#forecasting\_house\_prices.py

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, GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.linear\_model import LinearRegression

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

# Load dataset (example: Kaggle's House Prices dataset)

data = pd.read\_csv('house\_prices.csv')

# Drop columns with too many missing values

data = data.dropna(thresh=data.shape[0]\*0.8, axis=1)

# Fill missing values for numerical and categorical separately

for col in data.select\_dtypes(include='number'):

data[col] = data[col].fillna(data[col].median())

for col in data.select\_dtypes(include='object'):

data[col] = data[col].fillna(data[col].mode()[0])

# Encode categorical features

data = pd.get\_dummies(data, drop\_first=True)

# Define features and target

X = data.drop('SalePrice', axis=1)

y = data['SalePrice']

# Split the data

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

# Scale the data

scaler = StandardScaler()

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

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

# Initialize models

models = {

"Linear Regression": LinearRegression(),

"Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=42),

"Gradient Boosting": GradientBoostingRegressor(n\_estimators=100, learning\_rate=0.1, random\_state=42)

}

# Train and evaluate models

for name, model in models.items():

model.fit(X\_train\_scaled, y\_train)

preds = model.predict(X\_test\_scaled)

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

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

print(f"{name} -> RMSE: {rmse:.2f}, R2: {r2:.2f}")

# Plot actual vs predicted

best\_model = models["Gradient Boosting"]

preds = best\_model.predict(X\_test\_scaled)

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

sns.scatterplot(x=y\_test, y=preds)

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual vs Predicted House Prices")

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--')

plt.show()