Birth Rate Prediction

Alessandro Dare(40208154), Ahmad Elmamlouk(40171892), Suin Kang(40129337), Weilun Zhang (40190549)

https://github.com/suinkangme/Birth_Rate_Prediction

Table of contents

01

Data Collection & Integration

Data Collection

Data sources:

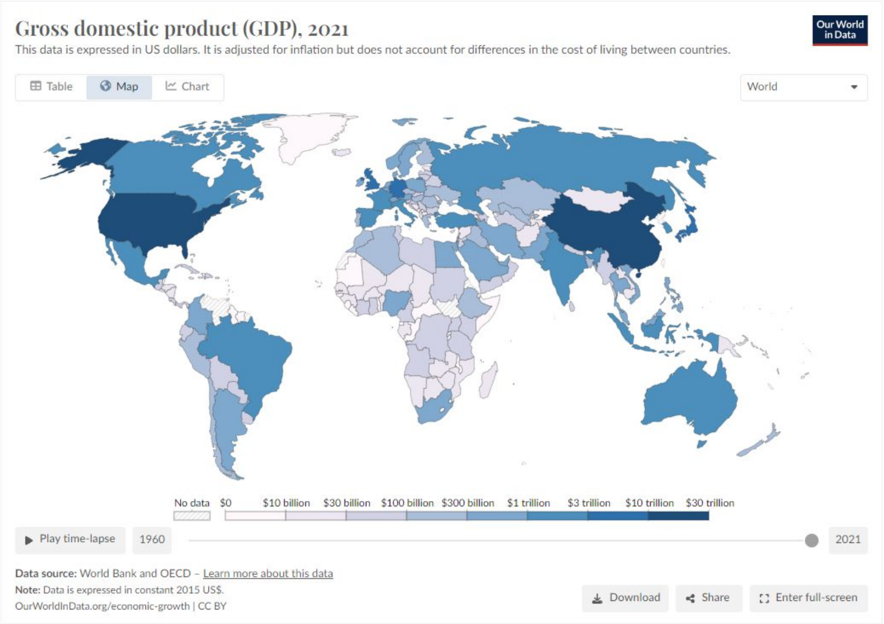
- https://ourworldindata.org/
- https://www.thearda.com/world-religion/national-profiles?u=234c

Files:

- Over 300 files: 18 world data, 300 religious
- Csv excel format

Storage:

- google docs, github, google colab



ETL - Extract

Chosen data:

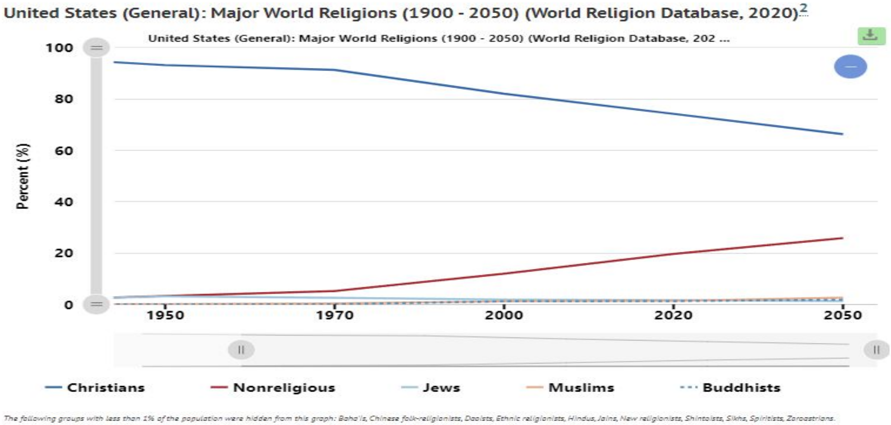
- Relevance
- Accessibility
- Difference to other data
- Ability to integrate

Format chosen:

- Primary key by country name, id and year
- Additional attribute that affects the birth rate
- Data collected from 1990-2020

Features chosen:

- Population, birth rare, fertility rate, urban population, gdp, gdp per capita, consumer price index, religious population, mortality rate, infant mortality rate, female labor force, education, life expectancy



Data Integration

Integration:

- Convert original datasets into base schema
- Rename and Delete Columns
- Merge all schemas into complete schema

Base Schema	
Primary Key	Entity:string Code:string Region:string Year:int
Non Pk	Fourth column:int

```
[ ] national_birth_rate= pd.read_csv("/content/crude-birth-rate.csv")
    national_birth_rate.rename(columns = {'Birth rate - Sex: all - Age: all - Variant: estimates':'Birth rate(pre 1000)'}, inplace = True)
    complete_dateset(national_birth_rate,'Birth rate(pre 1000)','complete_birth_rate')
```

```
complete_info = pd.DataFrame(columns=['Entity','Code','region', 'Year',forth_column_name])
global_info = assign_regions(global_info)
all_countries = global_info['Code'].unique()
print('all countries', len(all_countries))
for country in all_countries:
    country_info = global_info.loc[global_info['Code']==country]
    if len(country_info)>0:
        years_range = range(1990, 2021)
        country_info = country_info[country_info['Year'].isin(years_range)].sort_values('Year')
        country_info = check_for_outlier_values(country_info,forth_column_name)
        missing_info = fill_missing(country_info,country_info['Entity'].iloc[0] , country_info['region'].iloc[0] ,country,forth_column_name)
        complete_info = pd.concat([complete_info, missing_info]).sort_values('Code')
print('all countries', len(complete_info['Code'].unique()))
complete_info = filling_country_by_region(complete_info, forth_column_name)
file_path = "/content/output/"+save_file_name+".csv"
complete_info = complete_info.sort_values('Code')
print('all countries', len(complete_info['Code'].unique()))
complete_info.to_csv(file_path, index=False)
```

