Source code

```
# Forecasting House Prices Using Smart Regression Techniques

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, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
data = pd.read_csv("train.csv")  # Replace with your dataset path

# Data preprocessing
# Keep only numerical features and drop columns with missing values
data = data.select_dtypes(include=[np.number]) data =
data.dropna(axis=1)

# Split features and target
X = data.drop("SalePrice", axis=1)
y = data["SalePrice"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define regression models
models = {
    "LinearRegression": LinearRegression(),
    "Ridge": Ridge(alpha=1.0),
    "Lasso": Lasso(alpha=0.1),
    "RandomForest": RandomForestRegressor(n_estimators=100, random_state=42),
    "XGBoost": XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
}

# Train models and evaluate performance
results = {} for name, model in
models.items():     model.fit(X_train,
y_train)     y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)     results[name]
= {"MSE": mse, "R2": r2}     print(f"{name}: MSE =
{mse:.2f}, R2 = {r2:.2f}")

# Visualize results
results_df = pd.DataFrame(results).T
plt.figure(figsize=(10, 6))
results_df.plot(kind="bar", title="Model Performance Comparison", figsize=(10, 6))
plt.ylabel("Score")
plt.xticks(rotation=45)
plt.grid(True)
```

```
plt.tight_layout()
plt.show()
```