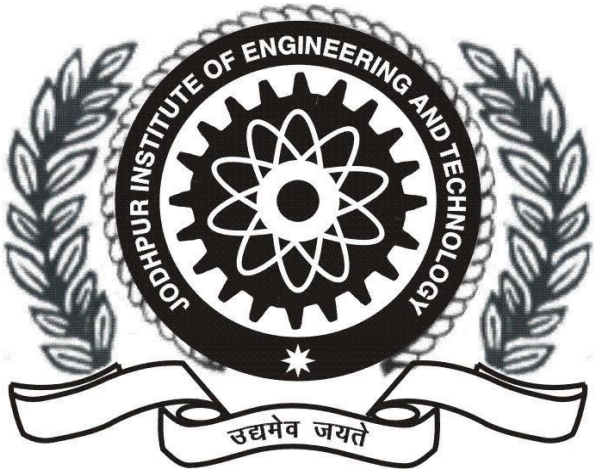
A

MACHINE LEARNING LAB

House Price Prediction Project Report

In partial fulfillment of

***B.Tech 3rd yr* (Computer Science & Engg.)**



Submitted by:
Ujjwal Kumar (22EJICS154)
Batch: 5C2

Submitted To:
Anamika Choudhary
Associate Professor

Acknowledgment

I would like to acknowledge the contributions of the following people without whose help and guidance this project would not have been completed. I respectfully thank Anamika Choudhary, for providing me an opportunity to do this project work and giving me all support and guidance, which made me complete the project up to very extent.

I am also thankful to Mamta Garg, HOD of Computer Science and Engineering Department, Jodhpur Institute of Engineering and Technology, for his/her constant encouragement, valuable suggestions and moral support and blessings.

Although it is not possible to name individually, I shall ever remain indebted to the faculty members of Jodhpur Institute of Engineering and Technology, for their persistent support and cooperation extended during his work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement.

Table of Contents

1. Introduction

1.1 Overview of the Project

Predicting house prices is a significant challenge in the real estate industry, with implications for buyers, sellers, and financial institutions alike. By leveraging historical data and machine learning techniques, it's possible to build models that can estimate the price of a house based on various attributes. These attributes typically include factors like the number of rooms, square footage, location, and other characteristics that influence the market value of a property .This project aims to develop a robust predictive model that can accurately estimate house prices. By using data from a specific real estate dataset, the goal is to analyze the factors that have the greatest impact on house prices and build a machine learning model to forecast prices for new, unseen properties.

1.2 Problem Statement

The objective of this project is to develop a machine learning model that can accurately predict house prices based on various features, such as square footage, number of bedrooms, location, and other property characteristics. Accurate price prediction is crucial for real estate professionals, buyers, and investors to make informed decisions. The challenge is to analyze the available data and build a model that generalizes well to unseen properties while identifying the key factors that influence house prices.

1.3 Objective of the Project

The objective of this project is to develop an accurate machine learning model to predict house prices based on various property features. This includes analyzing key factors that influence house prices, preprocessing the data, building and evaluating predictive models, and providing insights into which features most impact property values.

2. Technology used in Project

2.1 Python

Python: Used for data analysis, preprocessing, and building machine learning models due to its extensive libraries and tools.

2.2 Libraries Used

- **Pandas**: For data manipulation and analysis.
- **NumPy**: For numerical computations and array handling.
- **Matplotlib/**: For data visualization and exploratory data analysis (EDA).
- **Scikit-learn**: For building and evaluating machine learning models, including regression and tree-based models.

Development Environment:

- **Integrated Development Environment (IDE)**: Visual Studio Code, PyCharm, or Jupyter for writing and testing code.
- **Version Control**: Git for tracking changes and collaboration.

Details of Project

3.1 Functions/Modules Details

The project consists of several Python modules and functions that together provide the functionality of the house price prediction:

The methodology involves the following steps:

Step 1: Data Preprocessing

- Load the dataset using pandas.
- Display the first few rows to verify the data integrity.
- Check for missing values and handle them appropriately.

