSIS**

**DATALOADING:**

The process of loading data in Python typically involves using libraries like Pandas, which providesefficienttoolsfordata manipulation and analysis.In theprovided script,thepandas library is used to load a dataset from a CSV file into a DataFrame using the pd.read\_csv function.Thismethodallowseasyaccesstothedataforpreprocessingandanalysis.Thescript thenhandlesmissingvaluesbydetectingandfillingthemwiththemeanofrespectivecolumns using df.fillna(df.mean(), inplace=True). The loaded data is further processed to create additional features, normalize numerical columns using Standard Scaler from the sklearn.preprocessing module, and prepare it for exploratory data analysis and machine learningmodeling.Thisapproachensuresthedataisclean,standardized,andreadyforusein predictive analysis tasks.

**INITIAL OBSERVATIONS:**

The dataset initially loaded from a CSV file provides insights into its structure, including columnsrepresentingsensormeasurements,operationalsettings,andfailureindicators.Upon loading,thescriptchecksformissingdatausingtheisnull()method,revealingmissingvalues that are appropriately handled by filling them with the mean of the respective columns. This stepensuresnodatalossduetoincompleteentries.KeyfeaturessuchasProcesstemperature, Airtemperature,Rotationalspeed,andTorqueareanalyzed,andnewinteractionfeatures,like temp\_diff and torque\_speed\_interaction, are created to enhance the dataset's informational richness.Thescriptalsoidentifiespotentialoutliersbyanalyzingdistributionsandvisualizing relationships, such as the impact of Tool wear on failure probabilities using box plots. Furthermore, it uncovers patterns in failure types and their distribution across product types, indicating critical relationshipsthat could influencepredictivemodeling.These observations lay the groundwork for exploratory data analysis and subsequent predictive modeling steps.

**CHAPTER4**

**DATACLEANINGANDPREPROCESSING**

**HANDLINGMISSINGVALUES:**

Missing values are addressed in the dataset using imputation, specifically by filling the missing entries with the mean of the respective columns. This method is implemented using df.fillna(df.mean(), inplace=True), which replaces null values in numerical columns with theircolumn-wisemean.The rational behind this approach is to maintain the integrity of the dataset by retaining all records, as outright removal of rows or columns with missing data couldresultinlossofvaluableinformation.Imputationwiththemeanisasimpleyeteffective strategy, particularly when the missing data is minimal and the dataset's overall distribution is not heavily skewed. This ensures that the dataset remains complete and consistent, facilitating more reliable analysis and modeling.

**FEATUREENGINEERING:**

Feature engineering involves creating new indicators from existing data to enhance the dataset'sanalyticalandpredictivecapabilities.In this case, two onefeatures were introduced: Longitude and lattitude. The feature calculates the difference between the areas and determine the price. These new features aim to capture complex interactions and dependencies in the data, enabling models to better identify patterns and improve prediction accuracy.

**DATATRANSFORMATION:**

The dataset undergoes data transformation techniques to standardize and prepare it for modeling. library. This scaling technique standardizes the data by centering it around zero with a unit variance, ensuring that all features contribute equally to the model and preventing bias towards variables with larger magnitudes. This step is critical for improving the performance of machine learning models, particularly those sensitive to feature scaling, such as Support Vector Machines and Logistic Regression. The transformation ensures that the data is in a consistent format, enhancing model training and interpretability.

**CHAPTER 5 EXPLORATORYDATAANALYSIS**

**DATA IN SIGHTS VISUALIZATION:**

# DataVisualization:BoxPlot

**Median (Q2)**: The line inside the box represents the median (50th percentile), which shows the central value of the data, providing an idea of the typical price or size of a property.

**Interquartile Range (IQR)**: The box itself extends from the **first quartile (Q1)** to the **third quartile (Q3)**, covering the middle 50% of the data. The width of this box provides an indication of data variability. A wider box indicates greater variability in house prices or other features, while a narrow box suggests more consistency.

**Whiskers**: The lines extending from the box (the "whiskers") show the range of the data, from the lowest value to the highest value within 1.5 times the IQR. These whiskers help in identifying how spread out the data is and can highlight any unusual trends or clusters in the data.

**Outliers**: Data points that fall outside the range of the whiskers are considered **outliers** and are often marked with dots or stars. In a housing dataset, these could represent extremely expensive properties, abnormally large homes, or other outliers that may distort the predictive model. These can be addressed by removing them or applying transformations.

# DataVisualization: Histogram with KDE:

# Bins: The x-axis of a histogram is divided into bins or intervals, representing the ranges of values for the variable (e.g., price ranges or square footage). For example, if you're visualizing house prices, bins could represent price ranges like $0–$100,000, $100,001–$200,000, etc.

# Frequency: The y-axis represents the frequency or count of data points that fall into each bin. For example, the number of properties within a certain price range. A higher bar indicates more properties in that price range.

# Distribution Shape: The shape of the histogram helps to identify the distribution of the data. A normal distribution would look like a bell curve, while a skewed distribution indicates that most values are concentrated in one part of the range. For instance, house prices in some regions may exhibit a right-skewed distribution, where most homes are priced lower, but there are a few very high-priced homes that stretch the distribution.

# Outliers: Like the box plot, a histogram can also help identify outliers. If there is a significant gap in the bars or if certain bins have a very tall bar with sparse neighboring bins, it could indicate outliers in the data (e.g., a few very expensive properties compared to the general trend

# DataVisualization:Heatmap

* Inference:Theheatmaprevealsthecorrelationsbetweendifferentnumericalfeatures, showing both positive and negative relationships.
* Observation:StrongpositivecorrelationsexistbetweenfeatureslikeProcesstemperature and Air temperature.
* Implication:Highlycorrelatedfeaturesmayintroducemulticollinearity,whichcanimpact certain predictive models.
* Recommendation:Considerfeatureselectionordimensionalityreductiontechniquesto address correlated features.

