```
import pandas as pd
from google. colab import files
Data=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving global_housing_market_extended.csv to global_housing_market_extended.csv
```

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

```
df = pd.read_csv("global_housing_market_extended.csv")

df.dropna(inplace=True)
df = df[df['House Price Index'] > 0]  # filter invalid records

features = df.drop(columns=["Country", "Year", "House Price Index"])
target = df["House Price Index"]

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
models = {
    "Linear Regression": LinearRegression(),
    "Ridge": Ridge(alpha=1.0),
    "Lasso": Lasso(alpha=0.1),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42),
    "XGBoost": XGBRegressor(objective='reg:squarederror', n_estimators=100)
}

results = {}

for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    preds = model.predict(X_test_scaled)
```

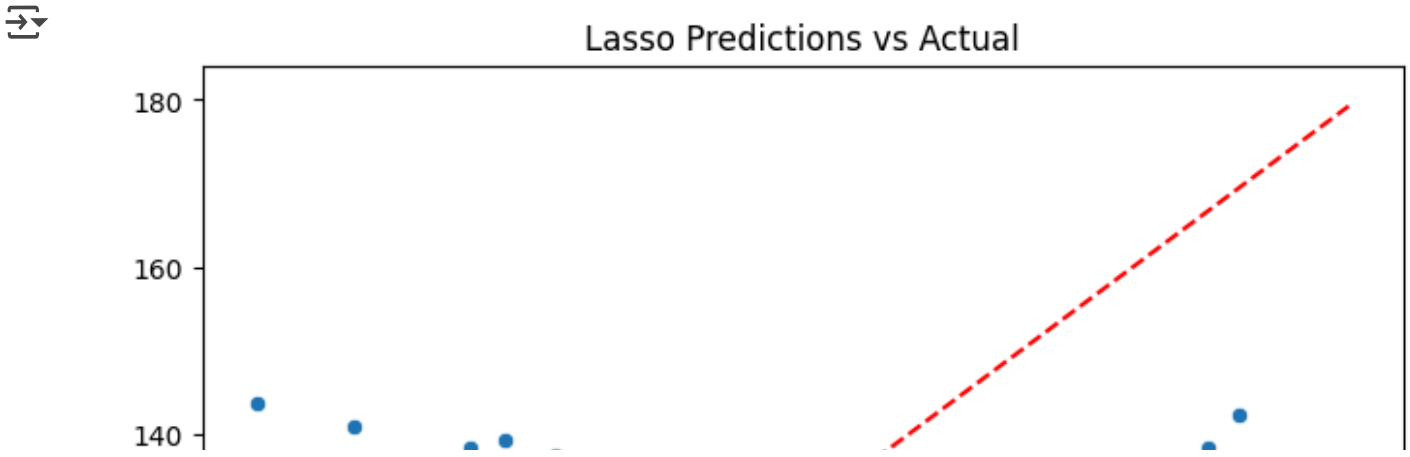
```
    mse = mean_squared_error(y_test, preds)
    r2 = r2_score(y_test, preds)
    results[name] = {"MSE": mse, "R2": r2}

results_df = pd.DataFrame(results).T.sort_values("R2", ascending=False)
print(results_df)
```

```
                            MSE        R2
Lasso                928.340653 -0.007526
Ridge                930.308636 -0.009662
Linear Regression    930.747005 -0.010138
Random Forest       1004.764189 -0.090469
XGBoost             1038.871449 -0.127485
```

```
best_model_name = results_df.index[0]
best_model = models[best_model_name]
predictions = best_model.predict(X_test_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=predictions)
plt.xlabel("Actual House Price Index")
plt.ylabel("Predicted")
plt.title(f"{best_model_name} Predictions vs Actual")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--')
plt.show()
```




```
print(df.shape)
print(df.columns)
print(df.head())
print(df.isnull().sum())
df = df.dropna(subset=['House Price Index'])

df.fillna(df.mean(numeric_only=True), inplace=True)
print(df.isnull().sum().sum())
```

```
(200, 11)
Index(['Country', 'Year', 'House Price Index', 'Rent Index',
       'Affordability Ratio', 'Mortgage Rate (%)', 'Inflation Rate (%)',
       'GDP Growth (%)', 'Population Growth (%)', 'Urbanization Rate (%)',
```

```
       'Construction Index'],
      dtype='object')
  Country  Year  House Price Index  Rent Index  Affordability Ratio  \
0     USA  2015         117.454012  116.550001             9.587945
1     USA  2016         150.807258   51.440915            11.729189
2     USA  2017         123.194502   70.386040             8.506676
3     USA  2018         131.423444   91.469020             3.418054
4     USA  2019         110.461377   56.837048             9.158097

   Mortgage Rate (%)  Inflation Rate (%)  GDP Growth (%)  \
0           4.493292            1.514121       -0.752044
1           5.662213            1.880204       -0.545400
2           2.197469            2.398940        0.930895
3           4.537724            1.608407       -1.479587
4           3.700762            1.293249        1.961415

   Population Growth (%)  Urbanization Rate (%)  Construction Index
0              -0.796707              85.985284          118.089201
1              -0.358084              69.127267          111.980515
2               0.596245              83.555279           85.973903
3               2.321099              88.968961          134.671788
4              -0.879640              87.279612           90.702399
Country                   0
Year                      0
House Price Index         0
Rent Index                0
Affordability Ratio       0
Mortgage Rate (%)         0
Inflation Rate (%)        0
GDP Growth (%)            0
Population Growth (%)     0
Urbanization Rate (%)     0
Construction Index        0
dtype: int64
0
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)
    mae = mean_absolute_error(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, predictions)

    return {
        'MAE': mae,
        'MSE': mse,
        'RMSE': rmse,
        'R2 Score': r2
    }



results = {}
```