Schema Integration

GDP	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	GDP:int

Women In Workforce %	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	Women In Workforce:itn

Completed Schema	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	GDP Years of education Women In Workforce Years of education

CPI	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	CPI:int

Years of education:	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	Years of education:int

Conversion (Our world data):

Original Schema:

Fertility-vs-child-mortality	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	Child mortality rate – Sex: all – Age: 0–4 – Variant: estimates, :double Fertility rate – Sex: all – Age: all – Variant: estimates Population (historical estimates) Continent

New Schema:

Base	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	Child mortality rate %:double

Conversion (Religious data):

Original Schema

Albania	
Primary key	Year (int)
Non Pk	R496 R603 R609 R505 R530 R585

Country Schema

Albania	
Primary key	Entity (string) Code (string) Region (string) Year (int)
Non Pk	Atheists rate(double)

Combined Schema

Complete religious	
Primary key	Entity:string Code:string Region:string Year:int
Non Pk	Atheists rate:double

Process happens normally afterwards

02

Data Cleaning & Transformation

Data Cleaning

For missing values:

- **Year Gaps:**
  - Take difference between years fill up the gap by increase/decrease
  - Ex: Gap between 1991 value:1, and 1995 value:5
  - Fill gaps:1992 value:2, 1993 value:3, 1994 value:4
- **For zero values:**
  - Replace with decade mean
  - **Mean1:** 1990-2000 **Mean2:** 2000-2010, **Mean3:** 2010-2020
- **For missing countries:**
  - Replace with average of region
  - For Every Year

Outliers Detection

Set to zero:

- Negative values
- Nan values

Robust Random Cut Forest:

- Direct checking
- Values removed

Year limit:

- 1990-2020

```
for index, row in country.iterrows():
    if np.isnan(row[what_is_missing]) or row[what_is_missing] <0:
        country.loc[index, what_is_missing]=0
        row[what_is_missing] = 0
data_array = country[what_is_missing].values[:, None]
num_trees = 50
forest = []
```

```
for i in range(data_array.shape[0]):
    point = data_array[i]
    for tree in forest:
        tree.insert_point(point, index=i)
avg_codisp = np.zeros(data_array.shape[0])
index = np.zeros(data_array.shape[0])
for tree in forest:
    for i in range(data_array.shape[0]):
        codisp = np.array(tree.codisp(i))
        avg_codisp += codisp
    index += 1
avg_codisp /= index
threshold = np.percentile(avg_codisp, 95)
outliers = country[avg_codisp > threshold]
if len(outliers) >0:
    print("Outliers:")
    print(outliers)
    print(len(country))
    country = country.drop(outliers, axis = 0).reset_index(drop=True)
    print(len(country))
```

```
years_range = range(1990, 2021)
country_info = country_info[country_info['Year'].isin(years_range)].sort_values('Year')
```

Data Transformation

1. Data Normalization
2. Data Scaling
3. Data Encoding

Data Normalization

- **Elimination of Redundancy**
  - **Country** feature and **Code** feature
  - Certain datasets provide both name and code
    - country name (ex: **Canada**)
    - country code (ex: **CAN**)
  - This information is redundant and was consequently removed from the dataset
- **Elimination of Dependency**
  - **Birth rate** feature and **Fertility rate** feature
  - Birth rate is dependent on fertility rate therefore the fertility rate feature was dropped from the dataset

Data Scaling

- **Min-max scaling**
  - Min max scaled features with absolute values to fit a [0–1] range
    - **EX:** GDP – years of education
- **Percentage scaling**
  - All features expressed as ratios or percentages were scaled to fit a [0–1] range
    - **EX:** Child mortality rate – Female labor force participation rate
- **Transforming absolute values to ratio values**
  - Scaled absolute values of urban/rural populations to population rates to fit a [0–1] range

Min-max scaling

```
def min_max_scale_columns(dataframe,list_of_column_names):
  min_max_scaler = MinMaxScaler()

  dataframe[list_of_column_names] = min_max_scaler.fit_transform(dataframe[list_of_column_names])
  return dataframe
```

Percentage scaling

```
def scale_percentage_columns(dataframe, list_of_column_names):
  for i in range(len(list_of_column_names)):
    dataframe[list_of_column_names[i]] = dataframe[list_of_column_names[i]]/100
  return dataframe
```

Absolute to ratio scaling

```
def scale_population_columns(dataframe, list_of_column_names):
  for i in range(len(list_of_column_names)):
    dataframe[list_of_column_names[i]] = dataframe[list_of_column_names[i]]/dataframe['Population']
  return dataframe
```

Data Encoding

- **One-hot encoding**
  - one categorical feature in our dataset, **country name.**
  - Represent each categorical value (**country name**) with an encoded feature of its own
  - Each encoded feature holds a binary vector indicating the country of the row

Country
Canada
United States
France

Canada	United States	France
1	0	0
0	1	0
0	0	1

One hot encoding categorical values

```
def generate_one_hot_encode_dataframe(dataframe, column_name):
  encoder = OneHotEncoder(sparse = False)

  extracted_text_column = dataframe[column_name]
  transformed_text_column = encoder.fit_transform(extracted_text_column)

  encoded_df = pd.DataFrame(transformed_text_column, columns=encoder.get_feature_names_out(text_column_name))
  dataframe.drop(column_name, axis=1, inplace=True)

  one_hot_encoded_df = pd.concat([dataframe,encoded_df], axis = 1)
  return one_hot_encoded_df
```

