

**COLLEGE CODE : 6208**

**NAME : V. S. Rubika**

**IBM REG NO : au620821106076**

**PROJECT NAME : PREDICTING HOUSE PRICES USING MACHINE LEARNING**

**Phase 5: Project Documentation & Submission**

**Topic: In this section we will document the complete**

**Project and prepare it for submission.**

Introduction to house price predictor :

* The predictor gathers a comprehensive dataset of past property sales, including information like location, square footage, number of bedrooms, bathrooms, year built, and other relevant factors. Additional data may include economic indicators, neighborhood characteristics, and market trends.
* Raw data is cleaned, transformed, and organized to remove inconsistencies and prepare it for analysis. This may involve handling missing values, encoding categorical variables, and scaling numerical features.
* The predictor identifies which features (attributes) have the most impact on property prices. Some features may be more relevant than others, and machine learning models aim to focus on the most influential factors.
* Various machine learning algorithms, such as linear regression, decision trees, random forests, or neural networks, can be used to build predictive models. These models learn from the historical data and create a mathematical relationship between the property features and prices.
* The selected model is trained on a portion of the dataset, typically called the training set. The model learns from this data and adjusts its parameters to make accurate predictions.
* The model’s performance is evaluated using a separate dataset, often referred to as the validation set. This ensures the model can generalize its predictions to unseen data and provides an estimate of its accuracy.
* Once the model is trained and validated, it can be used to predict the prices of new or unlisted properties. Users input property details, and the model generates an estimated price based on the patterns it has learned.

Here’s a list of tools and software commonly used in the process:

1. Programming Language:
   * Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy,pandas, scikit-learn, and more.
2. Integrated Development Environment (IDE):
   * Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.
3. Machine Learning Libraries:
   * You’ll need various machine learning libraries, including:
   * Scikit-learn for building and evaluating machine learning models.
   * TensorFlow or PyTorch for deep learning, if needed.
   * XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

* Tools like Matplotlib, Seaborn, or Plotly are essential for dataexploration and visualization.

5. Data Preprocessing Tools:

* Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6.Data Collection and Storage:

* Depending on your data source, you might need web scraping Tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite,PostgreSQL) for data storage.

7. Version Control:

* Version control systems like Git are valuable for tracking Changes in your code and collaborating with others.

8. Notebooks and Documentation:

* Tools for documenting your work, such as Jupyter Notebooks Or Markdown for creating README files and documentation.

9. Hyperparameter Tuning:

* Tools like GridSearchCV or RandomizedSearchCV from Scikit-learn can help with hyperparameter tuning.

10. Web Development Tools (for Deployment):

* If you plan to create a web application for model deployment,Knowledge of web development tools like Flask or Django for backend Development, and HTML, CSS, and JavaScript for the front-end can be Useful.

11. Cloud Services (for Scalability):

* For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources.

12. External Data Sources (if applicable):

* Depending on your project’s scope, you might require tools to Access external data sources, such as APIs or data scraping tools.

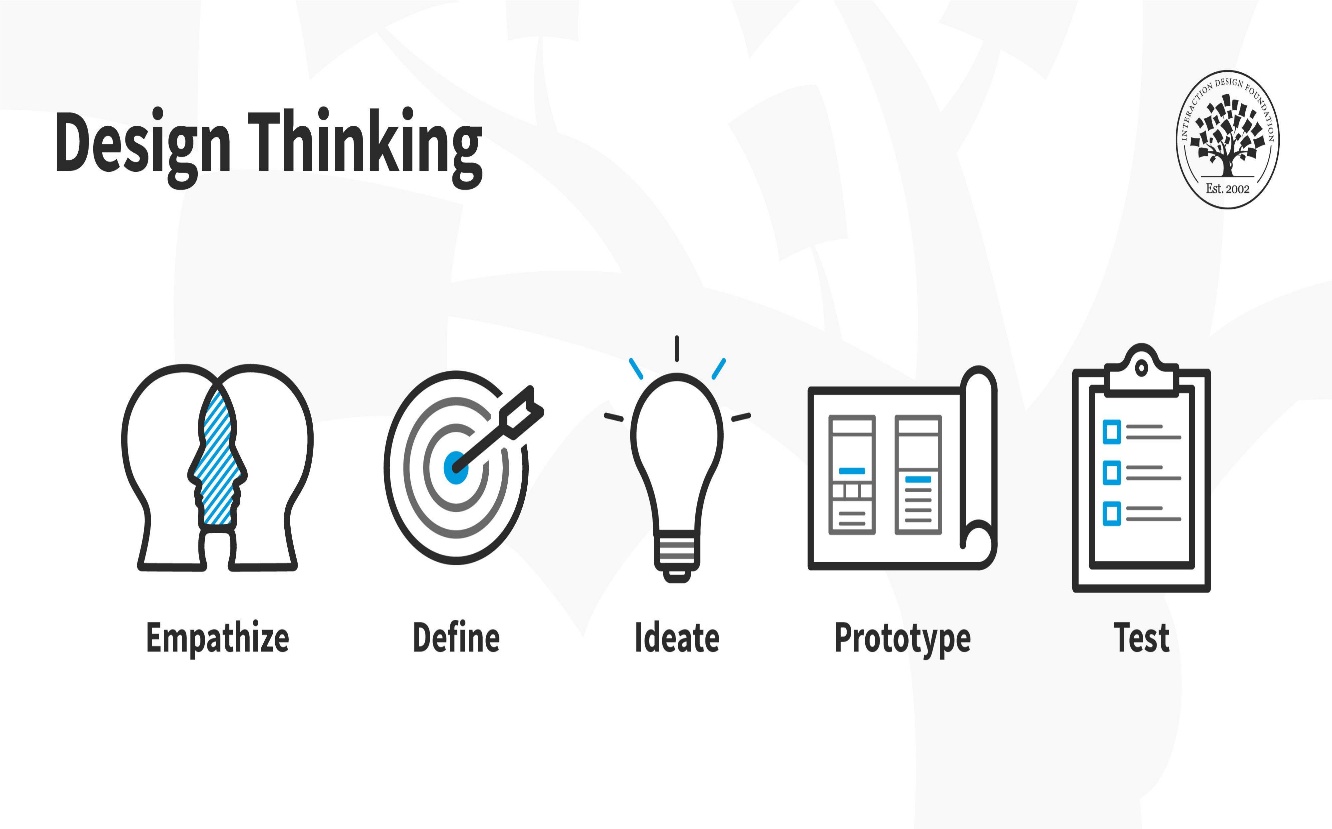
13. Data Annotation and Labeling Tools (if applicable):

* For specialized projects, tools for data annotation and Labeling may be necessary, such as Labelbox or Supervisely.

