Korean Birthrate Problem

Team 8

19102085 Park Sehong
19102088 Park Jinsoo
21102044 Oh Seyeon

CONTENTS

01 Topic Introduction

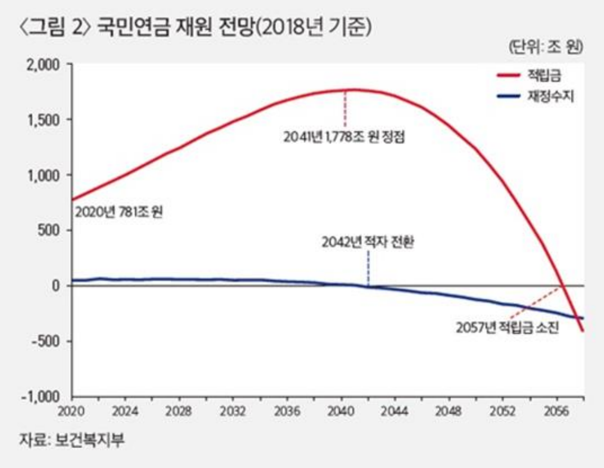
Background

Korea's fertility rate hits 0.78, a new low and still lowest in OECD

[SHUTTERSTOCK]

▪ **What are the risks of decreasing the birth rate in Korea?**

- Demographic Challenges
- Economic Impact
- National Defense Concerns



▪ There are so **many reasons for decreasing the birth rate in Korea!!**

- Economic Pressures, Demanding Work Culture, Educational Pressure, Limited Childcare & Parental Leave, Delay in Marriage, Housing Challenges

Purpose of Analysis

1) The purpose of creating a birth rate prediction model is to analyze and identify the factors that most significantly influence birth rates.

2) Proposing Solutions from a Socio-Economic Standpoint Based on the Identified Causes

02 Data

Data Acquisition

- Period: 1990 ~ 2021
- 15 OECD Member Countries

Country Code	Country Name
AUT	AUSTRIA
CAN	Canada
CHE	Switzerland
DEU	Germany
ESP	Spain

Country Code	Country Name
FIN	Finland
FRA	France
GRC	Greece
HUN	Hungary
ITA	Italy

Country Code	Country Name
JPN	Japan
KOR	Korea
LUX	Luxembourg
POL	Poland
PRT	Portugal

Data Acquisition

- 27 Features : related to socio-economic factors

1) Population and birth rate data:

- Fertility rate (OECD)
- Population (The World Bank)

2) Family and marital data:

- Mean age of women at childbirth (OECD)
- Average age at marriage (Our World in Data)
- Mean age at first marriage (The World Bank)
- Marriage Rate (Our World in Data)

3) Data related to women:

- Female Labor Participation Rate (The World Bank)

4) Housing-related data:

- Housing prices (OECD)
- Short-term interest rates (OECD)

Data Acquisition

- 27 Features : related to socio-economic factors

5) Work and employment data:

- Average annual hours worked (OECD)
- Employment rate (OECD)
- Unemployment rate(OECD)
- Part-time employment rate (OECD)
- Proportion of dual-income households (OECD)
- Labor Participation Rate (The World Bank)

6) Economic and financial data:

- Current health public expenditure (The World Bank)
- Public spending on family benefits (OECD)
- Public spending on labor markets (OECD)
- Gross national income – GNI (OECD)
- Gender Development Index - GDI (Our World in Data)
- Gross domestic product – GDP (OECD)
- Public expenditure for labor market (OECD)
- Public expenditure to compensate for unemployment (OECD)
- Public expenditure on education (% of GDP) (The World Bank)
- Public expenditure on education (The World Bank)
- Inflation rate (OECD)
- Poverty Gap (The World Bank)