ETL - Transform

Clean

- Imputation of missing countries
- Imputation of missing years
- Imputation of missing values

Transform

- Address formatting discrepancies
- Transform categorical values to numerical values
- Scale numerical values to a [0–1] scale

03

Prediction with Machine Learning

ETL - Load

Load the transformed data into a model for training

```
# load the fully transformed data
dataset = pd.read_csv('final_transformed_dataset.csv')
print(dataset.columns)
```

```
dataset.head()
```

	Year	Atheists rate	Child mortality rate %	Consumer price index	female labor force participation rate	GDP	GDP per capita	Period life expectancy	Death rate	Population	...
0	0.000000	0.0	0.022342	0.002414	0.441184	0.000078	0.193799	0.814113	0.047246	0.000040	...
1	0.866667	0.0	0.016182	0.005016	0.513658	0.000151	0.233354	0.849185	0.031638	0.000067	...
2	0.233333	0.0	0.021660	0.003254	0.466299	0.000116	0.229769	0.819308	0.041741	0.000051	...
3	0.900000	0.0	0.015907	0.004964	0.513834	0.000159	0.244937	0.853136	0.031570	0.000068	...
4	0.333333	0.0	0.021312	0.003527	0.470444	0.000129	0.234609	0.820918	0.039420	0.000056	...

Machine Learning - Regression

Model 1 – Linear Regression

```
from sklearn.linear_model import LinearRegression
model_1 = LinearRegression()
model_1.fit(X_train, y_train)
```

```
# model 1 evaluation
y_pred_1 = model_1.predict(X_test)
mse_1 = mean_squared_error(y_test, y_pred_1)
```

Model 2 – Support Vector Machine (SVM)

```
from sklearn.svm import SVR
model_2 = SVR(kernel = 'rbf')
model_2.fit(X_train, y_train)
```

```
y_pred_2 = model_2.predict(X_test)
mse_2 = mean_squared_error(y_test, y_pred_2)
```

Machine Learning - Regression

Model 3 – Multi Layer Perceptron (MLP)

```
class Model(nn.Module):
    def __init__(self, input_size):
        super(Model, self).__init__()
        self.fc = nn.Linear(input_size, 1)

    def forward(self, x):
        return self.fc(x)
```

```
criterion = nn.MSELoss()
optimizer = optim.Adam(model_3.parameters(), lr=0.001)
```

```
# training
num_epochs = 10

for epoch in range(num_epochs):
    model_3.train()
    total_loss = 0.0

    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model_3(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
```

```
# testing
model_3.eval()
with torch.no_grad():
    y_pred = []
    for inputs, targets in test_loader:
        outputs = model_3(inputs)
        y_pred.append(outputs.numpy())
y_pred = np.concatenate(y_pred)

mse_3 = mean_squared_error(y_test, y_pred)
```

Model performance without transformation

- **Prediction without feature scaling**

```
MSE from Model 1(Linear Regression): 4.6253
MSE from Model 2(SVM Regression): 111.9755
MSE from Model 3(MLP): 975997493775162540032.0000
```

- **Prediction with feature scaling**

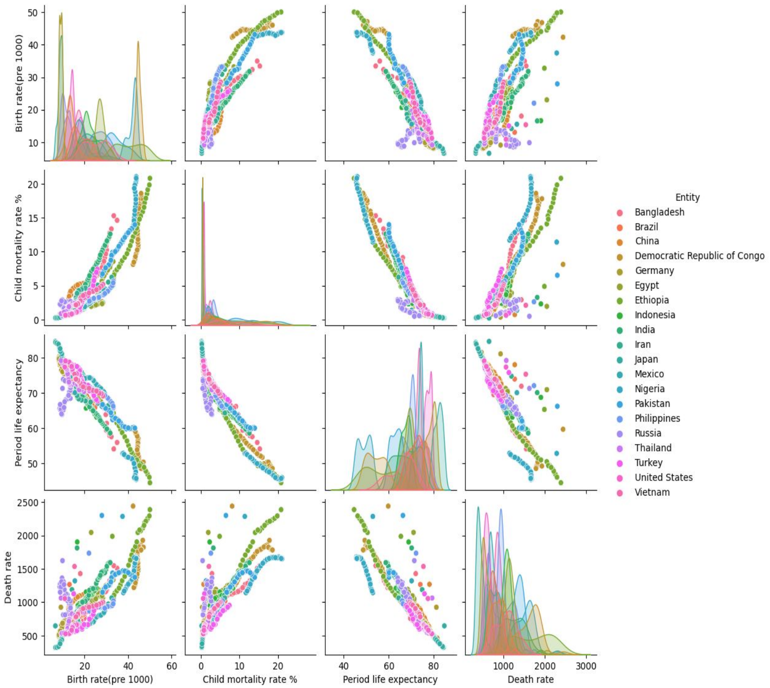
```
MSE from Model 1: 3.9375
MSE from Model 2: 119.7355
MSE from Model 3: 201.8581
```