14. Geospatial Tools (for location-based features):

* If your dataset includes geospatial data, geospatial libraries Like GeoPandas can be helpful.

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:



**1.Empathize:**

* Understand the needs and expectations of potential users, such as homebuyers, sellers, and real estate professionals.
* Conduct interviews, surveys, or focus groups to gather insights on what factors are most important in their decision-making.

**2. Define:**

* Clearly define the problem you are trying to solve, such as accurately predicting house prices.
* Establish specific goals and metrics for your machine learning model, like mean absolute error or root mean squared error.

**3.Ideate:**

* Brainstorm potential features or attributes that might influence house prices (e.g., square footage, neighborhood, school quality).
* Explore different machine learning algorithms suitable for regression tasks, like linear regression, decision trees, or neural networks.

1. **Prototype:**

* Create a preliminary version of your machine learning model using a small dataset.
* Develop a simple interface or visualization to showcase the predicted prices.

1. **Test**:

* Evaluate the prototype’s performance using cross-validation or holdout datasets.
* Gather feedback from potential users and adjust your model and interface accordingly.

1. **Refine**:

* Fine-tune your model by optimizing hyperparameters and feature selection.
* Continuously refine the user interface based on feedback and usability testing.

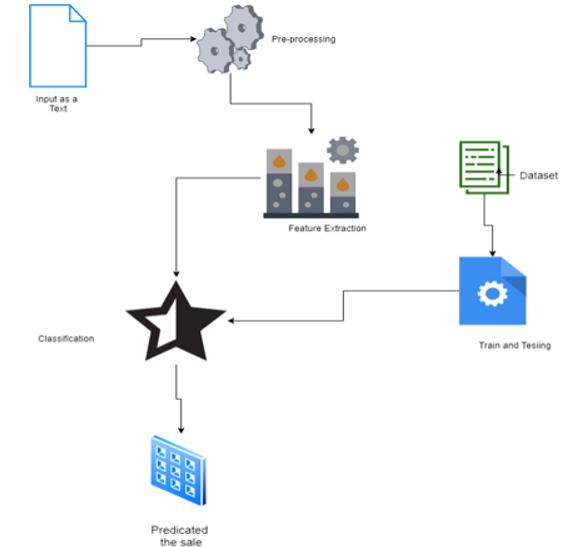
1. **Implement**:

* Deploy the finalized machine learning model into a production environment, ensuring scalability and reliability.
* Make the prediction tool accessible to users through a web app, mobile app, or API.

1. **Iterate**:

* Regularly update the model with new data to ensure its accuracy and relevance.
* Keep refining the user experience based on user feedback and evolving user needs.

Design thinking in this context helps create a user-centered and effective machine learning solution for predicting house prices while continuously improving it based on real-world feedback and changing market conditions.

2.DESIGN AND INNOVATION :

1.Deep Learning :

* Utilizing deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for house price prediction. Deep learning models can automatically learn complex patterns and relationships in the data.

2.Feature Engineering :

* Using automated feature engineering techniques and tools like feature selection algorithms and domain-specific feature creation to enhance model performance.

3.Ensemble Methods :

* Combining multiple machine learning models, such as random forests, gradient boosting, and stacking, to create more robust and accurate predictions.

4.Geo-Spatial Analysis :

* Incorporating geographic information systems (GIS) and spatial data analysis to account for location-specific factors that influence property prices, such as proximity to amenities, schools, or public transportation.

5.Natural Language Processing (NLP) :

* Analyzing textual data from real estate listings, user reviews, or social media to extract insights that impact property values.

6.Time Series Analysis :

* Recognizing and modeling time-dependent trends and seasonality in the housing market, which can be crucial for predicting future prices accurately.

7.AI Explainability :

* Developing methods to make machine learning models more interpretable and explainable, ensuring transparency and trust in predictions.

8.Data Augmentation :

* Using techniques like data synthesis and generative adversarial networks (GANs) to create synthetic data that can enhance model training when real data is limited.

9.Hybrid Models :

* Combining traditional statistical methods with machine learning algorithms to leverage the strengths of both approaches.

10.Mobile Apps and Chatbots :

* Developing user-friendly mobile applications and chatbots that allow users to get real-time house price estimates and market insights.

**PROGRAM:**

**Import numpy as np**

**Import pandas as pd**

**From sklearn.model\_selection import train\_test\_split**

**From xgboost import XGBRegressor**

**From sklearn.metrics import mean\_squared\_error**

**Import matplotlib.pyplot as plt**

**# Load your dataset**

**Data = pd.read\_csv(‘your\_dataset.csv’)**

**# Preprocess the data (e.g., handle missing values, encode categorical variables)**

**# Split data into training and testing sets**

**X = data.drop(‘target\_column’, axis=1)**

**Y = data[‘target\_column’]**

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

**# XGBoost regression model**

**Xgb\_model = XGBRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=3, random\_state=42)**

**Xgb\_model.fit(X\_train, y\_train)**

**Xgb\_predictions = xgb\_model.predict(X\_test)**

**# Evaluate the model (for regression tasks, you can use metrics like Mean Squared Error)**

**Mse = mean\_squared\_error(y\_test, xgb\_predictions)**

**Print(“Mean Squared Error:”, mse)**

**# Visualize predictions (optional)**

**Plt.scatter(y\_test, xgb\_predictions)**

**Plt.xlabel(“Actual Values”)**

**Plt.ylabel(“Predicted Values”)**

**Plt.title(“Actual vs. Predicted Values”)**

**Plt.show()**

**3.LOADING AND PREPROCESSING THE DATASET:**

1.Loading the dataset :

**Loading the dataset using machine learning is the process of bringing The data into the machine learning environment so that it can be usedTo train and evaluate a model.**

**The specific steps involved in loading the dataset will vary dependingOn the machine learning library or framework that is being used.Howev Identify er, there are some general steps that are common to mostMachine learning frameworks:**

**a. the dataset:**

* **The first step is to identify the dataset that you want to load. This Dataset may be stored in a local file, in a database, or in a cloud storage Service.**

**b.Load the dataset:**

* **Once you have identified the dataset, you need to load it into the Machine learning environment. This may involve using a built-in Function in the machine learning library, or it may involve writing your Own code.**

**c.Preprocess the dataset:**

* **Once the dataset is loaded into the machine learning environment,You may need to preprocess it before you can start training and Evaluating your model. This may involve cleaning the data, transforming.**