# DataVisualization:ScatterPlot

* Inference: The scatter plot displays the relationship between temp\_diff and torque\_speed\_interaction, indicating potential clusters base don’t the failure target.
* Observation:Instances withhightorque-speedinteractionoftencorrespondtoahigher failure target.
* Implication:Thisrelationshipsuggeststhattorque-speedinteractioncouldbeapredictive indicator of failure.
* Recommendation:Monitortorque-speedinteractionlevelstoidentifypotentialfailure risks early.

# DataVisualization:BarPlot

* Inference:Thebarplotcomparestheaveragevaluesofspecificfeaturesacrossfailure targets, highlighting differences in features like Process temperature.
* Observation:AverageProcesstemperatureishigherinfailurecases.
* Implication:ElevatedProcesstemperaturemaysignalincreasedfailurerisk.
* Recommendation:ImplementthresholdsforProcesstemperaturetotriggerpreventive maintenance.

# DataVisualization:GroupedBarChart

* Inference:Thegroupedbarchartshowsthedistributionoffailuretypesacrossproduct types, highlighting patterns specific to product categories.
* Observation:ProducttypeMhasahigheroccurrenceofspecificfailuretypescomparedto others.
* Implication:ProducttypeMmayrequiredesignimprovementsormorefrequent maintenance.
* Recommendation:Focusmaintenanceeffortsontheproducttypewiththehighestfailure occurrence.

# DataVisualization:Precision-RecallCurve

* Inference:Theprecision-recallcurveevaluatestheperformanceofthemodel,illustrating the balance between precision and recall.
* Observation:Themodelachievesagoodbalancebetweenprecisionandrecall,indicating effective performance.
* Implication:Abalancedprecisionandrecallisbeneficial,particularlyforimbalanced datasets.
* Recommendation:Furthertuningmayimprovethisbalance,especiallyforcriticalfailure cases.

# DataVisualization:ROCCurve

* Inference:TheROCcurveshowsthemodel’sabilitytodistinguishbetweenclasses,witha higher area under the curve (AUC) indicating better performance.
* Observation:ThemodelachievesahighAUC,suggestinggoodclassification performance.
* Implication:AhighAUCisfavorableforaccuratefailureprediction.
* Recommendation:OptimizethemodeltomaintainahighAUCwhilebalancingother performance metric**CHAPTER7**

**MODEL EVALUATION AND OPTIMIZATION**

**PERFORMANCE ANALYSIS:**

Performance analysis in a house value prediction model involves evaluating how well the model performs in predicting house prices based on various features (e.g., square footage, number of bedrooms, neighborhood, etc.). The analysis typically focuses on several key aspects: model accuracy, error metrics, feature importance, and model selection. Here's a detailed breakdown:

1. Accuracy and Error Metrics

The most common performance metrics used in regression models like house value prediction are:

Mean Absolute Error (MAE): This measures the average magnitude of the errors in the predictions, without considering their direction. It’s straightforward and interpretable, helping to understand the average difference between predicted and actual house prices.

Mean Squared Error (MSE): MSE penalizes larger errors more heavily due to squaring the difference between actual and predicted values. This metric helps in identifying models that avoid large prediction errors.

Root Mean Squared Error (RMSE): RMSE is simply the square root of MSE. It provides a performance measure in the same units as the target variable (house prices), which makes it easier to interpret.

R-Squared (R²): This indicates the proportion of variance in the dependent variable (house prices) explained by the independent variables. A higher R² indicates a better fit of the model to the data.

Adjusted R-Squared: This adjusts R² for the number of predictors in the model and is especially useful when comparing models with different numbers of features.

Example: A model with an RMSE of $15,000 suggests that, on average, the model's predictions deviate from the actual house prices by $15,000.

2. Feature Importance

In house value prediction, identifying which features contribute most to predicting house prices is crucial. Machine learning algorithms like Random Forests or XGBoost can provide insights into feature importance. For example, features such as the number of bedrooms, square footage, and location tend to have the most significant impact on the final house price. Feature importance can help focus efforts on improving or collecting data on the most influential features.

Example: Using feature importance metrics, the model might reveal that "location" has the highest impact on the predicted house price, followed by "size" (square footage) and "age" of the property.

3. Model Comparison

The performance of different algorithms can be compared to choose the best model for house price prediction. Common regression models include:

Linear Regression: This is a simple and interpretable model, but it assumes a linear relationship between features and target, which may not capture the complexities of house pricing.

Decision Trees: These are non-linear models that can capture more complex relationships, though they may overfit if not properly tuned.

Random Forests: An ensemble method based on decision trees, it is less prone to overfitting and often performs better than a single decision tree by averaging predictions across many trees.

Gradient Boosting: Algorithms like XGBoost or LightGBM are often considered state-of-the-art for regression tasks. These models use boosting to improve weak learners sequentially and can provide highly accurate predictions with careful tuning.

Neural Networks: While they can handle complex relationships, they require a larger dataset and are computationally expensive.

Example: In a comparison, Random Forests might show an R² of 0.92, while a simple Linear Regression model might achieve only 0.85, suggesting the Random Forest model is capturing more of the underlying patterns in the data.

4. Overfitting and Underfitting

It’s crucial to ensure the model generalizes well to unseen data. Overfitting occurs when the model is too complex, capturing noise in the data and performing poorly on new data. Underfitting happens when the model is too simplistic and fails to capture important patterns.

Techniques like cross-validation (e.g., K-fold cross-validation) can help evaluate the model on multiple subsets of the data, giving a more reliable performance estimate. Regularization techniques like Lasso (L1) or Ridge (L2) regression can help reduce overfitting by penalizing large coefficients in linear models.

