FORECASTING HOUSE PRICE

SOURCE CODE:

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

import numpy as np

 from sklearn.model_selection import train_test_split

 from sklearn.ensemble import
RandomForestRegressor, GradientBoostingRegressor

 from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,
r2_score

 from sklearn.preprocessing import StandardScaler

 from sklearn.pipeline import Pipeline

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

import joblib
import io

 from google.colab import files

 uploaded = files.upload()

df = pd.read_csv(io.BytesIO(uploaded['House Pric
India.csv']))

print(df.info()) print(df.describe())

df['Date'] = df['Date'].apply(lambda x:
datetime.fromordinal(datetime(1900, 1, 1).toordinal()
+ int(x) - 2))

current_year = datetime.now().year df['House_Age'] =
current_year - df['Built Year'] df['Renovated'] =
np.where(df['Renovation Year'] == 0, 0, 1)

df['Years_Since_Renovation'] =
np.where(df['Renovated'], current_year -
df['Renovation Year'], 0) df['Total_Area'] = df['Area of
the house(excluding basement)'] + df['Area of the
basement'] df['Bath_Bed_Ratio'] = df['number of
bathrooms'] / df['number of bedrooms']

features = [ 'number of bedrooms', 'number of
bathrooms', 'living area', 'lot area', 'number of floors',
'waterfront present', 'number of views', 'condition of
the house', 'grade of the house', 'Total_Area',
'House_Age', 'Renovated', 'Years_Since_Renovation',
'Number of schools nearby', 'Distance from the
airport', 'Bath_Bed_Ratio' ] target = 'Price'
```

```
df = df.dropna()
```

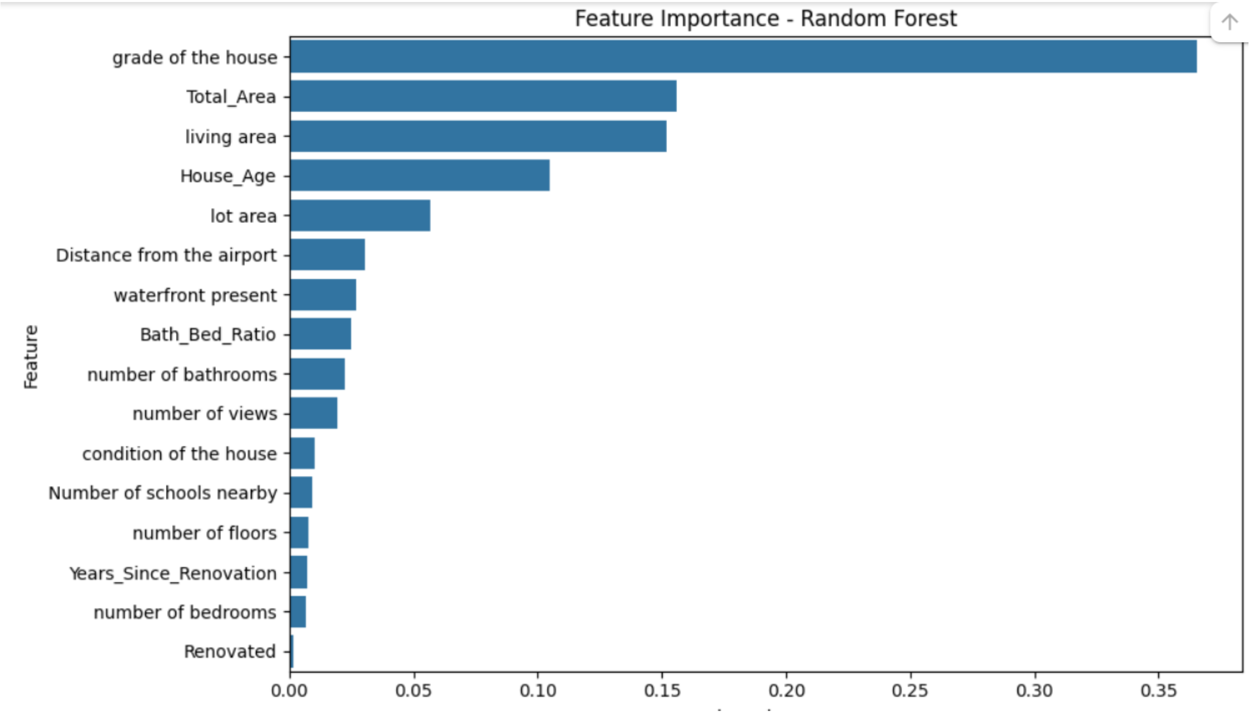
```
X = df[features] y = df[target] X_train, X_test, y_train,
y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
models = { 'Random Forest': Pipeline([ ('scaler',
StandardScaler()), ('model',
RandomForestRegressor(n_estimators=100,
random_state=42)) ]), 'Gradient Boosting':
Pipeline([ ('scaler', StandardScaler()), ('model',
GradientBoostingRegressor(n_estimators=100,
random_state=42)) ]), 'Linear Regression':
Pipeline([ ('scaler', StandardScaler()), ('model',
LinearRegression()) ]) }
```

```
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)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

results[name] = {'RMSE': rmse, 'R2 Score': r2}
print(f"{name} Performance:")
print(f"RMSE: {rmse:,.2f}")
print(f"R2 Score: {r2:.4f}\n")

best_model = models['Random Forest']

importances =
best_model.named_steps['model'].feature_importance
s_ feature_importance = pd.DataFrame({'Feature':
features, 'Importance':
importances}).sort_values('Importance',
ascending=False)

plt.figure(figsize=(10, 6)) sns.barplot(x='Importance',
y='Feature', data=feature_importance)
plt.title('Feature Importance - Random Forest')
plt.tight_layout() plt.show()

def predict_house_price(model, input_data): input_df
= pd.DataFrame([input_data]) return
model.predict(input_df)[0]

sample_input = { 'number of bedrooms': 4, 'number
of bathrooms': 2.5, 'living area': 2500, 'lot area':
10000, 'number of floors': 2, 'waterfront present': 0,
'number of views': 2, 'condition of the house': 4,
'grade of the house': 8, 'Total_Area': 2500,
'House_Age': 15, 'Renovated': 0,
'Years_Since_Renovation': 0, 'Number of schools
nearby': 3, 'Distance from the airport': 50,
'Bath_Bed_Ratio': 0.625 } predicted_price =
predict_house_price(best_model, sample_input)
print(f"\nPredicted House Price: ${predicted_price:,.2f}")

joblib.dump(best_model, 'house_price_predictor.pkl')
```

OUTPUT:


Feature Importance - Random Forest
Feature
grade of the house
Total_Area
living area
House_Age
lot area
Distance from the airport
waterfront present
Bath_Bed_Ratio
number of bathrooms
number of views
condition of the house
Number of schools nearby
number of floors
Years_Since_Renovation
number of bedrooms
Renovated