EDA

- Data Information

```
In [53]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 28 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   ID                             480 non-null    object 
 1   Year                           480 non-null    int64  
 2   Country Name                   480 non-null    object 
 3   FemaleLaborParticipationRate   480 non-null    float64
 4   AvgHoursWorked                 453 non-null    float64
 5   BothWorking                    191 non-null    float64
 6   FirstBirthAge                  455 non-null    float64
 7   MarriageAge                    174 non-null    float64
 8   MarriageRate                   406 non-null    float64
 9   EmploymentRate                 465 non-null    float64
 10  UnemploymentRate               434 non-null    float64
 11  HousingPrice                   414 non-null    float64
 12  InterestRate                   432 non-null    float64
 13  PartTimeRate                   462 non-null    float64
 14  FamilyExpenditure              445 non-null    float64
 15  HealthExpenditure              318 non-null    float64
 16  LaborMarketExpenditure         415 non-null    float64
 17  UnemploymentExpenditure        446 non-null    float64
 18  GDI                            470 non-null    float64
 19  GDP                            479 non-null    float64
 20  GNI                            455 non-null    float64
 21  PovertyGap                     308 non-null    float64
 22  EduExpenditureOfGDP            415 non-null    float64
 23  EduExpenditureOfGov            370 non-null    float64
 24  TotalLaborParticipationRate    480 non-null    float64
 25  InflationRate                  480 non-null    float64
 26  Population                     480 non-null    float64
 27  FertilityRate                  480 non-null    float64
dtypes: float64(25), int64(1), object(2)
memory usage: 105.1+ KB
```

EDA

- Check the missing ratio

결측치 비율

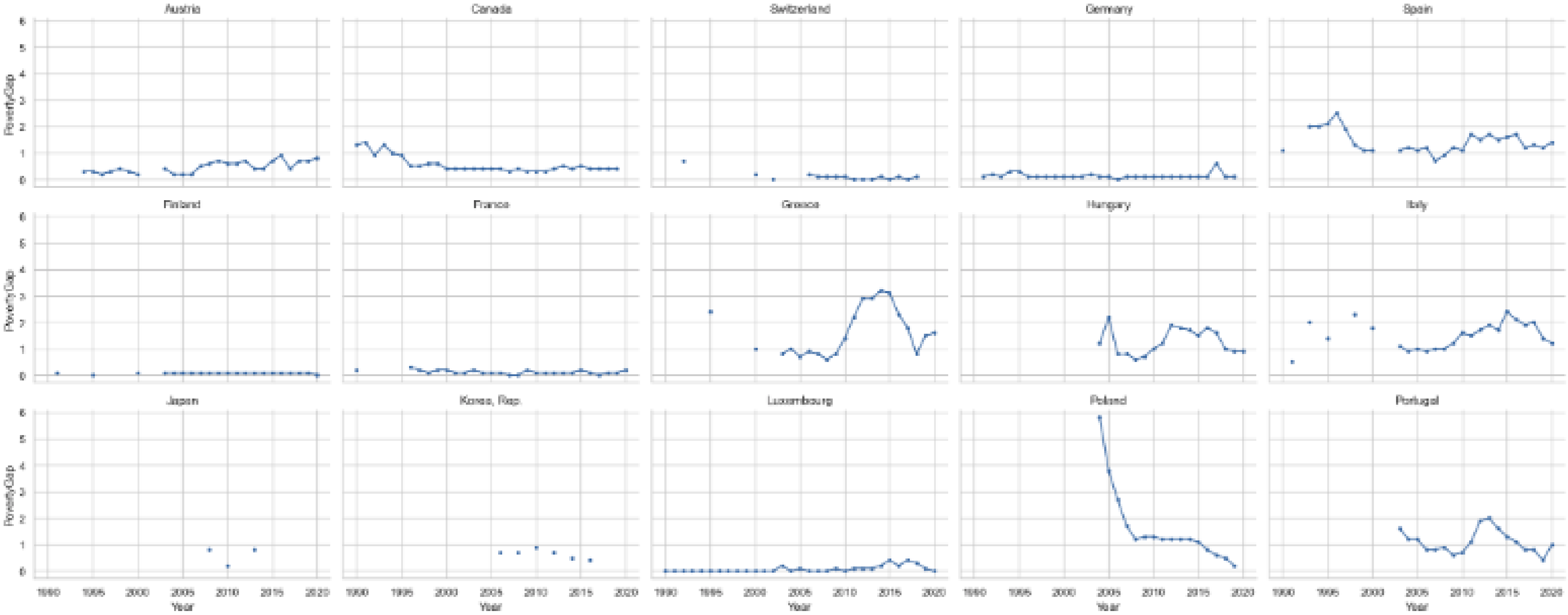
In [54]:
```
import pandas as pd
missing_ratio = data.isnull().mean() * 100
missing_ratio_df = pd.DataFrame(missing_ratio, columns=['missing_ratio'])
missing_ratio_df.sort_values(by='missing_ratio', ascending=False, inplace=True)
print(missing_ratio_df)
```

```
                              missing_ratio
MarriageAge                       63.750000
BothWorking                       60.208333
PovertyGap                        35.833333
HealthExpenditure                 33.750000
EduExpenditureOfGov               22.916667
MarriageRate                      15.416667
HousingPrice                      13.750000
LaborMarketExpenditure            13.541667
EduExpenditureOfGDP               13.541667
InterestRate                      10.000000
UnemploymentRate                   9.583333
FamilyExpenditure                  7.291667
UnemploymentExpenditure            7.083333
AvgHoursWorked                     5.625000
FirstBirthAge                      5.208333
GNI                                5.208333
PartTimeRate                       3.750000
EmploymentRate                     3.125000
GDI                                2.083333
GDP                                0.208333
TotalLaborParticipationRate        0.000000
InflationRate                      0.000000
Population                         0.000000
ID                                 0.000000
Year                               0.000000
FemaleLaborParticipationRate       0.000000
Country Name                       0.000000
FertilityRate                      0.000000
```