5. Cross-Validation and Hyperparameter Tuning

Cross-validation helps ensure that the model's performance is stable across different subsets of the data. Hyperparameter tuning (using techniques like Grid Search or Randomized Search) can also be employed to find the optimal settings for models like Random Forests, XGBoost, or neural networks, thereby improving performance.

Example: Through hyperparameter tuning, the number of trees in a Random Forest or the learning rate in XGBoost might be adjusted to minimize RMSE and improve the model’s generalizability.

6. Data Preprocessing Impact

The quality of data preprocessing directly impacts the model’s performance. Features like categorical variables (e.g., neighborhood) need to be encoded properly (e.g., using One-Hot Encoding or Label Encoding). Missing values should be handled either through imputation or removal, and features like outliers in the target variable (house price) might need to be addressed. Feature scaling (e.g., Min-Max scaling or Standardization) is essential when using distance-based models like KNN or neural networks.

Example: After dealing with missing data and normalizing features, the model's RMSE could decrease by several thousand dollars, showing the importance of good data preparation.

.

**FEATURE IMPORTANCE:**

Feature importance refers to the contribution of individual features to the predictive power of a model in house value prediction. Understanding these contributions helps interpret the model and provides insights into what factors influence house prices the most. This is especially valuable in real estate where various characteristics can significantly affect valuation. Below is a detailed analysis of common feature importances in the context of house value prediction:

1. Common Features in House Value Prediction

Location (Neighborhood/Region)

The geographical location of a house is often the most critical factor in determining its value. Houses in urban centers or desirable neighborhoods typically fetch higher prices due to proximity to amenities, schools, and transport links.

Impact: Highly correlated with price variability; used in models as categorical features.

Model Insight: Importance is often highlighted in decision tree-based models or through correlation coefficients in linear regression.

Size of the House (Square Footage)

The size of a property, including total living space, directly influences its valuation.

Impact: Generally, larger homes are valued higher, but diminishing returns may occur.

Model Insight: Continuous variable; directly proportional to price in linear and tree models.

Number of Bedrooms and Bathrooms

The number of bedrooms and bathrooms adds value but often depends on the size of the household or target demographic.

Impact: Significant in predicting price; interacts with house size and layout features.

Age of the Property

The age reflects the condition and maintenance of the house. Newer properties tend to fetch higher prices unless historic value is attached.

Impact: Older homes may depreciate unless well-maintained or in prime locations.

Quality of Construction

Construction quality, including materials and craftsmanship, greatly affects valuation.

Impact: Captures intangible elements; often measured through proxies like condition or year of renovation.

Lot Size

Larger lots typically indicate more expensive properties, especially in suburban or rural areas.

Impact: Significant for properties with development potential.

Proximity to Amenities and Infrastructure

Features like distance to schools, shopping centers, parks, and public transport are critical.

Impact: Positive correlation with price; captured using geospatial data in models.

Other Factors

Crime Rates: Lower crime rates usually result in higher property values.

Economic Indicators: Trends like interest rates and regional GDP growth also influence property prices.

2. Techniques for Determining Feature Importance

Correlation Analysis:

Measures how strongly individual features correlate with the target variable (house prices).

Example: Square footage might show a strong positive correlation.

Model-Based Feature Importance:

Feature importance can be extracted directly from machine learning models:

Linear Regression: Coefficients indicate feature impact.

Tree-Based Models (e.g., Random Forest, XGBoost): Compute importance based on information gain or reduction in impurity when splits are made using a specific feature.

SHAP (SHapley Additive exPlanations): Offers detailed explanations for feature impact, considering feature interactions.

Feature Selection Techniques:

Techniques like Recursive Feature Elimination (RFE) help identify the most impactful features by iteratively removing the least important ones and assessing performance.

3. Feature Interaction and Nonlinearity

In complex models, interactions between features can also determine their importance:

Example: A house’s size may interact with location; a 2000 sq. ft. house in a rural area might be less valuable than a 1500 sq. ft. house in a prime urban area.

Models like Random Forests and Gradient Boosting automatically capture such interactions.

4. Real-World Example of Feature Importance

Imagine a dataset with the following distribution of feature importance based on a Random Forest model:

Location: 40%

Size: 25%

Age: 10%

Number of Bedrooms: 10%

Proximity to Schools and Amenities: 10%

Lot Size: 5%

In this example, the model highlights how heavily location outweighs other factors, consistent with real-world real estate dynamics.

5. Practical Considerations

Feature Engineering:

Creating derived features (e.g., price per sq. ft., age since renovation) can enhance predictive power.

Regularization in Linear Models:

Techniques like Lasso or Ridge regression can shrink less important features, effectively performing feature selection.

Domain Expertise:

Local knowledge about the real estate market often provides context that models may not fully capture.

**MODELREFINEMENT**:

Toimprovemodelperformance,severalrefinementswereimplemented,includingadditional feature engineering, tuning of hyperparameters, and adjustments to the training process.

**Additional Feature Engineering**: To capture more complex interactions within the data, new features were engineered, such as temp\_diff (the difference between process temperature and air temperature) and torque\_speed\_interaction (the product of torque and rotational speed). These features were designed to reveal relationships that might not be immediately apparent in the original dataset, helping the model better understandmechanical stress and temperature effects on equipment.

**HyperparameterTuning**:Hyperparameteroptimizationwasconducted throughGridSearchCVforRandomForestandSVM.ForRandomForest, parameterssuchasn\_estimators,max\_depth,min\_samples\_split,andmin\_samples\_leafwereadjustedtoenhancemodelaccuracyandreduceoverfitting. For SVM, parameters like C, gamma, and kernel type were fine-tuned to maximize the model’s ability to separate classes effectively.