**Program :**

**Import pandas as pd**

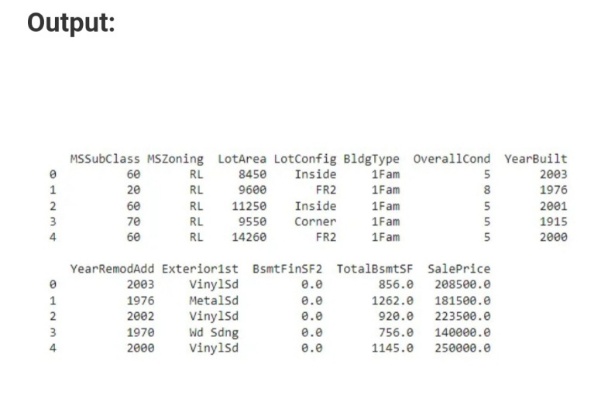
**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**Dataset = pd.read\_excel(“HousePricePrediction.xlsx”)**

**# Printing first 5 records of the dataset**

**Print(dataset.head(5))**

****

**2.Preprocessing the dataset:**

* **Data preprocessing is the process of cleaning, transforming, and Integrating data in order to make it ready for analysis.**
* **This may involve removing errors and inconsistencies, handling Missing values, transforming the data into a consistent format, and Scaling the data to a suitable range**

**Data Preprocessing:**

* **Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, greater is the reliance on the produced results. Incomplete, noisy, and inconsistent data are the properties of large real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.**
* **Incomplete data can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions.**

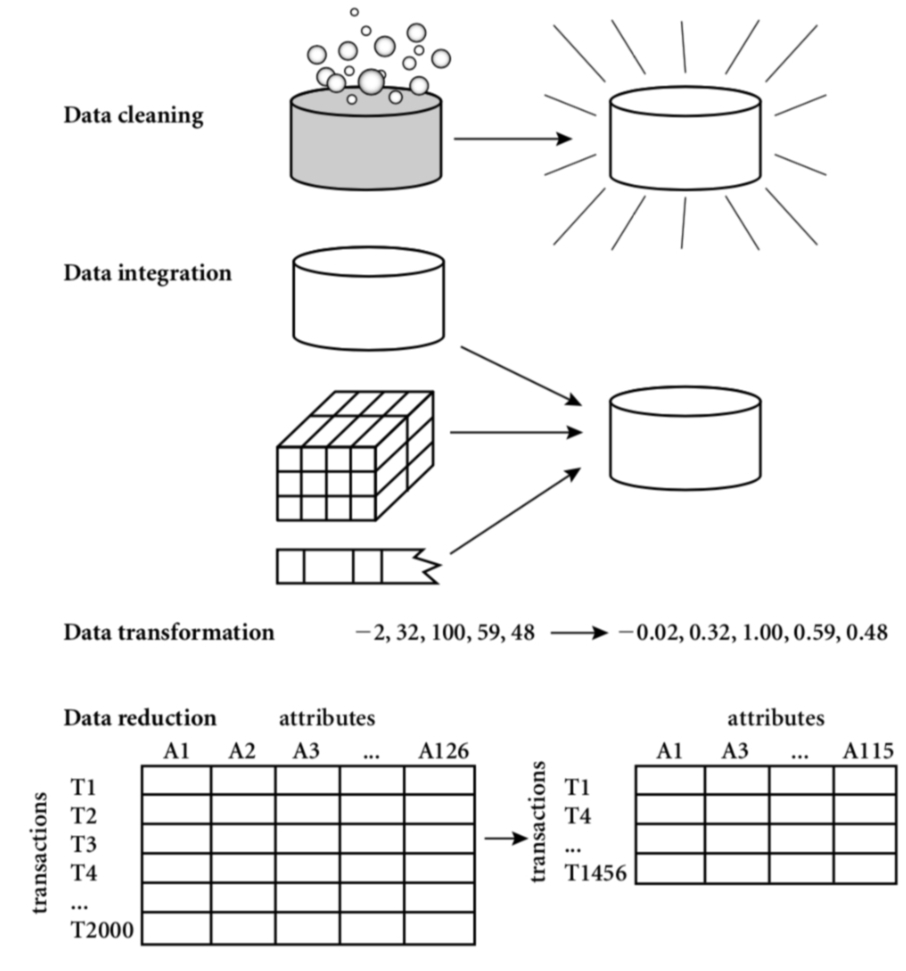
**There are a number of data preprocessing techniques available such as,**

**1).Data Cleaning**

**2).Data Integration**

**3).Data Transformation**

**4).Data Reduction**

****

* **Data cleaning can be applied to filling in missing values, remove noise, resolving inconsistencies, identifying and removing outliers in the data.**
* **Data integration merges data from multiple sources into a coherent data store, such as a data warehouse.**
* **Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.**
* **Data reduction can reduce the data size by eliminating redundant features, or clustering, for instance.**

**Reference: Data Mining:Concepts and Techniques Second Edition, Jiawei Han, Micheline Kamber.**

**PS: This is my first kaggle notebook contribution. Hope you like it!!**

**Import the required libraries**

**Import warnings**

**Warnings.filterwarnings(‘ignore’)**

**Import numpy as np**

**Import pandas as pd**

**Import matplotlib.pyplot as plt**

**From operator import itemgetter**

**From sklearn.experimental import enable\_iterative\_imputer**

**From sklearn.impute import IterativeImputer**

**From sklearn.preprocessing import OrdinalEncoder**

**From category\_encoders.target\_encoder import TargetEncoder**

**From sklearn.preprocessing import StandardScaler**

**From sklearn.ensemble import (GradientBoostingRegressor, GradientBoostingClassifier)**

**Import xgboost**

**Load the dataset for training and testing**

**Train = pd.read\_csv(‘../input/house-prices-advanced-regression-techniques/train.csv’)**

**Test = pd.read\_csv(‘../input/house-prices-advanced-regression-techniques/test.csv’)**

**4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING,EVALUATION etc.,**

FEATURE SELECTION:

Data Collection and Preprocessing:

* Gather your dataset, which should include various features like the number of bedrooms, square footage, location, etc. Preprocess the data by handling missing values, outliers, and encoding categorical variables.

Correlation Analysis:

* Calculate the correlation between each feature and the target variable (house price). Features with higher absolute correlations are generally more relevant. You can use methods like Pearson correlation for continuous features and point-biserial correlation for binary.

Feature Importance from Models:

* Train a baseline model (e.g., a decision tree, random forest, or XGBoost) and analyze the feature importance scores. Features with higher importance scores contribute more to the model’s predictions.