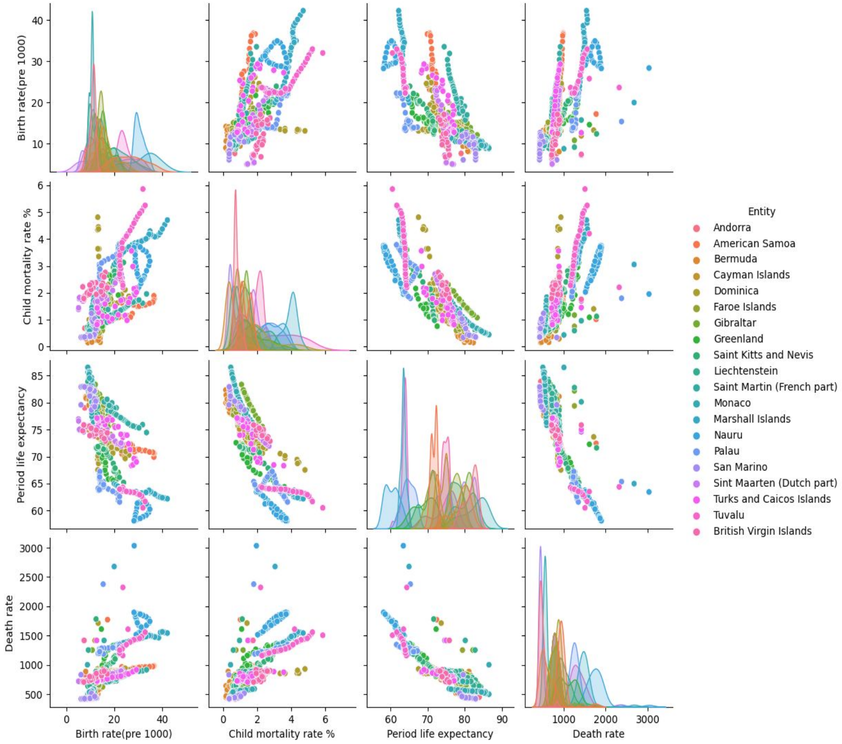
=> Best Model : Linear Regression

04

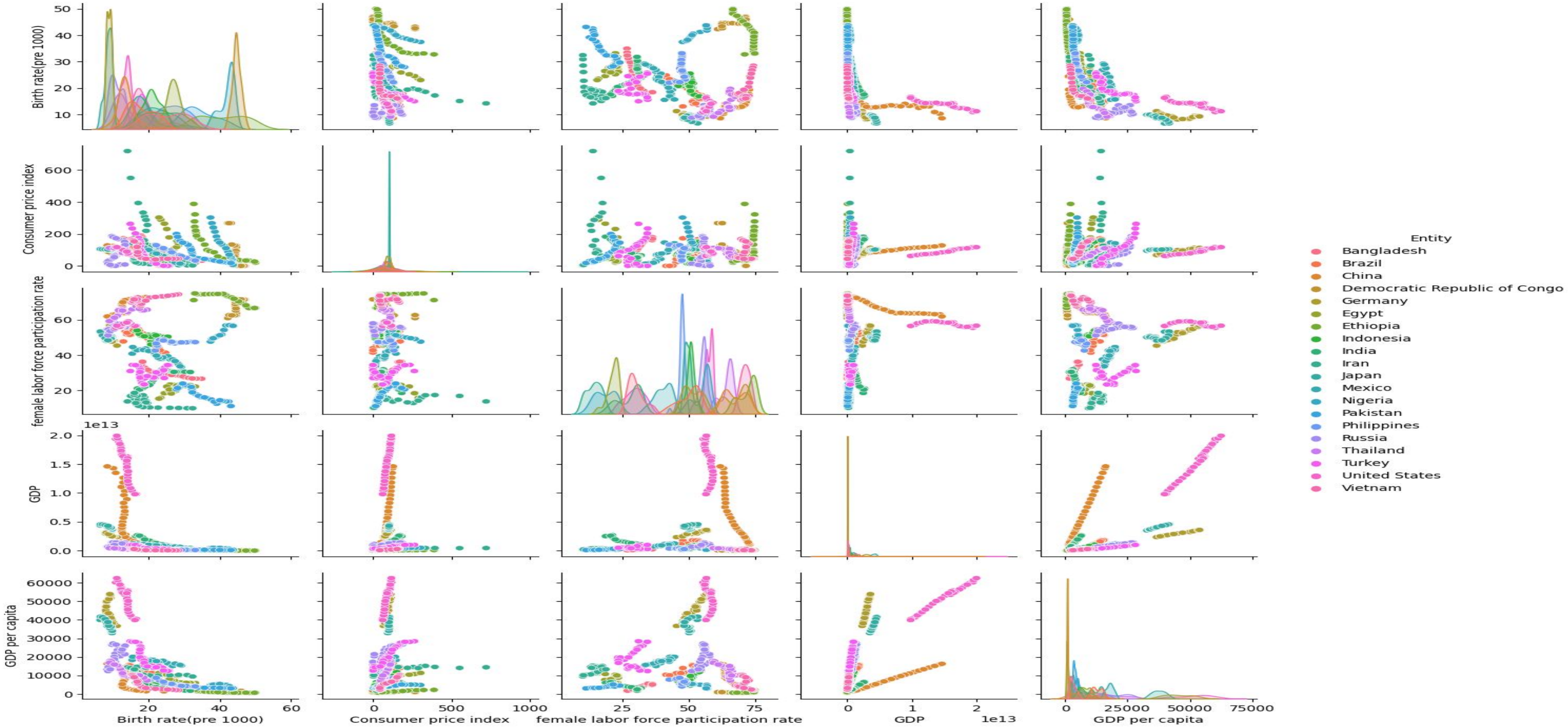
Data Visualization

Birth, Child Mortality, life expectancy, and Death rate (High and Low Population Countries)