Step 2: Feature Engineering

- Define features (X) by dropping the target variable (Price).
- Define the target variable (y) as the Price.

Step 3: Data Scaling

- Standardize the features using Standard Scaler to ensure that all features contribute equally to the distance calculations in algorithms like linear regression.

Step 4: Data Splitting

- Split the dataset into training (80%) and testing (20%) sets using train_test_split.

Step 5: Feature Selection

- Use Recursive Feature Elimination (RFE) with Linear Regression to select the most important features for model training.

Step 6: Model Training

- Initialize and train the following models:
  - Linear Regression
  - Ridge Regression (with regularization)
  - Decision Tree Regressor
  - Random Forest Regressor (with multiple trees)

Step 7: Model Evaluation

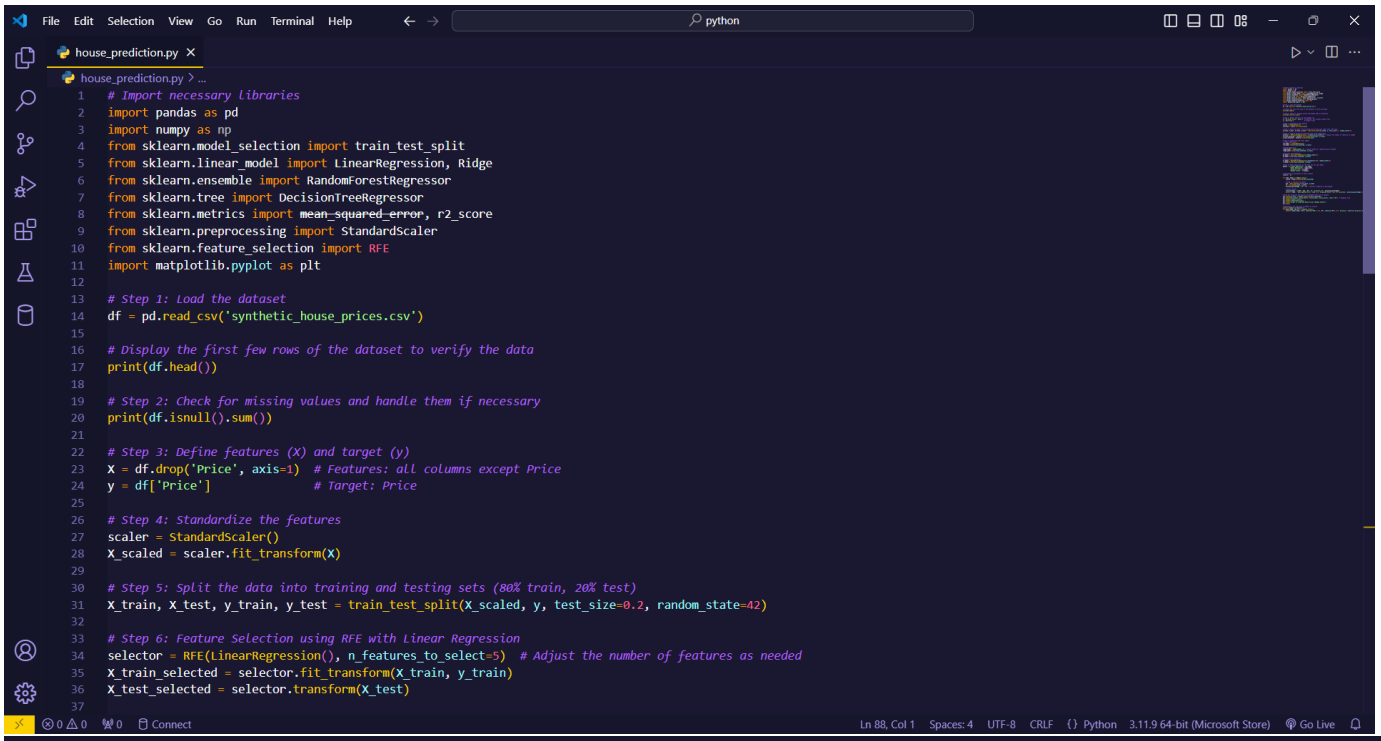
- For each model, make predictions on the test set.
- Calculate performance metrics:
  - Mean Squared Error (MSE)
  - R-squared Score (R^2)
  - Accuracy Percentage (derived from R^2)