Recursive Feature Elimination (RFE):

* RFE is a technique that recursively removes the least important features from the model and refits it until a desired number of features is reached.

SelectKBest or SelectPercentile:

* These methods use statistical tests to select the top ‘k’ features with the highest scores or a certain percentage of the best features.

L1 Regularization (Lasso):

* L1 regularization can be used to penalize and shrink the coefficients of less important features to zero, effectively removing them from model

Domain Knowledge:

* Use your domain knowledge to eliminate irrelevant or redundant features. For example, if you know that a feature is not logically related to house prices, it can be safely removed.

Feature Engineering:

* Create new features that may capture important information, which can replace or complement existing features.

Cross-Validation:

* Evaluate different feature subsets using cross-validation to ensure that your feature selection doesn’t overfit the training data.

Automated Feature Selection:

* Tools like Recursive Feature Elimination with Cross-Validation (RFECV) and feature selection libraries can automate the process.

FEATURE SELECTION CODING :

# Import necessary libraries

Import pandas as pd

From sklearn.feature\_selection import SelectKBest, f\_regression

From sklearn.model\_selection import train\_test\_split

From sklearn.linear\_model import LinearRegression

From sklearn.metrics import mean\_squared\_error

# Load your dataset

Data = pd.read\_csv(‘house\_price\_data.csv’) # Replace with your dataset file

# Define your features (X) and target variable (y)

X = data.drop(‘house\_price’, axis=1) # Adjust the target variable name

Y = data[‘house\_price’]

# Split the data into training and testing sets

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

# Feature selection using SelectKBest

K = 5 # Define the number of top features to select

Selector = SelectKBest(score\_func=f\_regression, k=k)

X\_new = selector.fit\_transform(X\_train, y\_train)

# Get the indices of the selected features

Selected\_features\_indices = selector.get\_support(indices=True)

# Get the names of the selected features

Selected\_features = X.columns[selected\_features\_indices]

# Now, you can use the selected features for modeling

X\_train\_selected = X\_train[selected\_features]

X\_test\_selected = X\_test[selected\_features]

# Train a model using the selected features

Model = LinearRegression()

Model.fit(X\_train\_selected, y\_train)

# Make predictions on the test set

Y\_pred = model.predict(X\_test\_selected)

# Evaluate the model’s performance

Mse = mean\_squared\_error(y\_test, y\_pred)

Print(f”Mean Squared Error: {mse}”)

MODEL TRAINING :

Choose a machine learning algorithm. There are a number of Different machine learning algorithms that can be used for house price Prediction, such as linear regression, ridge regression, lasso regression,Decision trees, and random forests are Covered above.

Machine Learning Models:

* The machine learning model is given the test data but without the price of the properties in order to predict the price for them given the various features for the properties. The predicted price is then compared to the actual price in the test data.

Models = pd.DataFrame(columns=[“Model”,”MAE”,”MSE”,”RMSE”,”R2S

Core”,”RMSE (Cross-Validation)”]

LINEAR REGRESSION

From sklearn.linear\_model import LinearRegression

# Create a Linear Regression model

Model = LinearRegression()

# Fit the model to your data

Model.fit(X, y) # X is your independent variable(s), y is your dependent variable

# Make predictions

Y\_pred = model.predict(X\_new) # X\_new is a new set of data for prediction

# Access the coefficients and intercept

B0 = model.intercept\_

B1 = model.coef\_

RIDGE REGRESSION :

Ridge regression is a linear regression technique that extends ordinary least squares (OLS) regression by adding a regularization term to the cost function. It’s used to address the issue of multicollinearity (high correlation between independent variables) and can help prevent overfitting in regression models. Ridge regression is also known as L2 regularization.In Ridge regression, the cost function is modified to include a regularization term that penalizes the magnitude of the coefficients. The modified cost function is as fofollow.

RIDGE REGRESSION CODING :

From sklearn.linear\_model import LinearRegression

# Create a Linear Regression model

Model = LinearRegression()

# Fit the model to your data

Model.fit(X, y) # X is your independent variable(s), y is your dependent variable

# Make predictions

Y\_pred = model.predict(X\_new) # X\_new is a new set of data for prediction

# Access the coefficients and intercept

B0 = model.intercept\_

B1 = model.coef\_

ELASTIC NET

* Elastic Net is useful when you want to select a subset of important features (as Lasso does) while also addressing multicollinearity issues (as Ridge does). The choice of the (\alpha) and (\lambda) parameters allows you to fine-tune the balance between the two types of regularization.In Python, you can implement Elastic Net using scikit-learn:

ELASTIC NET USING PYTHON CODING :

From sklearn.linear\_model import ElasticNet

# Create an Elastic Net model with specified alpha and lambda (l1\_ratio) values

Model = ElasticNet(alpha=1.0, l1\_ratio=0.5) # You can adjust the alpha and l1\_ratio values

# Fit the model to your data

Model.fit(X, y) # X is your independent variable(s), y is your dependent variable

# Make predictions

Y\_pred = model.predict(X\_new) # X\_new is a new set of data for prediction

# Access the coefficients

Coefficients = model.coef\_

SUPPORT VECTOR MACHINES

SVR model for stock prediction

In this study, support vector regression (SVR) analysis is used as a machine learning technique in order to predict the stock market price as well as to predict stock market trend. Moreover, different types of windowing operators are used as data preprocess or input selection technique for SVR models.

PYTHON CODING:

From sklearn import svm

# Create an SVM classifier (for classification tasks)

Clf = svm.SVC(kernel=’linear’, C=1.0)

# Fit the model to your training data

Clf.fit(X\_train, y\_train)

# Make predictions on new data

Y\_pred = clf.predict(X\_test)

# Evaluate the model’s performance

From sklearn.metrics import accuracy\_score

Accuracy = accuracy\_score(y\_test, y\_pred)

RANDOM FOREST REGRESSORS :

* The aim of this project is to predict house prices using one basic machine learning algorithm, Linear Regression, and one advanced algorithm, Random Forest. We will also use regression with regularization such as Ridge and Lasso to try to improve our prediction accuracy.
* The second approach that involves the use of Random forest algorithm was adopted into his study. Over the years, machine learning techniques have been greatly explored for price prediction. The results obtained have shown the predictive prowess of machine learning algorithm.
* That you have 100% train and test accuracy probably means that your model is massively overfitting because of your amount of data.

