In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv("/Users/myyntiimac/Desktop/P4-Demographic-Data (1).csv")
df.head()
```

Out[3]:

	Country Name	Country Code	Birth rate	Internet users	Income Group
0	Aruba	ABW	10.244	78.9	High income
1	Afghanistan	AFG	35.253	5.9	Low income
2	Angola	AGO	45.985	19.1	Upper middle income
3	Albania	ALB	12.877	57.2	Upper middle income
4	United Arab Emirates	ARE	11.044	88.0	High income

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country Name    195 non-null    object 
 1   Country Code    195 non-null    object 
 2   Birth rate      195 non-null    float64
 3   Internet users  195 non-null    float64
 4   Income Group    195 non-null    object 
dtypes: float64(2), object(3)
memory usage: 7.7+ KB
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Country Name', 'Country Code', 'Birth rate', 'Internet users',
       'Income Group'],
      dtype='object')
```

In [6]:
```
df1=df[['Country Name',  'Birth rate', 'Internet users','Income Group']]
df1.head()
```

Out[6]:

	Country Name	Birth rate	Internet users	Income Group
0	Aruba	10.244	78.9	High income
1	Afghanistan	35.253	5.9	Low income
2	Angola	45.985	19.1	Upper middle income
3	Albania	12.877	57.2	Upper middle income
4	United Arab Emirates	11.044	88.0	High income

In [7]:
```
df1.columns = ['CountryName', 'Birthrate', 'Internetusers', 'IncomeGroup']
```

In [8]:
```
df1.head()
```

Out[8]:

	CountryName	Birthrate	Internetusers	IncomeGroup
0	Aruba	10.244	78.9	High income
1	Afghanistan	35.253	5.9	Low income
2	Angola	45.985	19.1	Upper middle income
3	Albania	12.877	57.2	Upper middle income
4	United Arab Emirates	11.044	88.0	High income

In [9]:
```
df1.isnull().any()
```

Out[9]:
```
CountryName      False
Birthrate        False
Internetusers    False
IncomeGroup      False
dtype: bool
```

In [10]:
```
df1.isnull().sum().sum()
```

Out[10]:
```
0
```

In [11]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   CountryName    195 non-null    object 
 1   Birthrate      195 non-null    float64
 2   Internetusers  195 non-null    float64
 3   IncomeGroup    195 non-null    object 
dtypes: float64(2), object(2)
memory usage: 6.2+ KB
```

In [12]:
```
df1['CountryName'] = df1['CountryName'].astype('category')
```

In [13]:
```
df1['IncomeGroup'] = df1['IncomeGroup'].astype('category')
```

In [14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype   
---  ------         --------------  -----   
 0   CountryName    195 non-null    category
 1   Birthrate      195 non-null    float64 
 2   Internetusers  195 non-null    float64 
 3   IncomeGroup    195 non-null    category
dtypes: category(2), float64(2)
memory usage: 9.5 KB
```

In [15]:
```
df1.head()
```

Out[15]:

	CountryName	Birthrate	Internetusers	IncomeGroup
0	Aruba	10.244	78.9	High income
1	Afghanistan	35.253	5.9	Low income
2	Angola	45.985	19.1	Upper middle income
3	Albania	12.877	57.2	Upper middle income
4	United Arab Emirates	11.044	88.0	High income

In [16]:
```
col=['CountryName','Internetusers', 'IncomeGroup']
X = df1[col]
```

In [17]:
```
X.head()
```

Out[17]:

	CountryName	Internetusers	IncomeGroup
0	Aruba	78.9	High income
1	Afghanistan	5.9	Low income
2	Angola	19.1	Upper middle income
3	Albania	57.2	Upper middle income
4	United Arab Emirates	88.0	High income

In [18]:
```
y=df1["Birthrate"]
```

In [19]:
```
y.head()
```

Out[19]:
```
0    10.244
1    35.253
2    45.985
3    12.877
4    11.044
Name: Birthrate, dtype: float64
```

In [20]:
```
# Convert categorical columns to numerical columns using cat.codes
X['CountryName'] = X['CountryName'].astype('category').cat.codes
X['IncomeGroup'] = X['IncomeGroup'].astype('category').cat.codes
```

In [21]:
```
X.head()
```

Out[21]:

	CountryName	Internetusers	IncomeGroup
0	7	78.9	0
1	0	5.9	1
2	3	19.1	3
3	1	57.2	3
4	182	88.0	0

In [22]:
```
# train test split

from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(X,y,test_size=0.20,random_s
```

In [23]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
train_x = sc.fit_transform(train_x)
test_x = sc.transform(test_x)
```

In [26]:
```
#importing ML models from scikit-learn
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,Gradien
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [27]:
```
regressors = {
    'Linear Regression' : LinearRegression(),
    'Lasso' : Lasso(),
    'Ridge' : Ridge(),
    'ElasticNet' : ElasticNet(),
    'Decision Tree' : DecisionTreeRegressor(),
    'Random Forest' : RandomForestRegressor(),
    'AdaBoostRegressor' : AdaBoostRegressor(),
    'GradientBoostingRegressor' : GradientBoostingRegressor(),
    'Support Vector Machines' : SVR(gamma=1),
    'K-nearest Neighbors' : KNeighborsRegressor(n_neighbors=1),
}
```

In [28]:
```
results=pd.DataFrame(columns=['R2-score','MSE','MAE'])
for method,func in regressors.items():
    model = func.fit(train_x,train_y)
    pred = model.predict(test_x)
    results.loc[method]= [np.round(r2_score(test_y,pred),3),
                          np.round(mean_squared_error(test_y,pred),3),
                          np.round(mean_absolute_error(test_y,pred),3)
                         ]
```

In [29]:
```
results
```

Out[29]:

	R2-score	MSE	MAE
Linear Regression	0.775	20.111	3.516
Lasso	0.759	21.574	3.431
Ridge	0.777	19.996	3.500
ElasticNet	0.686	28.118	4.025
Decision Tree	0.747	22.669	3.829
Random Forest	0.793	18.541	3.335
AdaBoostRegressor	0.771	20.482	3.908
GradientBoostingRegressor	0.723	24.765	3.828
Support Vector Machines	0.691	27.668	4.171
K-nearest Neighbors	0.495	45.214	4.257

In [30]:
```
results.sort_values('R2-score',ascending=False)
```

Out[30]:

	R2-score	MSE	MAE
Random Forest	0.793	18.541	3.335
Ridge	0.777	19.996	3.500
Linear Regression	0.775	20.111	3.516
AdaBoostRegressor	0.771	20.482	3.908
Lasso	0.759	21.574	3.431
Decision Tree	0.747	22.669	3.829
GradientBoostingRegressor	0.723	24.765	3.828
Support Vector Machines	0.691	27.668	4.171
ElasticNet	0.686	28.118	4.025
K-nearest Neighbors	0.495	45.214	4.257

Insight:Random Forest has the highest R2-score of 0.793, indicating that it performs the best among the evaluated models in terms of capturing the variance in the target variable. It shows a relatively good fit to the data.

Ridge Regression comes second with an R2-score of 0.775. It performs slightly worse than the Random Forest but still shows a reasonably good fit to the data. And followed by Linear regression , adaboost, lasso, dicision tree, And gradient boost