Step 8: Visualization

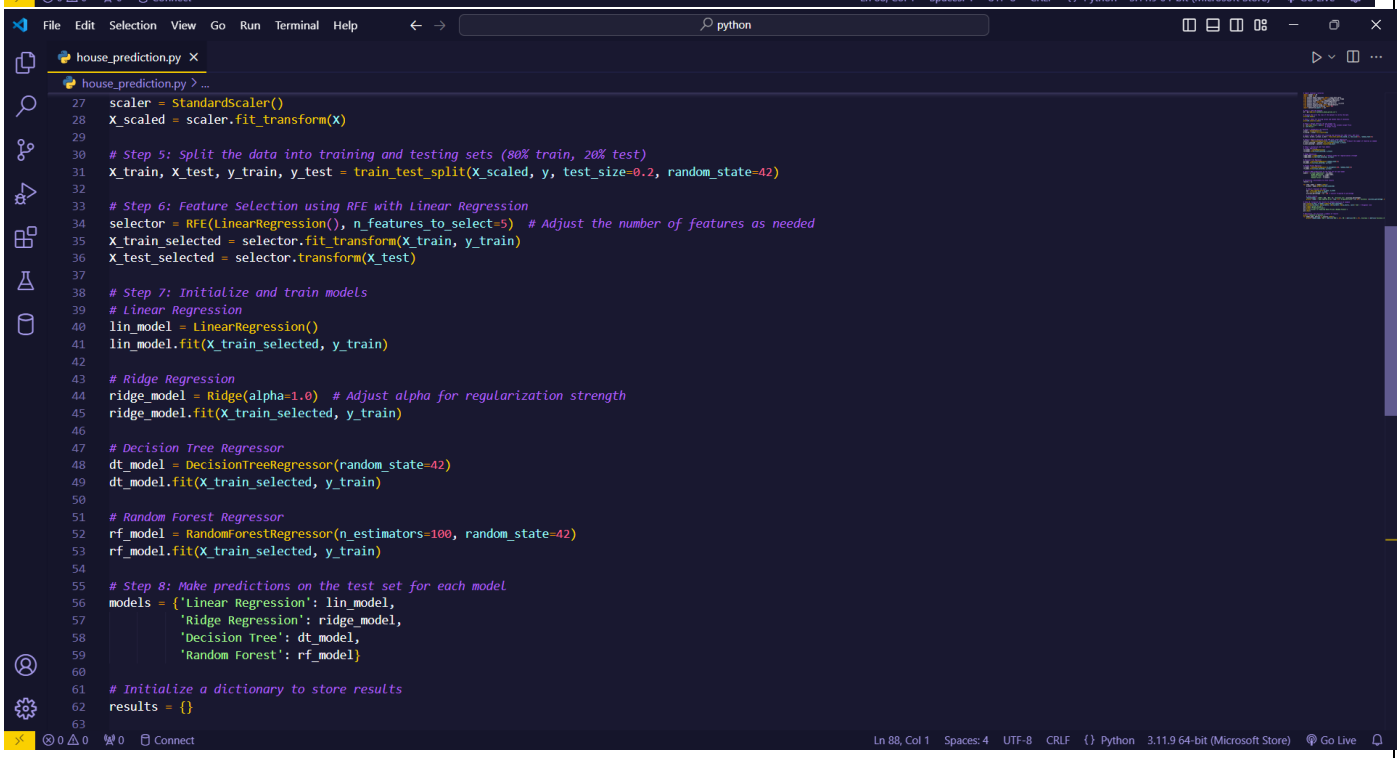
- Plot actual vs. predicted house prices for one of the models (e.g., Random Forest) to visually assess performance.