INPUT :

Random\_forest = RandomForestRegressor(n\_estimators=100)random\_forest. Fit(X\_train, y\_train)predictions = random\_forest.predict(X\_test)

Mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

Print(“MAE:”, mae)

Print(“MSE:”, mse)

Print(“RMSE:”, rmse)

Print(“R2 Score:”, r\_squared)

Print(“-“\*30)rmse\_cross\_val = rmse\_cv(random\_forest)

Print(“RMSE Cross-Validation:”, rmse\_cross\_val)

New\_row = {“Model”: “RandomForestRegressor”,”MAE”: mae, “MSE”: ms

E, “RMSE”: rmse, “R2 Score”: r\_squared, “RMSE (Cross-Validation)”: rms

E\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Output :

MAE: 18115.11067351598

MSE: 1004422414.0219476

RMSE: 31692.623968708358

R2 Score: 0.869050886899595

------------------------------ RMSE Cross-Validation: 31138.863315259332

XGBoost Regressor:

INPUT:

Xgb = XGBRegressor(n\_estimators=1000, learning\_rate=0.01)xgb.fit(X\_trai

N, y\_train)predictions = xgb.predict(X\_test)

Mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

Print(“MAE:”, mae)

Print(“MSE:”, mse)

Print(“RMSE:”, rmse)

Print(“R2 Score:”, r\_squared)

Print(“-“\*30)rmse\_cross\_val = rmse\_cv(xgb)

Print(“RMSE Cross-Validation:”, rmse\_cross\_val)

New\_row = {“Model”: “XGBRegressor”,”MAE”: mae, “MSE”: mse, “RMS

E”: rmse, “R2 Score”: r\_squared, “RMSE (Cross-Validation)”: rmse\_cross\_

Val}models = models.append(new\_row, ignore\_index=True)

Output:

MAE: 17439.918396832192

MSE: 716579004.5214689

RMSE: 26768.993341578403

R2 Score: 0.9065777666861116

------------------------------ RMSE Cross-Validation: 29698.84961808251

Polynomial Regression (Degree=2)

INPUT :

Poly\_reg = PolynomialFeatures(degree=2)X\_train\_2d = poly\_reg.fit\_transfo

Rm(X\_train)X\_test\_2d = poly\_reg.transform(X\_test)

Lin\_reg = LinearRegression()lin\_reg.fit(X\_train\_2d, y\_train)predictions = li

N\_reg.predict(X\_test\_2d)

Mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

Print(“MAE:”, mae)

Print(“MSE:”, mse)

Print(“RMSE:”, rmse)

Print(“R2 Score:”, r\_squared)

Print(“-“\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

Print(“RMSE Cross-Validation:”, rmse\_cross\_val)

New\_row = {“Model”: “Polynomial Regression (degree=2)”,”MAE”: mae, “ MSE”: mse, “RMSE”: rmse, “R2 Score”: r\_squared, “RMSE (Cross-Validat

Ion)”: rmse\_cross\_val}models = models.append(new\_row, ignore\_index=Tr

Ue)

Output :

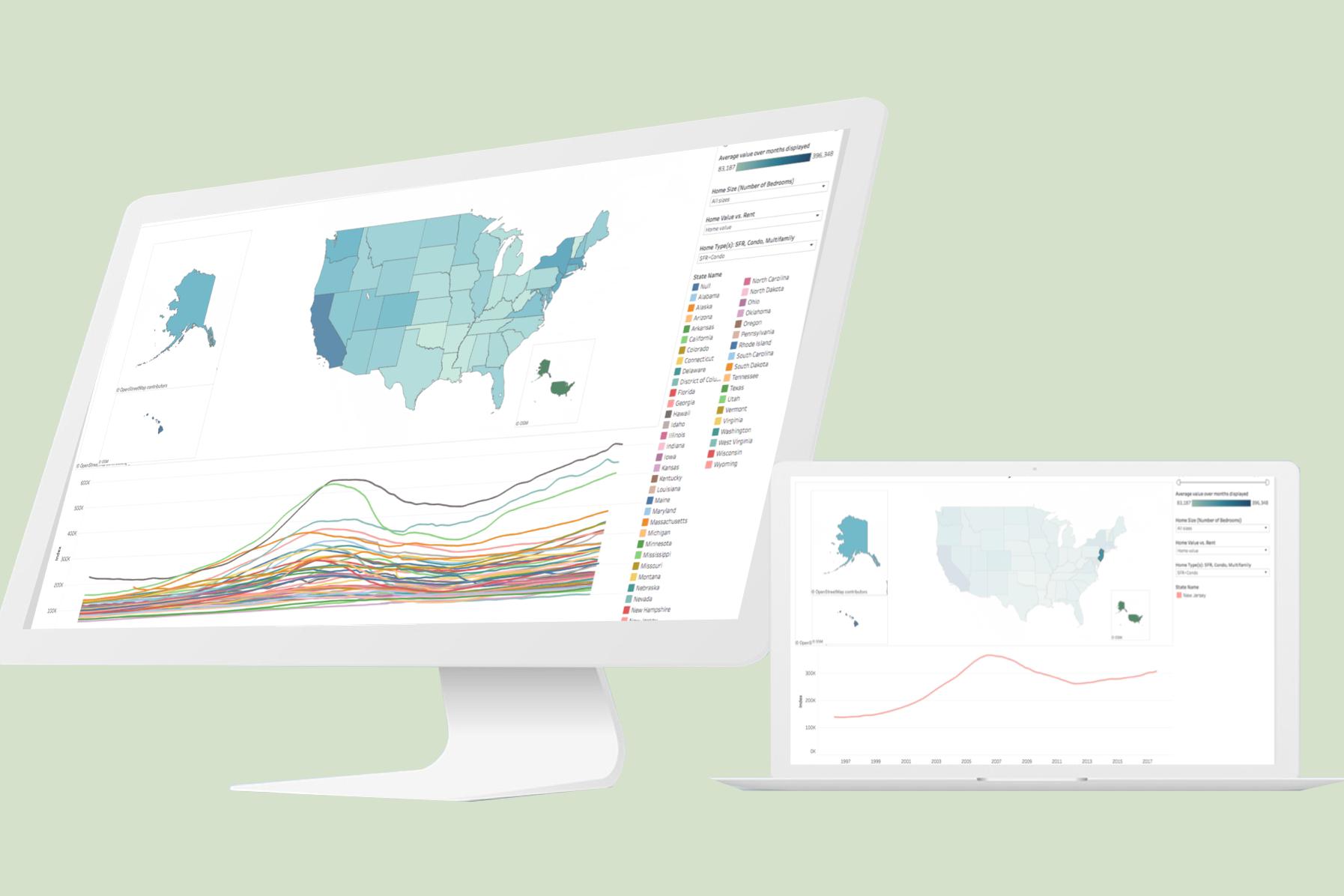
MAE: 2382228327828308.5

MSE: 1.5139911544182342e+32

RMSE: 1.230443478758059e+16

R2 Score: -1.9738289005226644e+22

------------------------------ RMSE Cross-Validation: 36326.451444669496

 FEATURE ENGINEERING :

* Feature Selection:

Identify and select the most relevant features that have the most impact on the target variable. You can use techniques like correlation analysis, mutual information, or feature importance from tree-based models.