EDA

- Check the missing ratio – graph

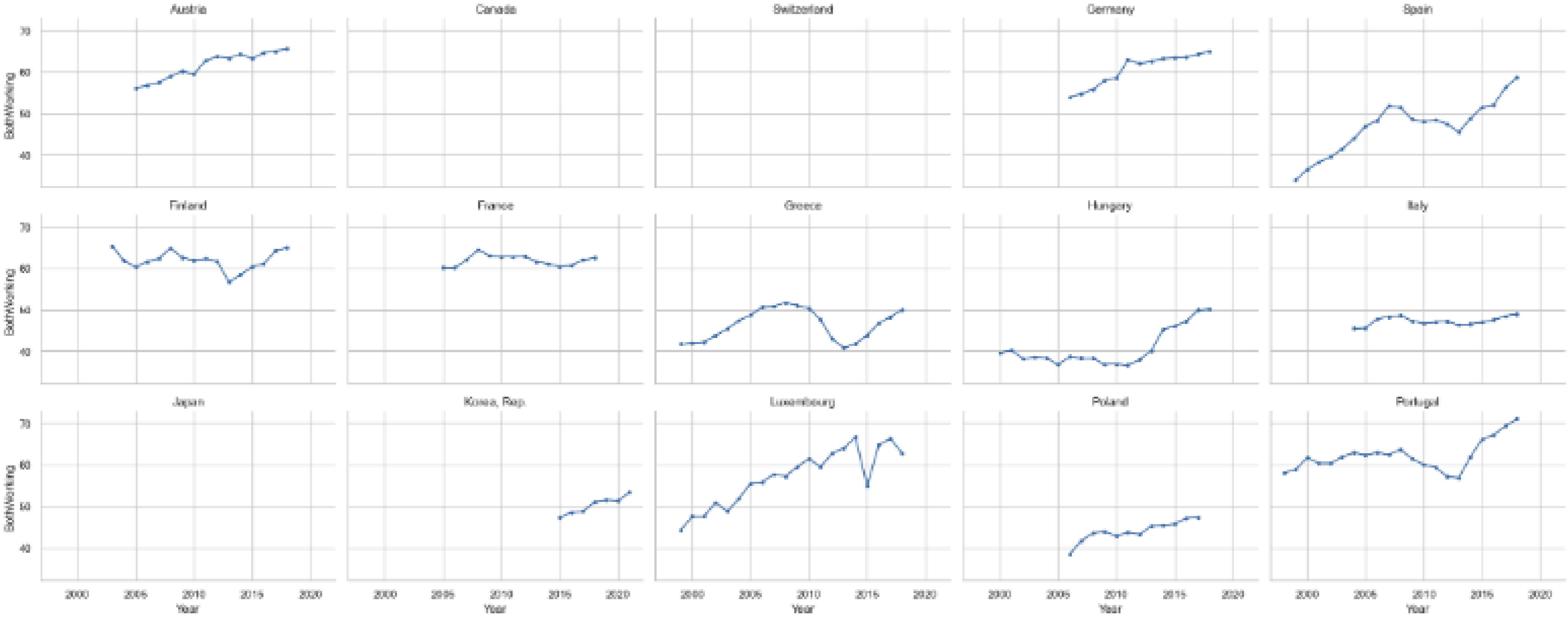
  – PovertyGap



EDA

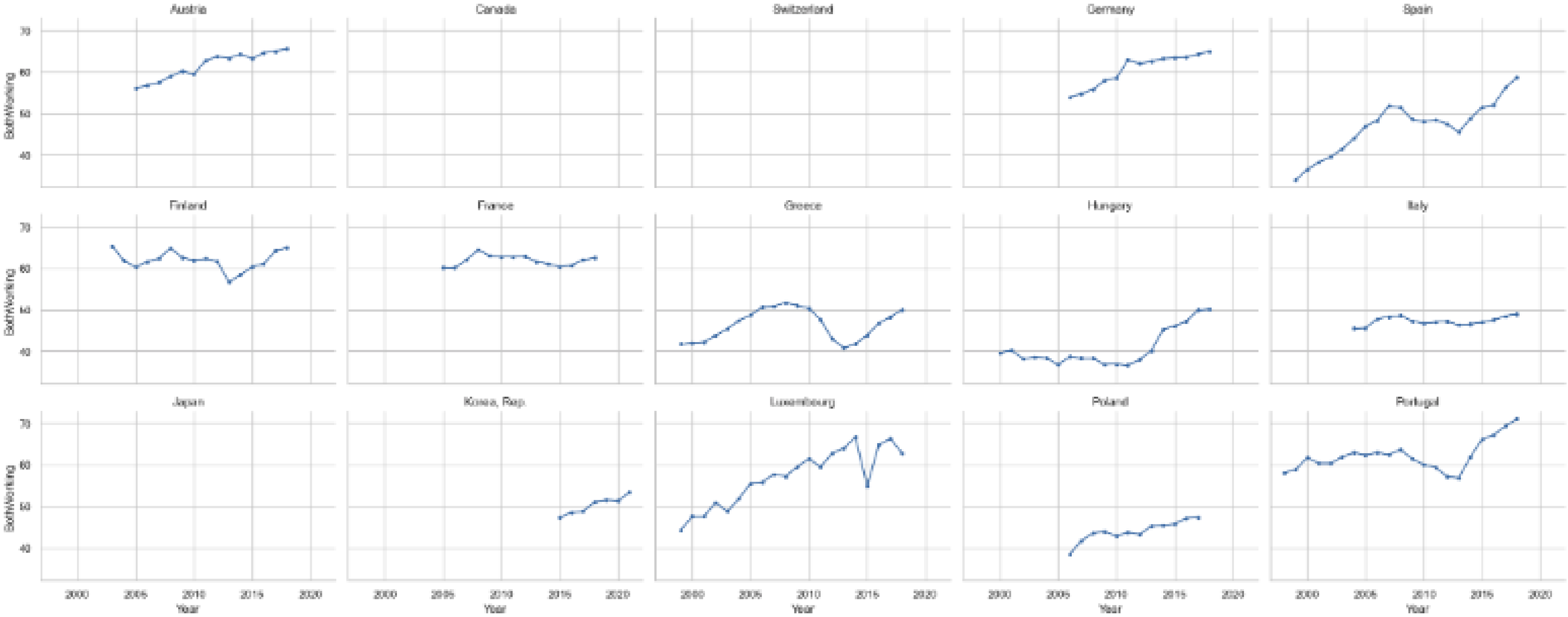
- Check the missing ratio - graph

  - BothWorking



EDA

There is a lot of missing data that needs to be handled further.



Preprocessing

- Handling Missing Values

1) List up the missing ratio in descending order

	missing_ratio
~~MarriageAge~~	~~63.750000~~
~~BothWorking~~	~~60.208333~~
~~PovertyGap~~	~~35.833333~~
~~HealthExpenditure~~	~~33.750000~~
~~EduExpenditureOfGov~~	~~22.916667~~
MarriageRate	15.416667
HousingPrice	13.750000
LaborMarketExpenditure	13.541667
EduExpenditureOfGDP	13.541667
InterestRate	10.000000
UnemploymentRate	9.583333
FamilyExpenditure	7.291667
UnemploymentExpenditure	7.083333
AvgHoursWorked	5.625000
FirstBirthAge	5.208333
GNI	5.208333
PartTimeRate	3.750000
EmploymentRate	3.125000
GDI	2.083333
GDP	0.208333
TotalLaborParticipationRate	0.000000
InflationRate	0.000000