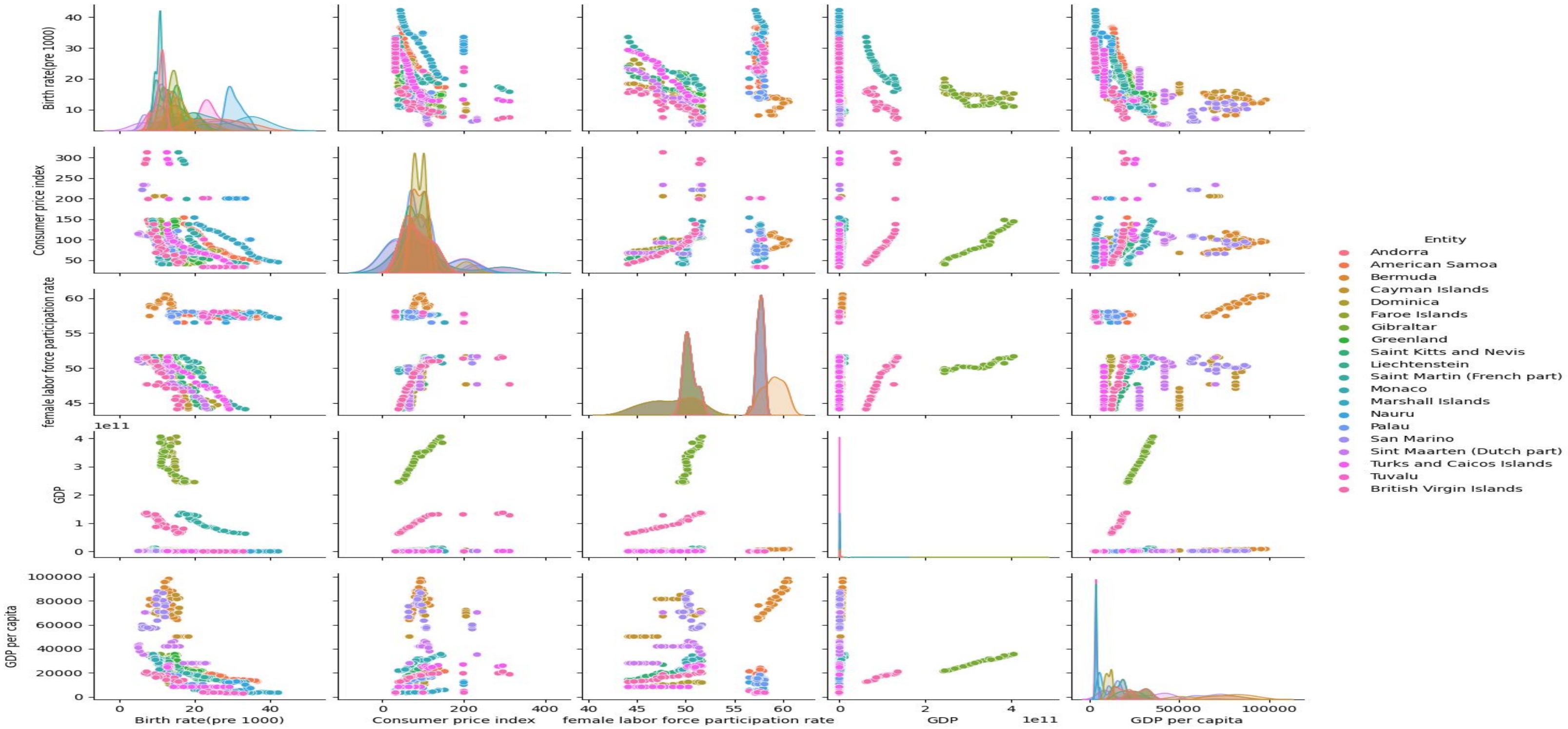




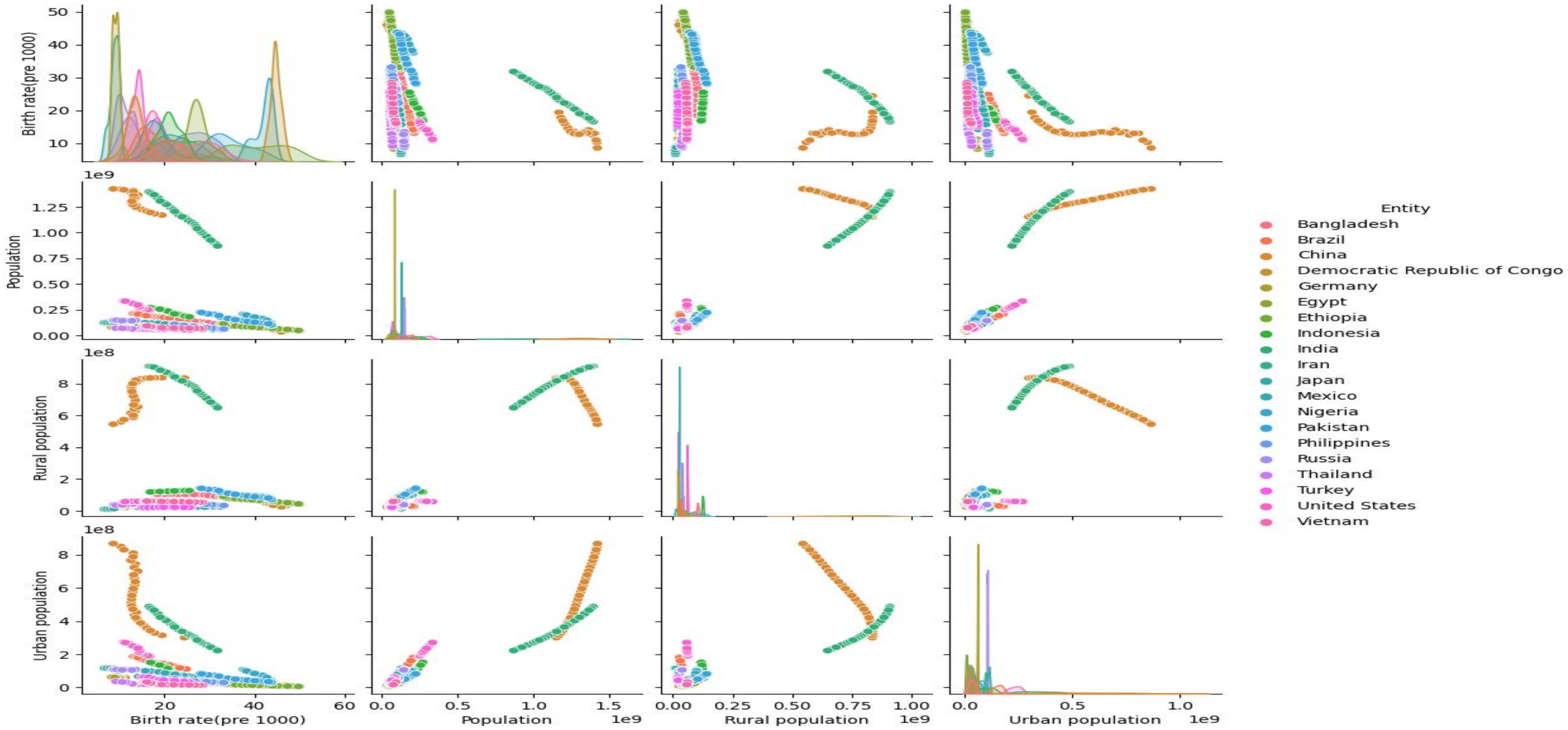
Birth, Consumer price index, female labor force participation rate, and GDP (High Population)



