Here is a structured outline for a \*House Price Prediction System\* project using Python. I've placed the necessary information and steps under each heading for your reference.

---

Title of Project:

House Price Prediction Using Machine Learning

objective:

The objective of this project is to develop a machine learning model that predicts house prices based on various features such as location, size, number of bedrooms, bathrooms, and more. This will help in estimating house prices based on existing market data.

---

Data Source:

- Kaggle Housing Prices Dataset

- Zillow Open Data

- Real Estate Listings API

- Local Property Databases

---

#Import Library

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import StandardScaler

#Import Data:

# Example: Load a dataset from a CSV file

house\_data = pd.read\_csv('house\_data.csv')

#Describe Data:

# Overview of the dataset

print(house\_data.info())

print(house\_data.describe())

print(house\_data.head())

# Check for missing values

print(house\_data.isnull().sum())

\*Explanation\*: The dataset may include columns like house size (sqft), number of bedrooms, bathrooms, location, year built, etc. Understanding missing data and the structure is crucial.

---

### \*Data Visualization:\*

python

# Visualizing the correlation between different features

plt.figure(figsize=(10, 8))

sns.heatmap(house\_data.corr(), annot=True, cmap='coolwarm')

plt.title('Feature Correlation Heatmap')

plt.show()

# Distribution of house prices

sns.histplot(house\_data['price'], bins=50, kde=True)

plt.title('Distribution of House Prices')

plt.show()

# Visualizing relationship between house size and price

sns.scatterplot(x='sqft\_living', y='price', data=house\_data)

plt.title('House Size vs Price')

plt.show()

---

### \*Data Preprocessing:\*

python

# Handling missing values

house\_data.fillna(method='ffill', inplace=True)

# Dropping irrelevant columns

house\_data = house\_data.drop(['id', 'date', 'zipcode'], axis=1)

# Converting categorical columns (if any) using one-hot encoding

house\_data = pd.get\_dummies(house\_data, columns=['waterfront', 'condition', 'grade'], drop\_first=True)

# Feature scaling (especially for numerical features like sqft, lot size)

scaler = StandardScaler()

scaled\_columns = ['sqft\_living', 'sqft\_lot', 'sqft\_above', 'sqft\_basement']

house\_data[scaled\_columns] = scaler.fit\_transform(house\_data[scaled\_columns])

#Define Target Variable (y) and Feature Variables (X):

# Defining features and target variable

X = house\_data.drop('price', axis=1)

y = house\_data['price']

#Train Test Split:

# Splitting the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

#Modeling:

1. Linear Regression:

# Linear Regression Model

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

# Predictions

y\_pred\_lr = lr\_model.predict(X\_test)

2. \*Random Forest Regressor:\*

# Random Forest Regressor Model

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Predictions

y\_pred\_rf = rf\_model.predict(X\_test)

#Model Evaluation:

1. Linear Regression Evaluation:

# Mean Squared Error & R-squared for Linear Regression

mse\_lr = mean\_squared\_error(y\_test, y\_pred\_lr)

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

print(f"Linear Regression MSE: {mse\_lr}")

print(f"Linear Regression R2 Score: {r2\_lr}")

2. Random Forest Evaluation:

python

# Mean Squared Error & R-squared for Random Forest

mse\_rf = mean\_squared\_error(y\_test, y\_pred\_rf)

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

print(f"Random Forest MSE: {mse\_rf}")

print(f"Random Forest R2 Score: {r2\_rf}")

#Prediction:

# Predict the price for a new house with certain features

new\_house = np.array([[4, 3, 2000, 5000, 1000, 1000, 2, 1, 7, 2020]]) # Example features: Bedrooms, Bathrooms, Sqft, etc.

predicted\_price = rf\_model.predict(new\_house)

print(f"Predicted Price for the new house: ${predicted\_price[0]}")

Explanation:

1. \*Linear Regression\*: Simple model but might not perform well if the relationships are non-linear.

2. \*Random Forest Regressor\*: Generally performs better as it captures non-linear interactions between features and can provide feature importance.

3. \*Feature Importance\*: Random Forest offers insights into which features contribute most to the house price predictions.

# Feature importance from the Random Forest model

importances = rf\_model.feature\_importances\_

feature\_names = X.columns

sorted\_importances = sorted(zip(importances, feature\_names), reverse=True)

for importance, feature in sorted\_importances:

print(f"Feature: {feature}, Importance: {importance}")

This outline provides the foundation for a \*House Price Prediction System\* using Python, which can be customized depending on the dataset and requirements of the project.