3.2Project Screenshots

Include screenshots of the application showcasing the user interface, resume upload functionality, and filtering results. Example screenshots could include:



```python
# 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, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt

# Step 1: Load the dataset
df = pd.read_csv('synthetic_house_prices.csv')

# Display the first few rows of the dataset to verify the data
print(df.head())

# Step 2: Check for missing values and handle them if necessary
print(df.isnull().sum())

# Step 3: Define features (X) and target (y)
X = df.drop('Price', axis=1)  # Features: all columns except Price
y = df['Price']               # Target: Price

# Step 4: Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Step 6: Feature Selection using RFE with Linear Regression
selector = RFE(LinearRegression(), n_features_to_select=5)  # Adjust the number of features as needed
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)
```



```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Step 6: Feature Selection using RFE with Linear Regression
selector = RFE(LinearRegression(), n_features_to_select=5)  # Adjust the number of features as needed
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

# Step 7: Initialize and train models
# Linear Regression
lin_model = LinearRegression()
lin_model.fit(X_train_selected, y_train)

# Ridge Regression
ridge_model = Ridge(alpha=1.0)  # Adjust alpha for regularization strength
ridge_model.fit(X_train_selected, y_train)

# Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train_selected, y_train)

# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_selected, y_train)

# Step 8: Make predictions on the test set for each model
models = {'Linear Regression': lin_model,
          'Ridge Regression': ridge_model,
          'Decision Tree': dt_model,
          'Random Forest': rf_model}

# Initialize a dictionary to store results
results = {}
```