Population	0.000000
ID	0.000000
Year	0.000000
FemaleLaborParticipationRate	0.000000
Country Name	0.000000
FertilityRate	0.000000

2) Drop features with a missing value over 20%

결측치 20% 이상 제거

```
columns_to_drop = data.columns[data.isnull().mean() > 0.2]
data = data.drop(columns_to_drop, axis=1)
```

Preprocessing

- Handling Missing Values

3) If missing values are concentrated in a country
, remove the feature

"LaborMarketExpenditure"
data in Greece

6.697394	0.651		0.57	0.387	0.919	13221.94
6.875668	0.784		0.63	0.464	0.921	13974.52
7.202623	0.766		0.6	0.419	0.927	14288.45
7.129302	0.784		0.55	0.37	0.934	14314.17
7.838299	1.005		0.58	0.388	0.932	14844.89
7.788225	1		0.75	0.392	0.945	15433.24
8.005857	0.978		0.73	0.397	0.95	16149.88
8.315235	0.926		0.67	0.414	0.953	17155.8
9.0686	0.965			0.405	0.962	18034.13
7.965456	0.911			0.387	0.968	18464.73
5.344826	0.818	545.6469		0.589	0.972	19519.98
4.704793	0.758	635.1075		0.581	0.97	20958.78
5.3786	0.726	706.644		0.594	0.965	22615.96
5.601154	0.757	937.7139		0.57	0.961	23897.76
5.89515	0.777	1045.115		0.625	0.96	25460.56
6.382767	[illegible]	1177.302		0.635	0.956	25577.45
7.431965	[illegible]	[illegible]		0.566	0.958	28547.22
[illegible]	[illegible]	[illegible]		0.543	0.961	29323.91
7.933897	[illegible]	[illegible]		0.658	0.962	30855.94
8.504816	[illegible]	1919.173		0.946	0.968	30359.45
8.90689	[illegible]	1766.964		0.937	0.965	27911.57
9.085164	1.021	1536.738		1.067	0.97	25671.41
9.786493	1.177	1239.16		0.852	0.969	24911.13
10.32657	1.329	1124.729		0.695	0.968	25986.64
11.15286	1.377	967.1238		0.52	0.968	26625.16
11.11607	1.285	854.2432		0.487	0.973	26760.28
10.95004	1.271	777.7966		0.533	0.97	27511.8
10.99145	1.655	787.5415		0.516	0.97	28604.83
10.42838	1.884	836.2425		0.527	0.969	29617.52
10.43254	1.768	765.0709		0.569	0.967	31155.95
9.748244		904.0972			0.967	28416.52
9.060123					0.969	31297.16