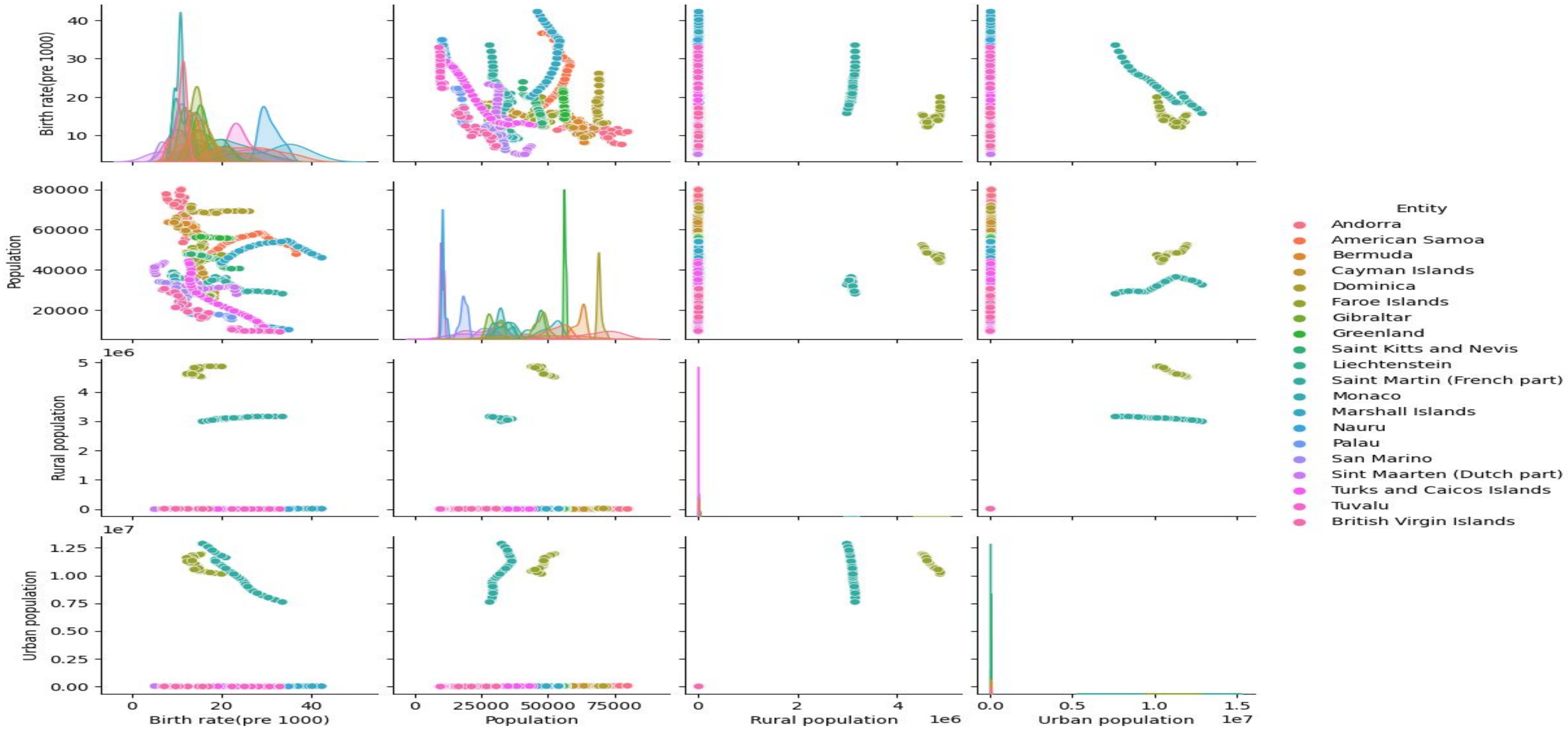
Birth, Consumer price index, female labor force participation rate, and GDP (Low Population)



