**SOURCE CODE**

**FORECASTING HOUSE PRICES ACCURATELY USING SMART REGRESSION TECHNIQUES IN DATA SCIENCE**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

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

from xgboost import XGBRegressor

def load\_data():

from google.colab import files

uploaded = files.upload()

filename = list(uploaded.keys())[0]

data = pd.read\_csv(filename)

print(f"Data loaded: {filename}, shape = {data.shape}")

return data

# Preprocess the dataset

def preprocess\_data(df):

df = df.drop(columns=['Id'], errors='ignore')

df = df.select\_dtypes(include=[np.number])

df = df.dropna()

return df

def visualize\_data(df, target='SalePrice'):

plt.figure(figsize=(8, 4))

sns.histplot(df[target], kde=True)

plt.title("Distribution of SalePrice")

plt.xlabel("SalePrice")

plt.ylabel("Frequency")

plt.show()

plt.figure(figsize=(12, 10))

corr = df.corr()

sns.heatmap(corr, cmap='coolwarm', annot=False, fmt=".2f")

plt.title("Correlation Heatmap")

plt.show()

top\_corr =

corr[target].drop(target).abs().sort\_values(ascending=False).head(5)

print("Top correlated features with SalePrice:")

print(top\_corr)

def prepare\_data(df, target='SalePrice'):

X = df.drop(columns=[target])

y = df[target]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

scaler = StandardScaler()

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

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

return X\_train\_scaled, X\_test\_scaled, y\_train, y\_test, X.columns

def train\_models(X\_train, y\_train):

lr = LinearRegression()

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

xgb = XGBRegressor(n\_estimators=100, learning\_rate=0.1,

random\_state=42)

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

return lr, xgb

def evaluate\_model(model, X\_test, y\_test, model\_name):

preds = model.predict(X\_test)

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

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

print(f"{model\_name} Performance:")

print(f" RMSE: {rmse:.2f}")

print(f" R² Score: {r2:.2f}")

print("

-

" \* 40)

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

sns.scatterplot(x=y\_test, y=preds, alpha=0.6)

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

color='red', linestyle='

--

')

plt.xlabel("Actual SalePrice")

plt.ylabel("Predicted SalePrice")

plt.title(f"{model\_name}: Actual vs Predicted")

plt.grid(True)

plt.show()

if \_name\_ == "\_main\_":

df = load\_data()

df\_clean = preprocess\_data(df)

visualize\_data(df\_clean)

X\_train, X\_test, y\_train, y\_test, feature\_names =

prepare\_data(df\_clean)

lr\_model, xgb\_model = train\_models(X\_train, y\_train)

evaluate\_model(lr\_model, X\_test, y\_test, "Linear Regression")

evaluate\_model(xgb\_model, X\_test, y\_test, "XGBoost Regressor")