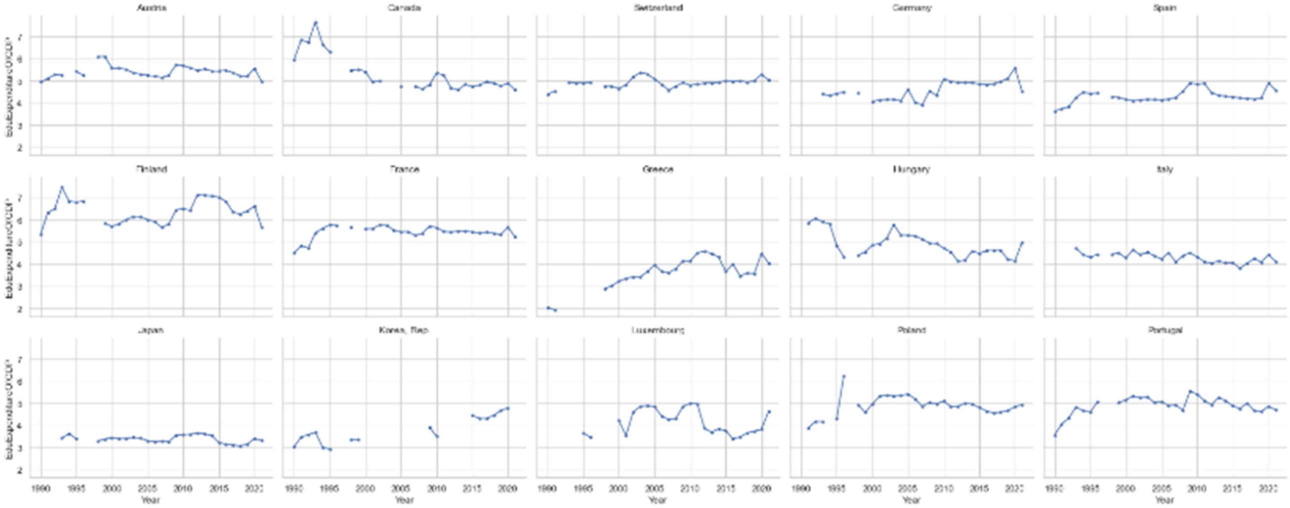
4) Missing value handling using <u>time-based Interpolation</u>

```python
def interpolate_with_direction(column):
    if column.isna().iloc[0] and column.isna().iloc[-1]:
        # 양 끝이 모두 NaN인 경우: 양방향 보간
        return column.interpolate(method='time', limit_direction='both')
    elif column.isna().iloc[0]:
        # 시작 부분이 NaN인 경우: 앞에서 뒤로 보간
        return column.interpolate(method='time', limit_direction='backward')
    elif column.isna().iloc[-1]:
        # 끝 부분이 NaN인 경우: 뒤에서 앞으로 보간
        return column.interpolate(method='time', limit_direction='forward')
    else:
        # 중간에 NaN이 있는 경우: 기본 보간
        return column.interpolate(method='time')
```