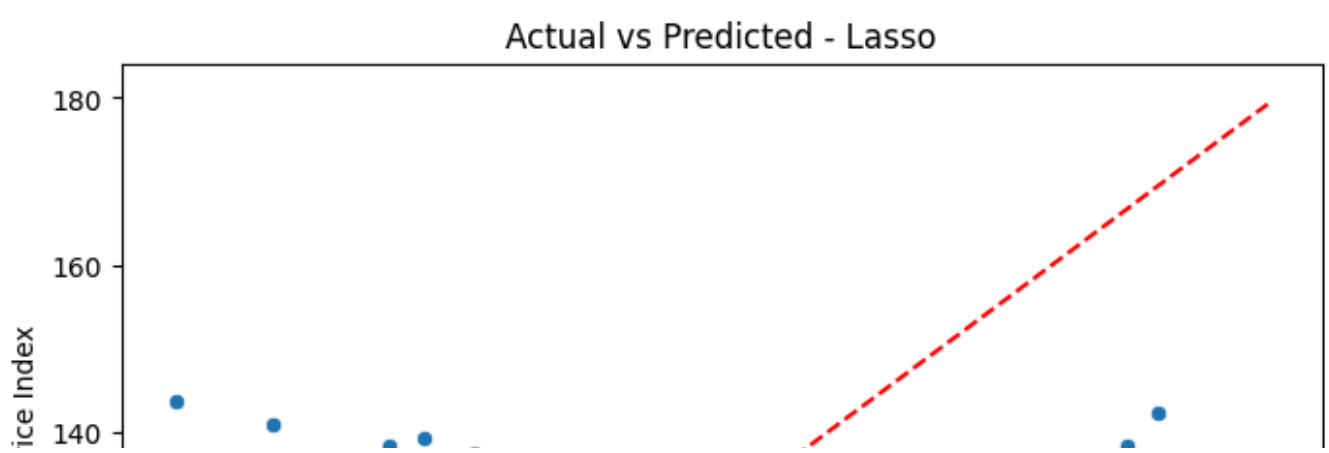
```
for name, model in models.items():
    results[name] = evaluate_model(model, X_test_scaled, y_test)

import pandas as pd
results_df = pd.DataFrame(results).T.sort_values(by='R2 Score', ascending=False)
print(results_df)
```

```
                            MAE           MSE       RMSE  R2 Score
Lasso                 26.956926    928.340653  30.468683 -0.007526
Ridge                 26.922620    930.308636  30.500961 -0.009662
Linear Regression     26.918329    930.747005  30.508147 -0.010138
Random Forest         27.253969   1004.764189  31.698016 -0.090469
XGBoost               26.387008   1038.871449  32.231529 -0.127485
```

```
best_model_name = results_df.index[0]
best_model = models[best_model_name]
y_pred = best_model.predict(X_test_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel('Actual House Price Index')
plt.ylabel('Predicted House Price Index')
plt.title(f'Actual vs Predicted - {best_model_name}')
plt.show()
```