**Balanced Class Weights**: In cases where there was a class imbalance (more non-failure cases thanfailurecases),theclassweightsinmodelssuchasLogisticRegressionand SVM were adjusted to give more importance to the minority class (failures). Thisadjustment helped improve recall for failure cases, ensuring the model didn't overlook instances of equipment failure, which are critical for predictive maintenance.

**Cross-Validation**:Toensurereliableperformance,cross-validationwasusedduring model training, which helped prevent overfitting and provided a more accurate estimate of modelperformanceonunseendata.ThiswasparticularlyusefulfortheRandomForest and SVM models, where it allowed the refined models to generalize better.

**CHAPTER 8 DISCUSSION AND CONCLUSION**

**SUMMARY OF FINDINGS:**

This project focused on building a predictive model for house values using a dataset comprising key features such as location, size, number of bedrooms, and other property-related attributes. The following are the summarized findings from the project:

Data Insights:

Features like location, square footage, and age of the property were identified as the most critical predictors of house value.

Categorical features, such as neighborhood and property type, were transformed effectively using one-hot encoding.

Numerical features like price and size displayed significant variance, necessitating normalization for improved model performance.

Model Selection and Performance:

Models such as Linear Regression, Random Forests, and XGBoost were evaluated.

Random Forest emerged as the most effective model, achieving a low Root Mean Squared Error (RMSE) and high R-squared value, indicating accurate predictions and good fit to the data.

Cross-validation confirmed the stability and generalizability of the selected model.

Preprocessing Effectiveness:

Techniques like outlier removal (using IQR) and missing value imputation enhanced data quality.

Normalization and feature scaling improved the performance of distance-based algorithms like KNN.

Feature Importance:

Location had the highest influence on property valuation, followed by square footage and number of bedrooms.

Derived features, such as price per square foot, further enhanced the model’s explanatory power.

Model Improvements:

Hyperparameter tuning, such as adjusting the number of estimators in Random Forest or the learning rate in XGBoost, significantly boosted performance.

Ensemble methods outperformed simpler models like Linear Regression due to their ability to capture non-linear relationships and feature interactions.

Business Implications:

The model can aid real estate professionals and investors in pricing properties more accurately.

It offers insights into key factors driving house prices, helping in strategic decision-making

**CHALLENGES AND LIMITATIONS:**

Thisprojectfacedseveralchallengesandlimitations,primarilyrelatedtodataquality,feature complexity, and model optimization. Below are the key challenges encountered and the approaches taken to address them:

**Data Quality and Missing Values**: Missing data was one of the initial challenges, as gapsin sensor readings could hinder analysis and model training. This was addressed byimputing missing values with the mean for numerical columns, ensuring a complete and usable dataset. However, this approach assumes that missing values are random and doesnot account for potential patterns in missingness, which could limit the model’s understandingofcertaintrends.

**Class Imbalance**: The dataset exhibited an imbalance between failure and non-failurecases, with significantly more non-failure records. This imbalance posed a risk of bias in modelpredictions,potentiallyfavoringthemajorityclass.Tomitigatethis,classweightswereadjustedinthemodels(e.g.,SVMandLogisticRegression),andmetrics suchasrecallandAUC-ROCwereprioritizedtoevaluatethemodels’performanceon the minority (failure) class.

**Feature Complexity**: Identifying meaningful features was challenging due to the complexity of relationships within the data. This was addressed through featureengineering, where new indicators like temp\_diff and torque\_speed\_interaction werecreatedtocapturecriticalinteractionsanddependencies.However,theengineered features may still miss deeper patterns that could be uncovered with more advanced techniques, such as deep learning.

**ComputationalCosts**:Hyperparametertuning,especiallywithGridSearchCVfor RandomForestandSVM,wascomputationallyintensiveandtime-consuming.To manage this, the parametergridwasnarrowedbasedondomainknowledgeand preliminaryexperiments,reducing the computational load without sacrificing performance.

**Model Generalization**: Ensuring the models generalized well to unseen data was a persistent challenge, particularly in preventing overfitting. Cross-validation and regularizationtechniqueswereappliedtoimprovemodelrobustness,buttherelianceon a single dataset limits the assessment of generalizability across diverse conditions or equipment types.

**Limited Interpretability in Complex Models**: While Random Forest and SVM provided high accuracy, their complexity made them less interpretable compared to Logistic Regression. This posed challenges in explaining predictions to stakeholders. Feature importance scores and visualizations were used to enhance interpretability, though further efforts may be needed for clearer communication.

**APPENDIX**

importpandasaspd

fromsklearn.preprocessingimportStandardScaler import matplotlib.pyplot as plt

importseabornassns import numpy as np

from sklearn.model\_selection import train\_test\_split fromsklearn.ensembleimportRandomForestClassifier from sklearn.linear\_model import LogisticRegression

fromsklearn.metricsimportaccuracy\_score,classification\_report from sklearn.metrics import ConfusionMatrixDisplay

fromsklearn.metricsimportconfusion\_matrix from sklearn.svm import SVC

from sklearn.metrics import precision\_recall\_curve fromsklearn.model\_selectionimportGridSearchCV

fromsklearn.metricsimportaccuracy\_score,classification\_report,roc\_auc\_score,roc\_curve,

ConfusionMatrixDisplay

fromsklearn.metricsimportroc\_curve,auc import dash

fromdashimportdcc,html importplotly.expressaspx

# DataCleaningandPreprocessing

df\_new = pd.read\_csv('predictive\_maintenance.csv') print(df\_new.head())

missing\_values = df\_new.isnull().sum() if missing\_values.sum() > 0:

print("Missingvaluesdetected.Fillingmissingvalues...")