* Feature Extraction:

Transform existing features into more informative representations. Common techniques include:Principal Component Analysis (PCA) to reduce dimensionality.

Polynomial features to capture non-linear relationships.Text vectorization methods (e.g., TF-IDF or Word2Vec) for textual data.Feature scaling to normalize features to the same scale.

* Handling Categorical Data:

Convert categorical variables into a numerical format suitable for machine learning. Options include one-hot encoding, label encoding, and target encoding.

* Dealing with Missing Data:

Decide on a strategy for handling missing values. You can remove rows with missing values, impute missing values with mean/median/mode, or use advanced imputation techniques.

* Temporal Features:

Extract meaningful features from time-related data, such as day of the week, month, or time elapsed since a specific event.

* Domain-Specific Features:

Incorporate domain knowledge to create features that are relevant to the specific problem you are solving.

* Feature Scaling:

Normalize or standardize features to ensure that they have the same scale. This is crucial for models that rely on distance-based calculations (e.g., k-means clustering or SVM).

* Feature Interaction:

Create new features by combining or interacting existing features. For example, if you have features for “length” and “width,” you can create a feature for “area.”

* Binning or Discretization:

Convert continuous numerical features into discrete bins or categories. This can help capture non-linear relationships.

* Outlier Handling:

Consider creating binary features that indicate whether a data point is an outlier based on domain knowledge or statistical methods.

* Feature Scaling:

Ensure that all features have the same scale, especially when using models like k-nearest neighbors or support vector machines. Common scaling techniques include Min-Max scaling or Z-score normalization.

* Feature Aggregation:

Aggregate data over time periods or categories, creating summary statistics like mean, sum, or count.

* Feature Crosses:

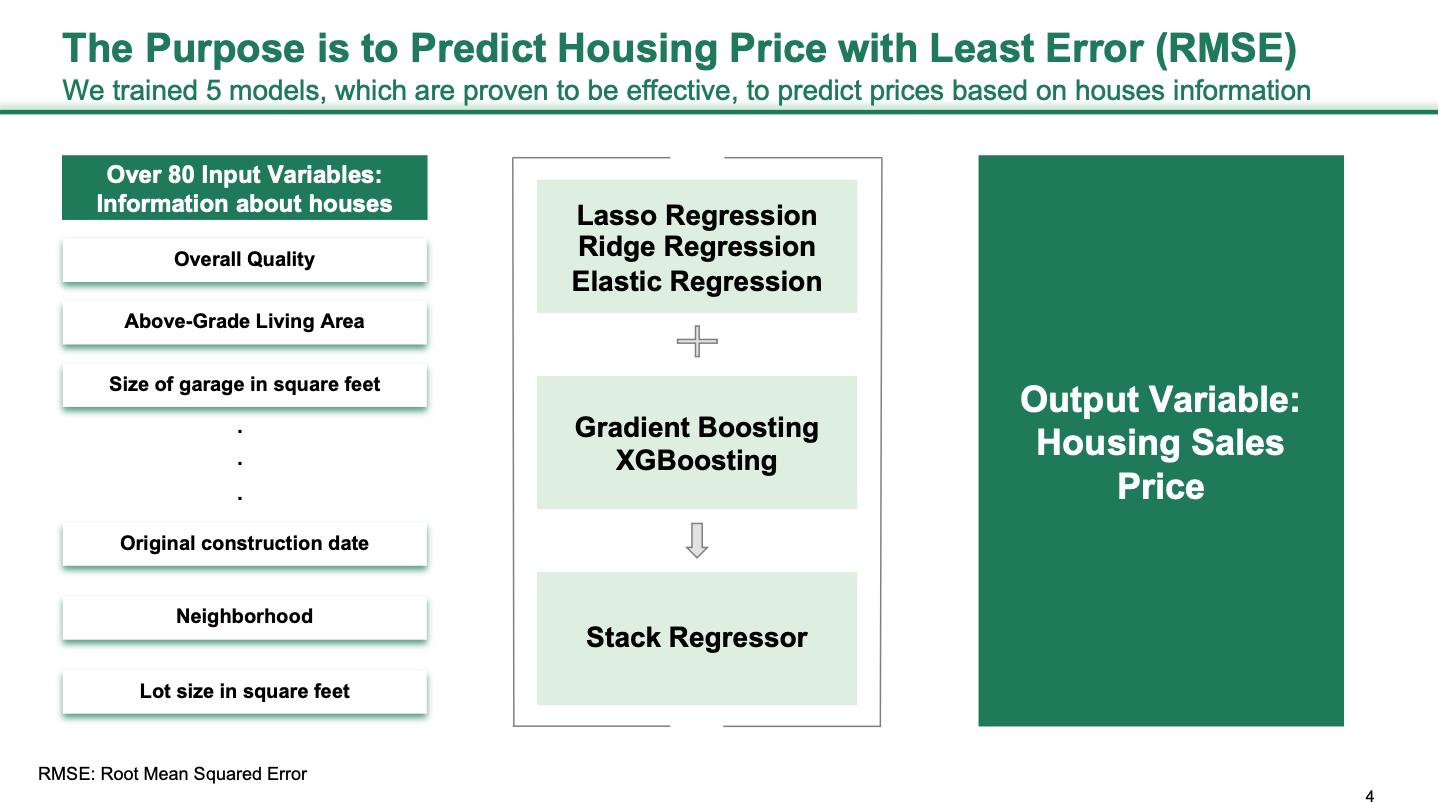
For tabular data, you can combine two or more features to capture interactions between them. This is particularly useful in deep learning models.

PREDICT HOUSING PRICE LEAST ERROR ( RMSE)

-Predicting housing prices with the least Root Mean Square Error (RMSE) involves using a regression model and optimizing it to minimize prediction errors. Here are some steps to achieve this:

-Regular Updates:

Continuously update your model as new data becomes available to maintain its accuracy over time.Remember that predicting housing prices accurately is a complex task and achieving the lowest RMSE might require experimentation and fine-tuning.