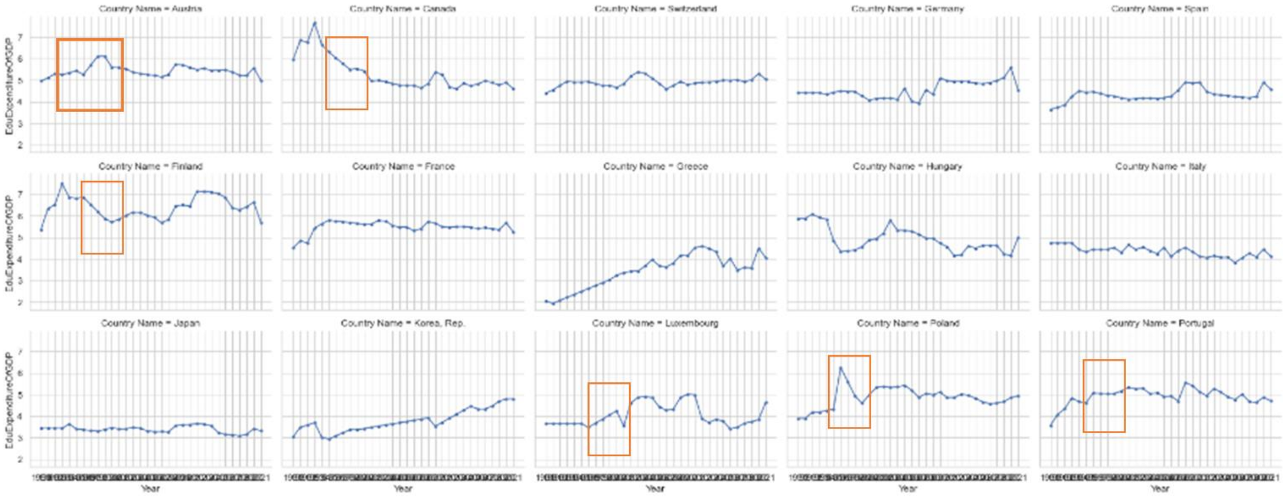
```python
# 각 데이터셋에 대해 결측치 보간
for dataset in country_data_list:
    dataset['Year'] = pd.to_datetime(dataset['Year'], format='%Y')
    dataset.set_index('Year', inplace=True)

    # 열별로 결측치 패턴에 따라 보간을 적용
    for column in dataset.columns:
        dataset[column] = interpolate_with_direction(dataset[column])
```

Preprocessing

[before]



[after]



- Missing values are filled

Feature Engineering

- ## Feature Extraction

1) Work-Leisure Balance Index

```
data['WorkLeisureBalanceIndex'] = data['AvgHoursWorked'] / data['TotalLaborParticipationRate']
```

- It provides a detailed look at the characteristics of the labor market. This ratio indicates the average working hours relative to the labor participation rate, offering insights into the intensity of work and leisure time available, which can impact birth rates.

2) LaborMarketStability

```
data['LaborMarketStability'] = data['EmploymentRate'] / data['UnemploymentRate']
```

- This feature helps in assessing the overall health and stability of the labor market, where a higher ratio indicates stability (high employment, low unemployment), and a lower ratio indicates instability (low employment, high unemployment).