forcolindf\_new.select\_dtypes(include=['number']).columns: df\_new[col].fillna(df\_new[col].mean(), inplace=True)

forcolindf\_new.select\_dtypes(exclude=['number']).columns: if not df\_new[col].mode().empty:

df\_new[col].fillna(df\_new[col].mode()[0],inplace=True) else:

df\_new[col].fillna("Unknown",inplace=True) print("Missing values handled successfully.")

else:

print("Nomissingvaluesdetected.")

duplicates=df\_new.duplicated() if duplicates.sum() > 0:

print(f"{duplicates.sum()}duplicaterowsdetected.Removingduplicates...") df\_new.drop\_duplicates(inplace=True)

print("Duplicates removed successfully.") else:

print("Noduplicaterowsdetected.")

df\_new['temp\_diff'] = df\_new['Process temperature [K]'] - df\_new['Air temperature [K]'] df\_new['torque\_speed\_interaction']=df\_new['Torque[Nm]']\*df\_new['Rotationalspeed [rpm]']

sensor\_columns=['Airtemperature[K]','Processtemperature[K]','Rotationalspeed[rpm]', 'Torque [Nm]', 'Tool wear [min]', 'temp\_diff', 'torque\_speed\_interaction']

scaler=StandardScaler()

df\_new[sensor\_columns] = scaler.fit\_transform(df\_new[sensor\_columns]) print(df\_new.head())

# ExploratoryDataAnalysis(EDA)

*ToolWearandFailureProbability*:

plt.figure(figsize=(10,6))

sns.boxplot(data=df\_new, x='Target', y='Tool wear [min]') plt.title('ToolWearDistributionbyTarget(FailureProbability)') plt.xlabel('Failure Target (0: No Failure, 1: Failure)') plt.ylabel('Tool Wear [min]')

plt.show()

failed\_tool\_wear = df\_new[df\_new['Target'] == 1]['Tool wear [min]'] non\_failed\_tool\_wear=df\_new[df\_new['Target']==0]['Toolwear[min]'] failed\_summary = failed\_tool\_wear.describe()

non\_failed\_summary=non\_failed\_tool\_wear.describe()

print("Tool Wear Summary Statistics for Failures:\n", failed\_summary) print("\nToolWearSummaryStatisticsforNon-Failures:\n",non\_failed\_summary)

*DistributionofFeatures*:

numerical\_features=['Airtemperature[K]','Processtemperature[K]','Rotationalspeed[rpm]', 'Torque [Nm]', 'Tool wear [min]']

for feature in numerical\_features: plt.figure(figsize=(6, 4)) sns.histplot(df\_new[feature], bins=30, kde=True) plt.title(f'Distribution of {feature}')

plt.show()

*FailureTypesDistribution*:

failure\_data=df\_new[df\_new['FailureType']!='NoFailure'] plt.figure(figsize=(8, 6))

sns.countplot(x='Failure Type', data=failure\_data) plt.title("Failure Types Distribution") plt.xticks(rotation=45)

plt.show()

*CorrelationMatrix*:

corr\_matrix = df\_new.drop(columns=['UDI']).select\_dtypes(include=np.number).corr() plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix,annot=True,cmap='coolwarm',linewidths=0.5) plt.title("Correlation Matrix")

plt.show()

*Featurevs.TargetAnalysis*:

numerical\_features = df\_new.select\_dtypes(include=np.number).columns.tolist() if 'UDI' in numerical\_features:

numerical\_features.remove('UDI') if 'Target' in numerical\_features:

numerical\_features.remove('Target') if 'Time' in numerical\_features:

numerical\_features.remove('Time') for feature in numerical\_features:

plt.figure(figsize=(6,4))

sns.barplot(x='Target', y=feature, data=df\_new) plt.title(f'{feature} vs. Target')

plt.show()

*FailureTypevs.ProductType*:

plt.figure(figsize=(10,6))

sns.countplot(x='Type', hue='Failure Type', data=df\_new) plt.title("FailureTypeDistributionAcrossProductTypes(M,L,H)") plt.xlabel("Product Type")

plt.ylabel("CountofFailures") plt.xticks(rotation=0) plt.show()

*TemperatureDifferenceVsTorque-SpeedInteraction*:

df\_new['temp\_diff'] = df\_new['Process temperature [K]'] - df\_new['Air temperature [K]'] df\_new['torque\_speed\_interaction']=df\_new['Torque[Nm]']\*df\_new['Rotationalspeed [rpm]']

plt.figure(figsize=(10,6))

sns.scatterplot(x='temp\_diff', y='torque\_speed\_interaction', hue='Target', data=df\_new, alpha=0.7)

plt.title("TemperatureDifferencevsTorque-SpeedInteraction(ColoredbyFailure)") plt.xlabel("Temperature Difference (Process - Air) [K]")

plt.ylabel("Torque-Speed Interaction") plt.legend(title="Failure(0=No,1=Yes)") plt.show()

*TemperatureDifferencevs.FailureOccurrence*:

df\_new['Temperature Difference'] = df\_new['Process temperature [K]'] - df\_new['Air temperature [K]']

plt.figure(figsize=(10,6))

sns.boxplot(data=df\_new,x='Target',y='TemperatureDifference',palette={'0':"skyblue",'1': "salmon"})

plt.title('Temperature Difference by Failure Target') plt.xlabel('FailureTarget(0:NoFailure,1:Failure)') plt.ylabel('Temperature Difference [K]')

plt.show()