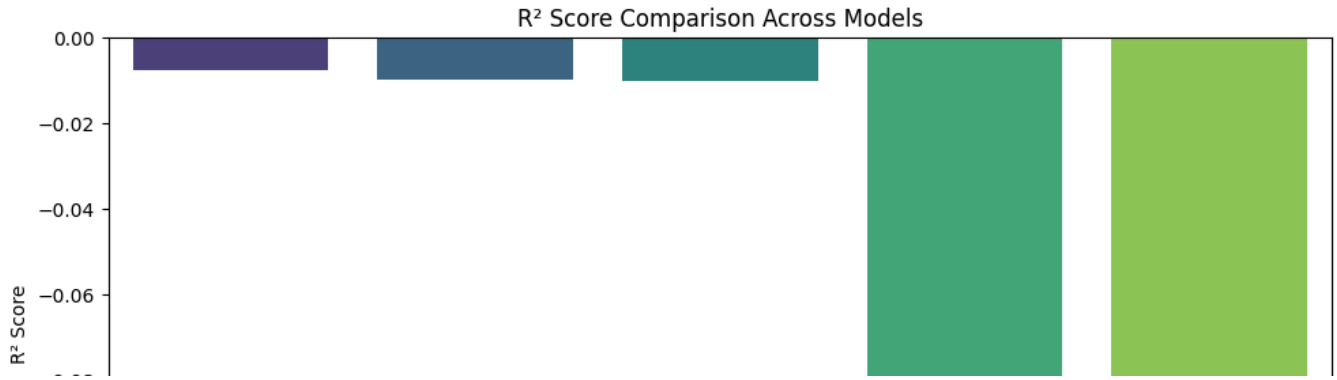

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x=results_df.index, y=results_df['R2 Score'], palette='viridis')
plt.xticks(rotation=45)
plt.title('R² Score Comparison Across Models')
plt.ylabel('R² Score')
plt.tight_layout()
plt.show()
```

```
<ipython-input-36-dcecf8e1b742>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v

  sns.barplot(x=results_df.index, y=results_df['R2 Score'], palette='viridis')
```




```
import joblib
joblib.dump(best_model, 'house_price_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

```
['scaler.pkl']
```

```
pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.30.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.1 (from gradio)
  Downloading gradio_client-1.10.1-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dis
Collecting ruff>=0.9.3 (from gradio)
```

```
  Downloading ruff-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.w
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dis
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.1
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-pa
```

```
import gradio as gr
import joblib
import numpy as np
import pandas as pd

model = joblib.load('house_price_model.pkl')
scaler = joblib.load('scaler.pkl')


features = [
    'Rent Index', 'Affordability Ratio', 'Mortgage Rate (%)',
    'Inflation Rate (%)', 'GDP Growth (%)', 'Population Growth (%)',
    'Urbanization Rate (%)', 'Construction Index', 'Country_Code'
]


def predict_price(rent_index, affordability, mortgage_rate, inflation, gdp, pop_growth,
                  urban_rate, construction_index, country_code):

    input_data = pd.DataFrame([[
        rent_index, affordability, mortgage_rate, inflation, gdp, pop_growth,
        urban_rate, construction_index, country_code
    ]], columns=features)

    scaled_input = scaler.transform(input_data)
    prediction = model.predict(scaled_input)
    return round(prediction[0], 2)
```

```
iface = gr.Interface(
    fn=predict_price,
    inputs=[
        gr.Number(label="Rent Index"),
        gr.Number(label="Affordability Ratio"),
        gr.Number(label="Mortgage Rate (%)"),
        gr.Number(label="Inflation Rate (%)"),
        gr.Number(label="GDP Growth (%)"),
        gr.Number(label="Population Growth (%)"),
        gr.Number(label="Urbanization Rate (%)"),
        gr.Number(label="Construction Index"),
        gr.Number(label="Country Code")
    ],
    outputs=gr.Number(label="Predicted House Price Index"),
    title="Smart House Price Predictor",
    description="Enter economic indicators to predict the House Price Index using smart reg
)

iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gra

Colab notebook detected. To show errors in colab notebook, set debug=True in lau
* Running on public URL: https://d7e5bb377efe7bd56e.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
```

Smart House Price Predictor

Enter economic indicators to predict the House Price Index using smart regression models.

Rent Index

0

Affordability Ratio