Forecasting house prices accurately using smart regression techniques in data science

import numpy as np

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

from sklearn.preprocessing import StandardScaler

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

import xgboost as xgb

import matplotlib.pyplot as plt

import seaborn as sns

# 1. Load Data (replace with your actual dataset)

# For demonstration, use a sample from seaborn

from sklearn.datasets import fetch\_california\_housing

data = fetch\_california\_housing()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['Price'] = data.target

# 2. Explore Dataset

print(df.head())

# 3. Preprocessing

X = df.drop('Price', axis=1)

y = df['Price']

# Train-test split

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

# Feature scaling (important for some models, optional for XGBoost)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# 4. Train XGBoost Regressor

xgb\_model = xgb.XGBRegressor(

n\_estimators=100,

learning\_rate=0.1,

max\_depth=5,

subsample=0.8,

colsample\_bytree=0.8,

random\_state=42

)

xgb\_model.fit(X\_train\_scaled, y\_train)

# 5. Predictions

y\_pred = xgb\_model.predict(X\_test\_scaled)

# 6. Evaluation

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

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

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

print(f"RMSE: {rmse:.3f}")

print(f"MAE: {mae:.3f}")

print(f"R² Score: {r2:.3f}")

# 7. Feature Importance Plot

xgb.plot\_importance(xgb\_model)

plt.title("Feature Importance")

plt.show()