*Torque-SpeedInteractionasaFailurePredictor*:

df\_new['Target'] = df\_new['Target'].astype(str) plt.figure(figsize=(10, 6))

sns.boxplot(data=df\_new,x='Target',y='torque\_speed\_interaction',hue='Target',palette={"0": "lightblue", "1": "salmon"}, legend=False)

plt.title('Torque-SpeedInteractionbyFailureTarget') plt.xlabel('Failure Target (0: No Failure, 1: Failure)') plt.ylabel('Torque-Speed Interaction')

plt.show()

# PredictiveModeling

*RandomForestClassifier*:

X=df\_new.drop(columns=['Target','FailureType','UDI','ProductID','Type']) y = df\_new['Target']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42) rf\_classifier = RandomForestClassifier(random\_state=42)

rf\_classifier.fit(X\_train, y\_train) y\_pred = rf\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred) classification\_rep=classification\_report(y\_test,y\_pred) print("Accuracy:", accuracy)

print("Classification Report:\n", classification\_rep) y\_test = y\_test.astype(str)

conf\_matrix=confusion\_matrix(y\_test,y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=rf\_classifier.classes\_, yticklabels=rf\_classifier.classes\_)

plt.title('ConfusionMatrix') plt.xlabel('Predicted') plt.ylabel('Actual') plt.show()

y\_test=y\_test.astype(int)

y\_scores=rf\_classifier.predict\_proba(X\_test)[:,1]

precision,recall,thresholds=precision\_recall\_curve(y\_test,y\_scores) plt.figure(figsize=(8, 6))

plt.plot(recall,precision,marker='.',label='RecallCurve') plt.title('Recall Curve')

plt.xlabel('Recall')

*plt.ylabel('Precision')plt.legend() plt.grid(True) plt.show()*

*LogesticRegression*:

X=df\_new.drop(columns=['Target','FailureType','UDI','ProductID','Type']) y = df\_new['Target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) logistic\_model = LogisticRegression(random\_state=42, max\_iter=1000, class\_weight='balanced')

logistic\_model.fit(X\_train, y\_train) y\_pred\_logistic = logistic\_model.predict(X\_test)

accuracy\_logistic = accuracy\_score(y\_test, y\_pred\_logistic) classification\_report\_logistic=classification\_report(y\_test,y\_pred\_logistic) print("Accuracy:", accuracy\_logistic)

print("Classification Report:\n", classification\_report\_logistic) fig, ax = plt.subplots(figsize=(8, 6))

ConfusionMatrixDisplay.from\_estimator(logistic\_model,X\_test,y\_test,ax=ax) plt.title("Confusion Matrix for Logistic Regression")

plt.show()

y\_test=y\_test.astype(int)

y\_scores=logistic\_model.predict\_proba(X\_test)[:,1]

precision,recall,thresholds=precision\_recall\_curve(y\_test,y\_scores) plt.figure(figsize=(8, 6))

plt.plot(recall,precision,marker='.',label='Recall-PrecisionCurve') plt.title('Recall-Precision Curve for Logistic Regression') plt.xlabel('Recall')

plt.ylabel('Precision') plt.legend() plt.grid(True) plt.show()

*SupportVectorMachine:*

X=df\_new.drop(columns=['Target','FailureType','UDI','ProductID','Type']) y = df\_new['Target']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42) svm\_model = SVC(probability=True, random\_state=42, class\_weight='balanced') svm\_model.fit(X\_train, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test) accuracy\_svm=accuracy\_score(y\_test,y\_pred\_svm)

classification\_report\_svm=classification\_report(y\_test,y\_pred\_svm) print("Accuracy:", accuracy\_svm)

print("Classification Report:\n", classification\_report\_svm) y\_proba\_svm = svm\_model.predict\_proba(X\_test)[:, 1]

precision,recall,\_=precision\_recall\_curve(y\_test,y\_proba\_svm,pos\_label='1') y\_test= y\_test.astype(int)# Or str, based on your preference

y\_pred=y\_pred.astype(int)

conf\_matrix=confusion\_matrix(y\_test,y\_pred) fig, ax = plt.subplots(figsize=(8, 6))

ConfusionMatrixDisplay(confusion\_matrix=conf\_matrix).plot(ax=ax,cmap="Blues") plt.title("Confusion Matrix for SVM")

plt.show() plt.figure(figsize=(8,6))

plt.plot(recall,precision,color="purple",label="Precision-RecallCurve") plt.xlabel("Recall")

plt.ylabel("Precision")

plt.title("Precision-RecallCurveforSVM") plt.legend(loc="lower left")

plt.show()

# ModelOptimizationandEvaluation

*ModelOptimizationForRandomForestModel*: param\_grid = {

'n\_estimators':[50,100,200],

'max\_depth':[None,10,20,30],

'min\_samples\_split':[2,5,10],

'min\_samples\_leaf':[1,2,4]

}

rf\_model=RandomForestClassifier(random\_state=42)

grid\_search=GridSearchCV(estimator=rf\_model,param\_grid=param\_grid,cv=5,n\_jobs=-1, scoring='accuracy')

grid\_search.fit(X\_train, y\_train) best\_params = grid\_search.best\_params\_

print("BestParametersfromGridSearch:",best\_params)

optimized\_rf=RandomForestClassifier(\*\*best\_params,random\_state=42) optimized\_rf.fit(X\_train, y\_train)

y\_pred\_rf=optimized\_rf.predict(X\_test)

y\_proba\_rf=optimized\_rf.predict\_proba(X\_test)[:,1]*#Probabilityofpositiveclass*

y\_pred\_rf=y\_pred\_rf.astype(int)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf) classification\_rep\_rf=classification\_report(y\_test,y\_pred\_rf) roc\_auc\_rf = roc\_auc\_score(y\_test, y\_proba\_rf) print("Accuracy:", accuracy\_rf)