**Model training:**

* Model training is the process of teaching a machine learning model To predict house prices. It involves feeding the model historical data On house prices and features, such as square footage, number of Bedrooms, and location. The model then learns the relationships Between these features and house prices.
* Once the model is trained, it can be used to predict house prices for New data. For example, you could use the model to predict the price Of a house that you are interested in buying.
* Prepare the data. This involves cleaning the data, removing any Errors or inconsistencies, and transforming the data into a format that is Compatible with the machine learning algorithm that you will be using.
* Split the data into training and test sets. The training set will be Used to train the model, and the test set will be used to evaluate The performance of the model on unseen data.
* Choose a machine learning algorithm. There are a number of Different machine learning algorithms that can be used for house price Prediction, such as linear regression, ridge regression, lasso regression, Decision trees, and random forests.
* Tune the hyperparameters of the algorithm. The Hyperparameters of a machine learning algorithm are parameters that Control the learning process. It is important to tune the hyperparameters Of the algorithm to optimize its performance.

Dividing Dataset in to features and target variable:

X = dataset[[‘Avg. Area Income’, ‘Avg. Area House Age’, ‘Avg. Area Number of

Rooms’, ‘Avg. Area Number of Bedrooms’, ‘Area Population’]]

Y = dataset[‘Price’]

Split the data into training and test sets. The training set will be

Used to train the model, and the test set will be used to evaluate

The performance of the model.

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_st

Ate=101)

IN :

Y\_train.head()

Out:

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64n

Train the model on the training set. This involves feeding the Training data to the model and allowing it to learn the relationships Between the features and the target variable.

Evaluate the model on the test set. This involves feeding the test Data to the model and measuring how well it predicts the target variable.

MODEL EVALUATION :

* Data Split: Divide your dataset into two or three parts: a training set, a validation set, and a test set. A common split might be 70% for training, 15% for validation, and 15% for testing.
* Model Selection: Choose the appropriate regression model for your task. Common choices include linear regression, decision trees, random forests, or neural networks.
* Training: Train the model on the training data. Monitor its performance on the validation set during training to detect overfitting.
* Performance Metrics: Use appropriate metrics to evaluate your model. Common regression metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Additionally, you can calculate the R-squared (R2) value to measure the proportion of variance explained by the model.
* Mean squared error (MSE): This metric measures the average Squared difference between the predicted and actual house prices.  Root mean squared error (RMSE): This metric is the square root Of the MSE.
* Mean absolute error (MAE): This metric measures the average Absolute difference between the predicted and actual house prices.  R-squared: This metric measures how well the model explains the Variation in the actual house prices. In addition to these metrics, it is also important to consider the following Factors when evaluating a house price prediction model:
* Bias: Bias is the tendency of a model to consistently over- or Underestimate house prices.
* Variance: Variance is the measure of how much the predictions of A model vary around the true house prices.
* Interpretability: Interpretability is the ability to understand how The model makes its predictions. This is important for house price Prediction models, as it allows users to understand the factors that Influence the predicted house prices.

MODEL EVALUATION :

# Import necessary libraries

Import pandas as pd

Import numpy as np

From sklearn.model\_selection import train\_test\_split

From sklearn.linear\_model import LinearRegression

From sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

Import matplotlib.pyplot as plt

# Load your dataset

Data = pd.read\_csv(‘your\_house\_price\_data.csv’) # Replace with your dataset

# Data preprocessing

X = data[[‘feature1’, ‘feature2’, ‘feature3’]] # Features

Y = data[‘price’] # Target variable

# Split the data into training and test sets

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

# Train the model

Model = LinearRegression()

Model.fit(X\_train, y\_train)

# Predict house prices on the test set

Y\_pred = model.predict(X\_test)

# Model evaluation

Mae = mean\_absolute\_error(y\_test, y\_pred)

Mse = mean\_squared\_error(y\_test, y\_pred)

Rmse = np.sqrt(mse)

R2 = r2\_score(y\_test, y\_pred)

# Print the evaluation metrics

Print(“Mean Absolute Error: “, mae)

Print(“Mean Squared Error: “, mse)

Print(“Root Mean Squared Error: “, rmse)

Print(“R-squared: “, r2)

# Visualization: Scatter plot of predicted vs. Actual prices

Plt.scatter(y\_test, y\_pred)

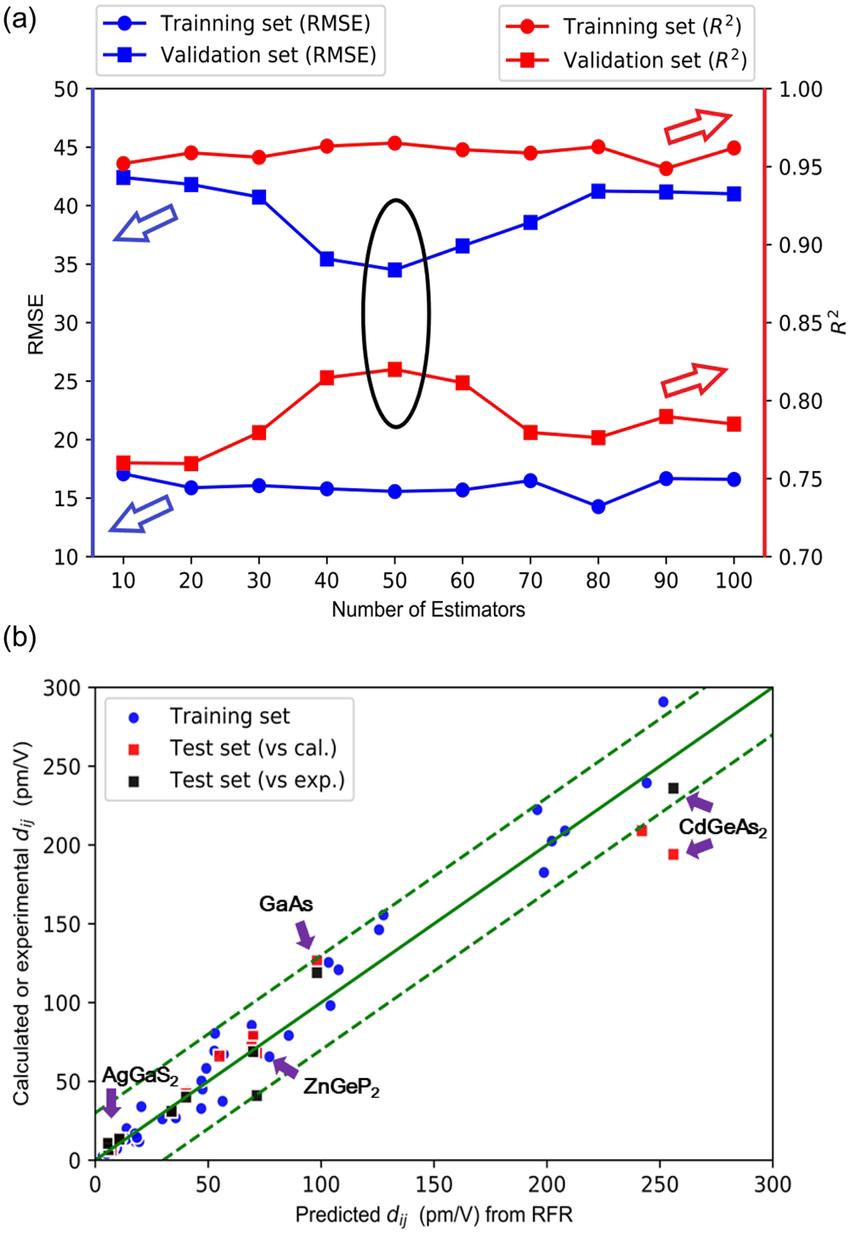
Plt.xlabel(“Actual Prices”)