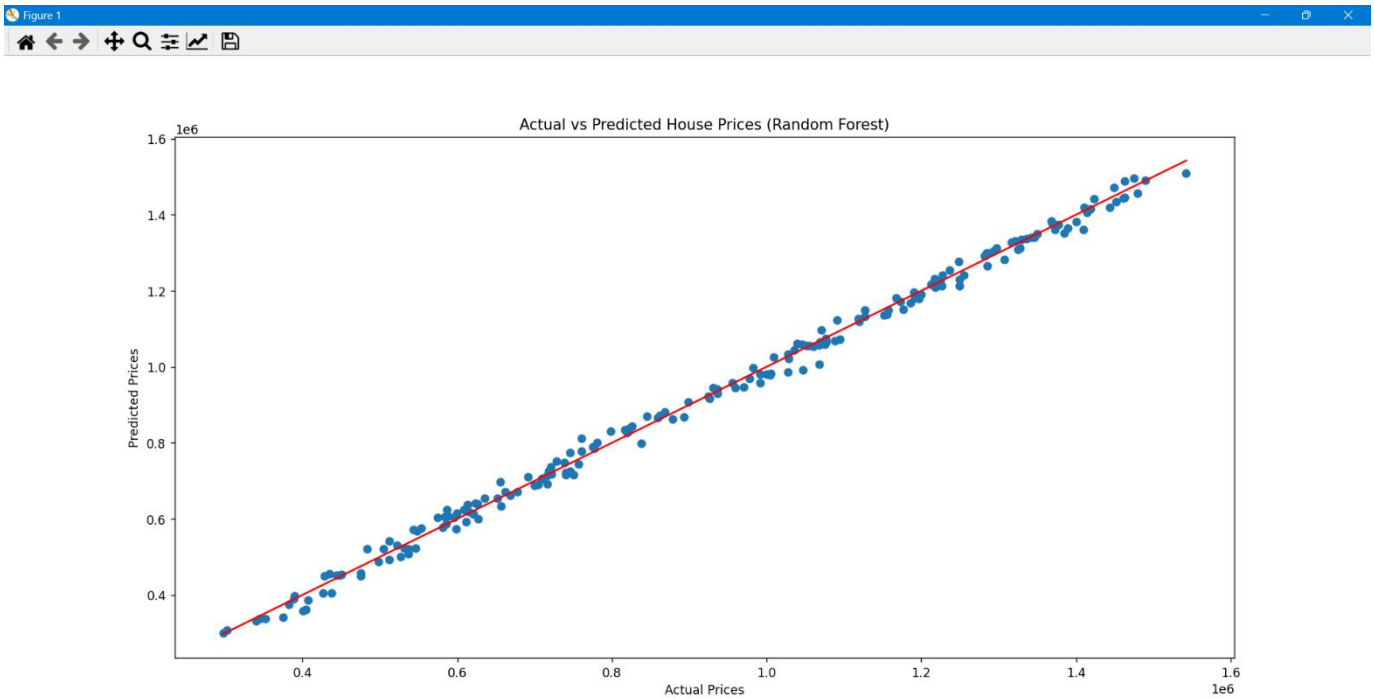
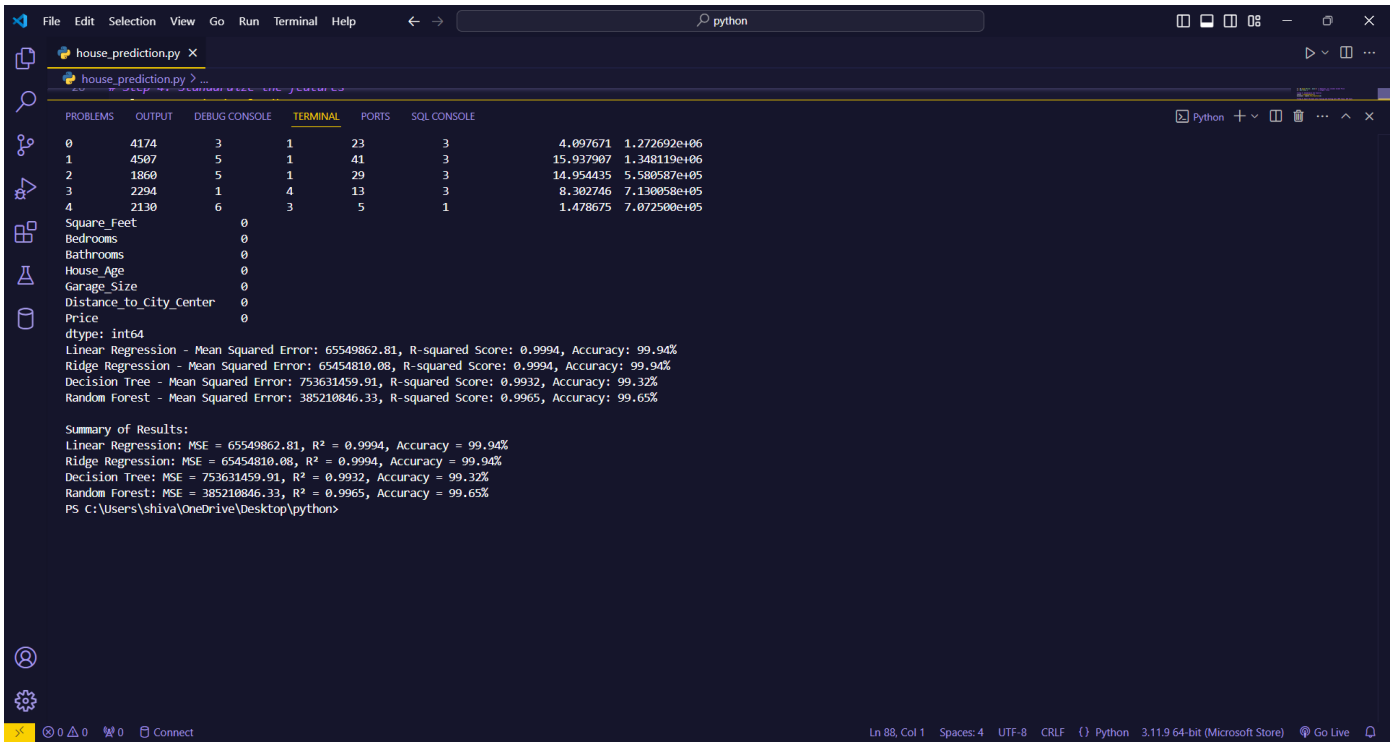