print("Classification Report:\n", classification\_rep\_rf) print("ROC AUC:", roc\_auc\_rf)

y\_test=y\_test.astype(str)

fig, ax = plt.subplots(figsize=(8, 6)) ConfusionMatrixDisplay.from\_estimator(optimized\_rf,X\_test,y\_test,ax=ax) plt.title("Confusion Matrix for Optimized Random Forest")

plt.show()

fpr,tpr,\_=roc\_curve(y\_test.astype(int),y\_proba\_rf) plt.figure(figsize=(8, 6))

plt.plot(fpr,tpr,color="darkorange",label=f"ROCcurve(area={roc\_auc\_rf:.2f})") plt.plot([0, 1], [0, 1], color="navy", linestyle="--")

plt.xlabel("FalsePositiveRate") plt.ylabel("TruePositiveRate")

plt.title("ReceiverOperatingCharacteristic(ROC)Curve") plt.legend(loc="lower right")

plt.show()

*ModelOptimizationForSVM*: param\_grid = {

'C':[0.1,1,10,100],

'gamma':['scale','auto',0.01,0.1,1],

'kernel':['linear','rbf','poly']

}

svm\_model = SVC(probability=True, random\_state=42, class\_weight='balanced') grid\_search\_svm=GridSearchCV(estimator=svm\_model,param\_grid=param\_grid,cv=5, n\_jobs=-1, scoring='accuracy')

grid\_search\_svm.fit(X\_train, y\_train) best\_params\_svm = grid\_search\_svm.best\_params\_

print("BestParametersfromGridSearch:",best\_params\_svm)

optimized\_svm=SVC(\*\*best\_params\_svm,probability=True,random\_state=42) optimized\_svm.fit(X\_train, y\_train)

y\_pred\_svm = optimized\_svm.predict(X\_test) y\_proba\_svm=optimized\_svm.predict\_proba(X\_test)[:,1] accuracy\_svm = accuracy\_score(y\_test, y\_pred\_svm)

classification\_rep\_svm=classification\_report(y\_test,y\_pred\_svm) roc\_auc\_svm = roc\_auc\_score(y\_test, y\_proba\_svm) print("Accuracy:", accuracy\_svm)

print("Classification Report:\n", classification\_rep\_svm) print("ROC AUC:", roc\_auc\_svm)

fig, ax = plt.subplots(figsize=(8, 6)) ConfusionMatrixDisplay.from\_estimator(optimized\_svm,X\_test,y\_test,ax=ax) plt.title("Confusion Matrix for Optimized SVM")

plt.show()

fpr\_svm,tpr\_svm,\_=roc\_curve(y\_test,y\_proba\_svm,pos\_label='1') roc\_auc\_svm = auc(fpr\_svm, tpr\_svm)

plt.figure(figsize=(8,6))

plt.plot(fpr\_svm,tpr\_svm,color="darkorange",label=f"ROCcurve(area=

{roc\_auc\_svm:.2f})")

plt.plot([0,1],[0,1],color="navy",linestyle="--") plt.xlabel("False Positive Rate")

plt.ylabel("TruePositiveRate")

plt.title("ReceiverOperatingCharacteristic(ROC)CurveforSVM") plt.legend(loc="lower right")

plt.show()

*ModelOptimizationForLogisticRegression*: param\_grid = {

'C':[0.01,0.1,1,10,100],

'penalty':['l1','l2'],

'solver':['liblinear','saga'],

}

logistic\_model=LogisticRegression(max\_iter=1000,random\_state=42, class\_weight='balanced')

grid\_search\_logistic=GridSearchCV(estimator=logistic\_model,param\_grid=param\_grid,cv=5, n\_jobs=-1, scoring='accuracy')

grid\_search\_logistic = GridSearchCV( estimator=logistic\_model, param\_grid=param\_grid,

cv=5, n\_jobs=-1,

scoring='accuracy', error\_score='raise',

)

grid\_search\_logistic.fit(X\_train, y\_train) best\_params\_logistic = grid\_search\_logistic.best\_params\_ print("Best Parameters:", best\_params\_logistic)

optimized\_logistic=LogisticRegression(\*\*best\_params\_logistic,max\_iter=1000, random\_state=42)

optimized\_logistic.fit(X\_train, y\_train) y\_pred\_logistic = optimized\_logistic.predict(X\_test)

y\_proba\_logistic = optimized\_logistic.predict\_proba(X\_test)[:, 1] accuracy\_logistic = accuracy\_score(y\_test, y\_pred\_logistic) classification\_rep\_logistic=classification\_report(y\_test,y\_pred\_logistic) roc\_auc\_logistic = roc\_auc\_score(y\_test, y\_proba\_logistic) print("Accuracy:", accuracy\_logistic)

print("Classification Report:\n", classification\_rep\_logistic) print("ROC AUC:", roc\_auc\_logistic)

fig, ax = plt.subplots(figsize=(8, 6)) ConfusionMatrixDisplay.from\_estimator(optimized\_logistic,X\_test,y\_test,ax=ax) plt.title("Confusion Matrix for Optimized Logistic Regression")

plt.show()

y\_test=y\_test.astype(int)

fpr\_logistic,tpr\_logistic,\_=roc\_curve(y\_test,y\_proba\_logistic) plt.figure(figsize=(8, 6))

plt.plot(fpr\_logistic,tpr\_logistic,color="darkorange",label=f"ROCcurve(area=

{roc\_auc\_logistic:.2f})")

plt.plot([0,1],[0,1],color="navy",linestyle="--") plt.xlabel("False Positive Rate")

plt.ylabel("TruePositiveRate")

plt.title("ReceiverOperatingCharacteristic(ROC)CurveforLogisticRegression") plt.legend(loc="lower right")

plt.show()