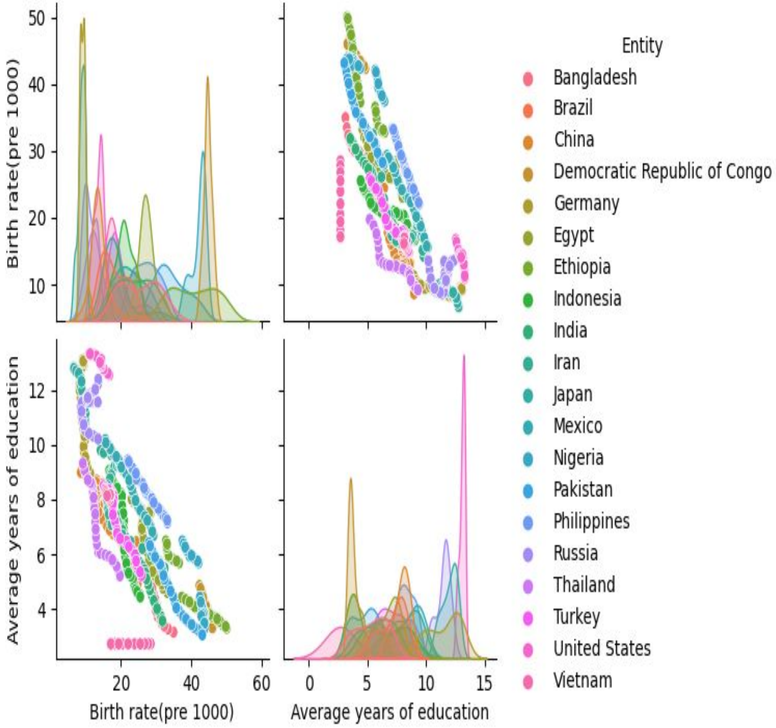
Birth, Population, Rural population rate, and Urban population (High Population)

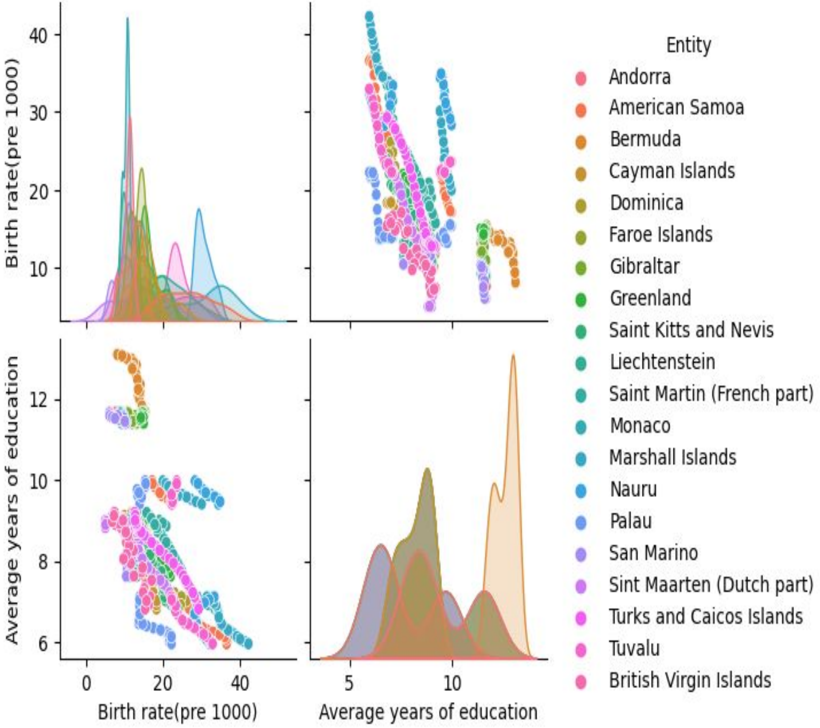


Birth, Population, Rural population rate, and Urban population (low Population)