Plt.ylabel(“Predicted Prices”)

Plt.title(“Actual Prices vs. Predicted Prices”)

Plt.show()

OUTPUT:



Model Comparison:

* The less the Root Mean Squared Error (RMSE), The better the model ).Model compression refers to the process of reducing the size of a machine learning model while maintaining or minimizing its performance.
* This can be important for various reasons, including deploying models on resource-constrained devices, reducing memory and storage requirements, and improving model inference speed. Here are some techniques for model compression:

Quantization: Convert model weights and activations from high-precision floating-point numbers to lower-precision integers. For example, you can use quantization-aware training to train models for inference with reduced precision.

Pruning: Identify and eliminate unimportant or redundant model parameters (weights, neurons, filters) based on their importance scores. This can significantly reduce the model size while maintaining most of its accuracy.

Knowledge Distillation: Train a smaller model (student) to mimic the behavior of a larger model (teacher). The student model learns from the teacher’s soft predictions, allowing for the transfer of knowledge from a larger model to a smaller one.

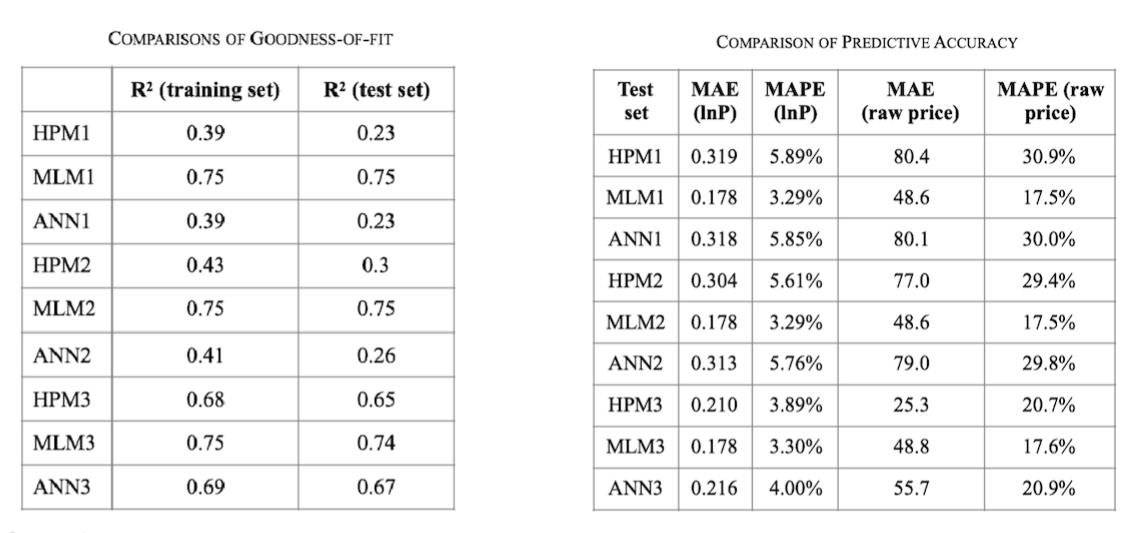
Low-Rank Factorization: Decompose weight matrices into low-rank factors, reducing the number of parameters and the computational cost of inference.

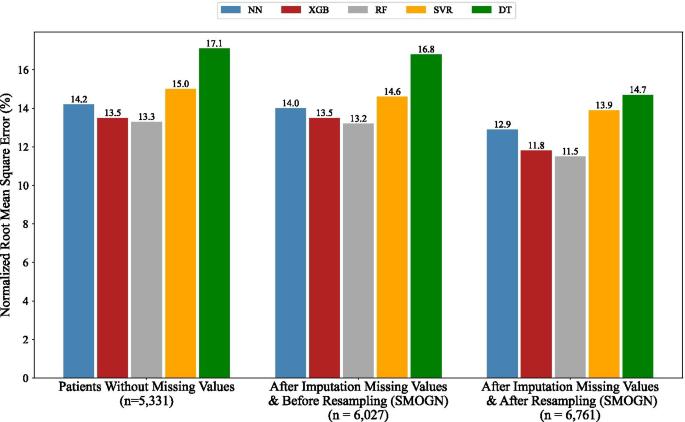
Model Quantization: Reduce the number of unique operations in a model’s architecture, which can help optimize inference on hardware accelerators.

Model Quantization with Compression Libraries: Use specialized libraries like TensorFlow Lite or ONNX Runtime to quantize and compress models for deployment on mobile or edge devices.

MODEL COMPARISON :

Models.sort\_values(by=”RMSE (Cross-Validation)”)





**CONCLUSION :**

* Improved Accuracy: Machine learning models consider a myriad of Variables, many of which may be overlooked by traditional methods.
* This results in more accurate predictions, benefiting both buyers and Sellers who can make informed decisions based on a property’s true

Value.

* Data-Driven Insights: These models provide valuable insights into the Real estate market by identifying trends, neighborhood characteristics,
* And other factors that influence property prices. This information can be Invaluable for investors, developers, and policymakers seeking to make

Strategic decisions.

* Market Efficiency: The increased accuracy in pricing predictions can Lead to a more efficient real estate market, reducing overvaluation and
* Undervaluation of properties. This contributes to a fairer and more Transparent marketplace.
* Challenges and Considerations: Machine learning for house price Prediction is not without its challenges. Data quality, modelInterpretability, and ethical concerns are important considerations.
* Addressing these issues is crucial for the responsible and ethical Deployment of this technology.