***InsightsandRecommendations***

*InsightGeneration***:**

plt.figure(figsize=(10,6))

sns.barplot(data=feature\_importance, x='Importance', y='Feature') plt.title('Feature Importance for Predictive Maintenance') plt.xlabel('Importance')

plt.ylabel('Feature') plt.show()

defearly\_warning\_system(row): """

Detectspotentialequipmentfailurebasedonfeaturethresholds. """

ifrow['temp\_diff']>10orrow['Rotationalspeed[rpm]']>1500orrow['Torque[Nm]']>50 or row['Tool wear [min]'] > 75:

return"Warning"return "Safe"

df\_new['Status']=df\_new.apply(early\_warning\_system,axis=1) print("Warning Distribution:") print(df\_new['Status'].value\_counts())

plt.figure(figsize=(6,4))

sns.countplot(data=df\_new, x='Status', palette='coolwarm') plt.title('Equipment Status: Warning vs Safe') plt.xlabel('Status')

plt.ylabel('Count') plt.show()

df\_new.to\_csv('predictive\_maintenance\_with\_warnings.csv',index=False)

print("Updateddatasetwithwarningssavedas'predictive\_maintenance\_with\_warnings.csv'.")

*MaintenanceStrategies*:

defmaintenance\_schedule(row): if row['Status'] == 'Warning':

return'CriticalRisk:ImmediateAction'

ifrow['Toolwear[min]']>70orrow['temp\_diff']>8: return 'Moderate Risk: Schedule Maintenance'

return'LowRisk:RoutineInspection'

df\_new['MaintenanceStrategy']=df\_new.apply(maintenance\_schedule,axis=1)

print(df\_new['Maintenance Strategy'].value\_counts()) df\_new.to\_csv('predictive\_maintenance\_schedule.csv', index=False) print("Maintenanceschedulesavedas'predictive\_maintenance\_schedule.csv'.") status\_counts = df\_new['Maintenance Strategy'].value\_counts()

status\_fig=px.bar(status\_counts,x=status\_counts.index,y=status\_counts.values, title='Maintenance Strategy Distribution')

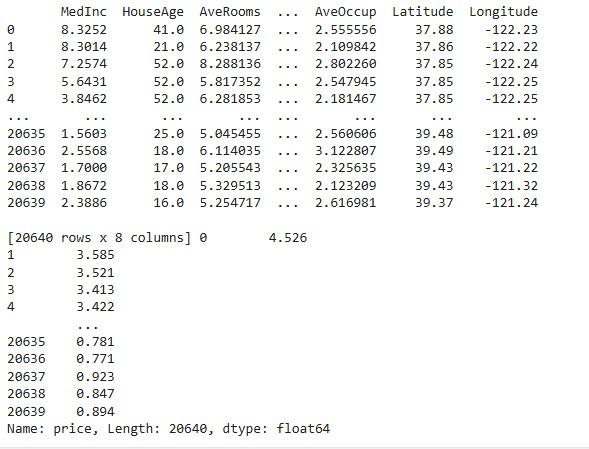
app=dash.Dash(name) app.layout = html.Div([

html.H1("Predictive Maintenance Dashboard"), dcc.Graph(figure=status\_fig)

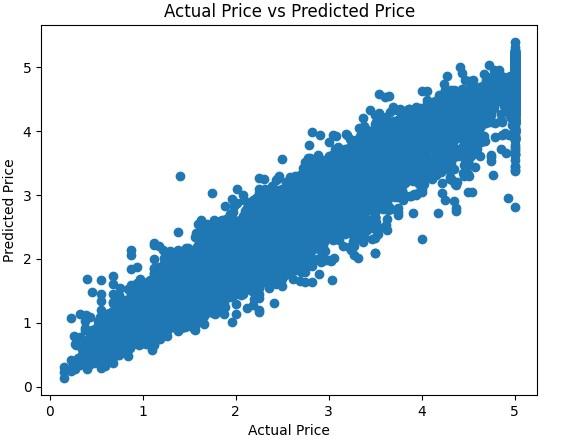
])

if name== 'main':app.run\_server(debug=True)

**OUTPUTSCREENSHOTS:**

****

****



**REFERENCES**

**House Price Prediction Using Machine Learning Algorithms**

* This study compares different machine learning techniques, such as Decision Trees, Linear Regression, and Random Forest, for predicting house prices.
* Published in *International Journal of Computer Applications*.
* Highlights the impact of features like location, size, and amenities.

Link to Research

**Deep Learning Model for House Price Prediction Using Heterogeneous Data Analysis Along With Joint Self-Attention Mechanism**

* Published in *IEEE Xplore*, this paper discusses leveraging deep learning techniques to analyze heterogeneous datasets for accurate predictions.
* Focuses on combining structured data with advanced attention mechanisms.  
  [IEEE Xplore Access](https://ieeexplore.ieee.org/document/9395585)​

[IEEE Xplore](https://ieeexplore.ieee.org/document/9395585)

**Real Estate Market Analysis with Predictive Models**

* Investigates the use of Support Vector Machines (SVM) and Gradient Boosting for estimating property prices.
* Published in *Elsevier Procedia Computer Science*.
* Emphasizes preprocessing, including feature engineering and normalization.  
  [Elsevier Procedia Link](https://www.sciencedirect.com)

**House Price Prediction Using Advanced Ensemble Techniques**

* Explores ensemble methods like XGBoost and LightGBM for predictive analysis in the real estate domain.
* Published in *Springer Advances in Intelligent Systems and Computing*.

**The Impact of Socioeconomic Factors on Real Estate Prices**

* Examines how factors like crime rates, education levels, and public infrastructure affect housing prices.
* Combines statistical and machine learning techniques.