Figure 1
Actual vs Predicted House Prices (Random Forest)
1e6
Predicted Prices
Actual Prices
1e6
0.4
0.6
0.8
1.0
1.2
1.4
1.6


File Edit Selection View Go Run Terminal Help
python
house_prediction.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
Python
0 4174 3 1 23 3 4.097671 1.272692e+06
1 4507 5 1 41 3 15.937907 1.348119e+06
2 1860 5 1 29 3 14.954435 5.580587e+05
3 2294 1 4 13 3 8.302746 7.130058e+05
4 2130 6 3 5 1 1.478675 7.072500e+05
Square_Feet 0
Bedrooms 0
Bathrooms 0
House_Age 0
Garage_Size 0
Distance_to_City_Center 0
Price 0
dtype: int64
Linear Regression - Mean Squared Error: 65549862.81, R-squared Score: 0.9994, Accuracy: 99.94%
Ridge Regression - Mean Squared Error: 65454810.08, R-squared Score: 0.9994, Accuracy: 99.94%
Decision Tree - Mean Squared Error: 753631459.91, R-squared Score: 0.9932, Accuracy: 99.32%
Random Forest - Mean Squared Error: 385210846.33, R-squared Score: 0.9965, Accuracy: 99.65%
Summary of Results:
Linear Regression: MSE = 65549862.81, R² = 0.9994, Accuracy = 99.94%
Ridge Regression: MSE = 65454810.08, R² = 0.9994, Accuracy = 99.94%
Decision Tree: MSE = 753631459.91, R² = 0.9932, Accuracy = 99.32%
Random Forest: MSE = 385210846.33, R² = 0.9965, Accuracy = 99.65%
PS C:\Users\shiva\OneDrive\Desktop\python>
Connect
Ln 88, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store) Go Live

3. Applications

This tool can be applied in several domains:

- **Recruitment Agencies**: To automate resume screening and improve the efficiency of candidate selection processes.
- **HR Departments**: Companies can use the tool internally to sift through resumes for open positions, saving time and reducing human errors in screening.
- **Job Portals**: Integrated into online job portals, this tool can help employers find suitable candidates by automatically analyzing uploaded resumes.
- **Freelance Marketplaces**: Platforms like Upwork or Fiverr can use this tool to help clients quickly identify freelancers with specific skills.

5.Future Work and Conclusion

Conclusion:

This project demonstrates the application of various regression techniques for predicting house prices. The results highlight the effectiveness of linear models for this dataset, while also providing insights into more complex methods like Random Forest. Future improvements could enhance model performance and applicability.

Future Work:

7. Future Work

- **Hyperparameter Tuning**: Use techniques like Grid Search or Random Search to optimize the parameters of the Decision Tree and Random Forest models for improved performance.
- **Cross-Validation**: Implement k-fold cross-validation to provide a more robust evaluation of model performance.
- **Additional Models**: Experiment with other algorithms like Gradient Boosting, Support Vector Regression (SVR), or Neural Networks to compare performance.
- **Deploying the Model**: Consider building a web application for end-users to input features and receive price predictions.

6. References

- **Datasets**
  - Kaggle. (n.d.). House Prices: Advanced Regression Techniques. Retrieved from Kaggle.
- **Machine Learning Frameworks and Libraries**
  - Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. Retrieved from http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
- **Regression Analysis**
  - James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.
- **Feature Selection**
  - Guyon, I. A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*