Birth and Average years of education

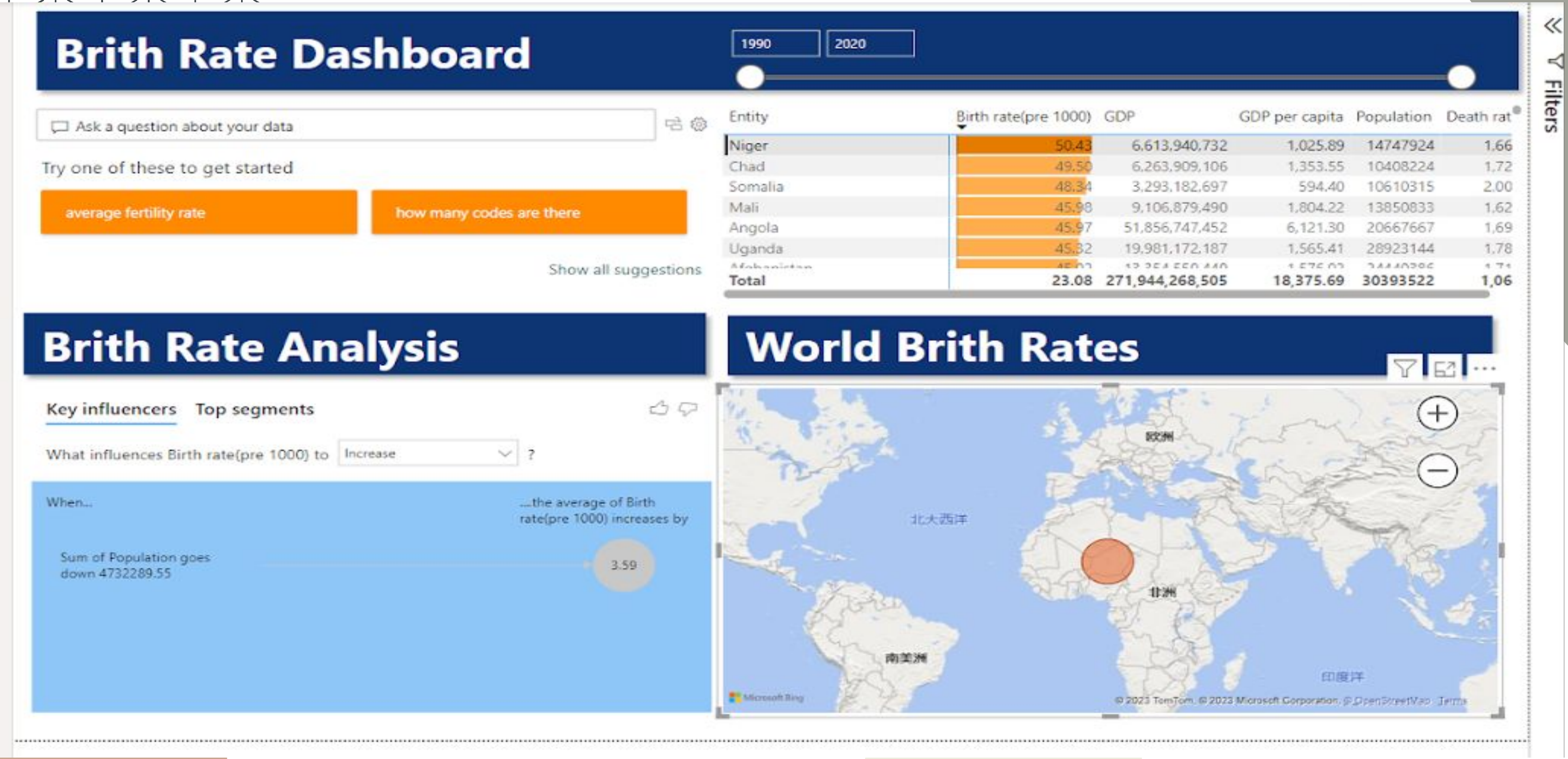




Dashboard


Brith Rate Dashboard
1990
2020
Ask a question about your data
Try one of these to get started
count codes
maximum birth rate(pre 1000)
Show all suggestions
Entity
Birth rate(pre 1000)
GDP
GDP per capita
Population
Zimbabwe 34.54 17,482,047,806 2,312.96 12508595
Zambia 42.89 13,235,058,684 2,636.56 12259995
Yemen 39.10 45,323,064,774 27,501.93 22034311
Vietnam 18.96 149,282,454,355 5,368.72 82877225
Venezuela 22.46 100,581,599,116 16,619.71 26107057
Vanuatu 33.30 593,343,008 2,873.97 222343
Total 23.08 271,944,268,505 18,375.69 30393522
Brith Rate Analysis
Key influencers
Top segments
What influences Birth rate(pre 1000) to
Increase
When...
...the average of Birth rate(pre 1000) increases by
Sum of GDP is 182997300 - 22859676000
7.89
Sum of GDP is 182997300 or less
7.57
World Brith Rates
北美洲
北大西洋
南美洲

Dashboard


Brith Rate Dashboard
1990
2020
Ask a question about your data
Try one of these to get started
average fertility rate
how many codes are there
Show all suggestions
Entity
Birth rate(pre 1000)
GDP
GDP per capita
Population
Death rat
Niger 50.43 6,613,940,732 1,025.89 14747924 1.66
Chad 49.50 6,263,909,106 1,353.55 10408224 1.72
Somalia 48.34 3,293,182,697 594.40 10610315 2.00
Mali 45.98 9,106,879,490 1,804.22 13850833 1.62
Angola 45.97 51,856,747,452 6,121.30 20667667 1.69
Uganda 45.32 19,981,172,187 1,565.41 28923144 1.78
Total 23.08 271,944,268,505 18,375.69 30393522 1.06
Brith Rate Analysis
Key influencers
Top segments
What influences Birth rate(pre 1000) to
Increase
?
When...
...the average of Birth rate(pre 1000) increases by
Sum of Population goes down 4732289.55
3.59
World Brith Rates
欧洲
北大西洋
非洲
南美洲
印度洋
Filters

Thank you