Feature Engineering

- Feature Extraction

3) HousingAffordabilityIndex

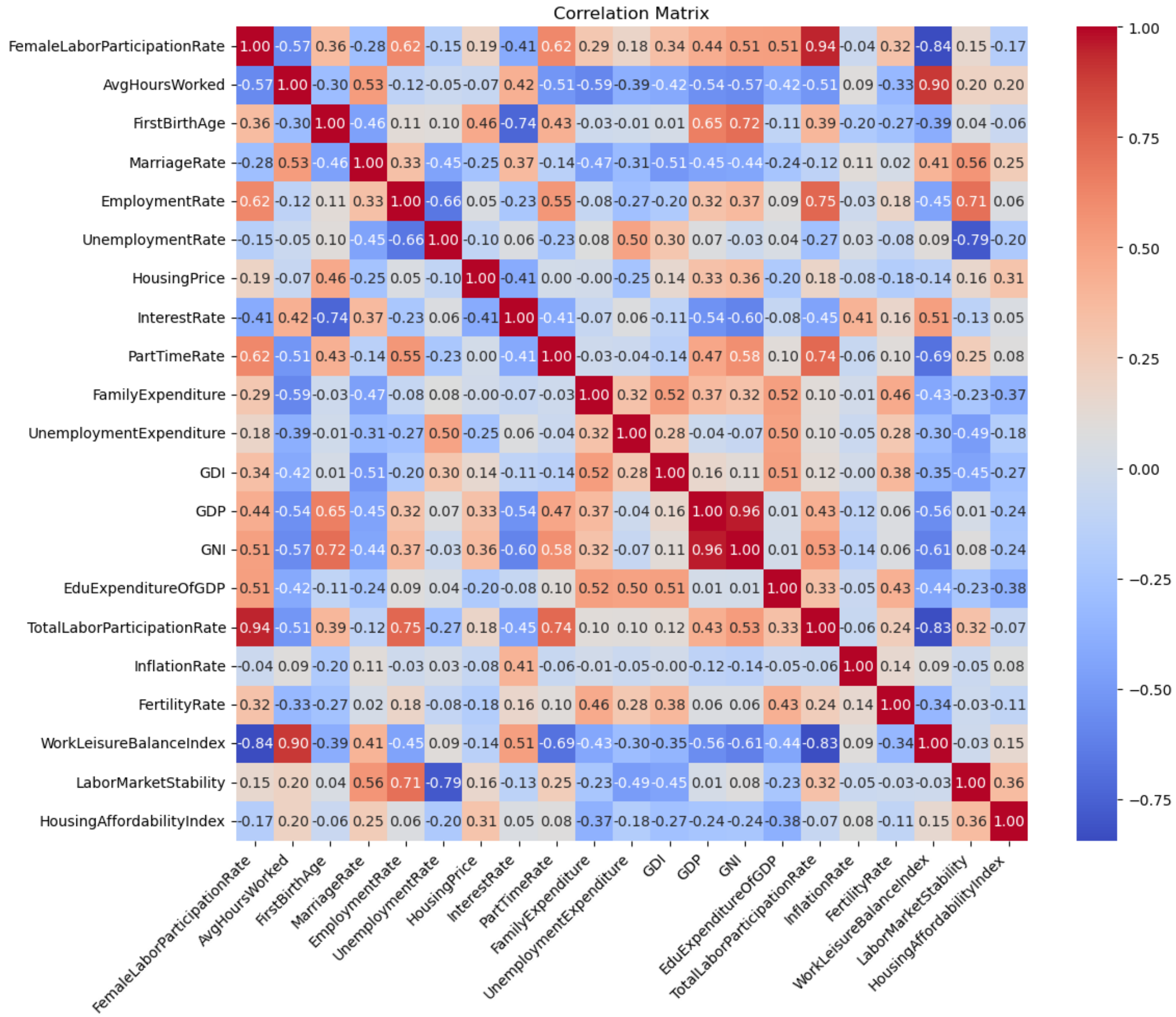
```
data['PerCapitaGDP'] = data['GDP'] / data['Population']
data['HousingAffordabilityIndex'] = data['HousingPrice'] / data['PerCapitaGDP']
```

- This index quantifies the economic burden of purchasing a home by comparing the cost of buying a house to the income level of individuals or households.

Feature Engineering

Correlation Analysis(over 0.7)

- Since there are many factors in which features influence each other, it seems that the correlation between them may have an impact on predicting birth rates. Therefore, a process of selecting features based on their correlation is necessary.



- WorkLeisureBalanceIndex
- TotalLaborParticipationRate
- GNI
- GDP
- LaborMarkeyStability
- UnemploymentRate
- EmploymentRate
- FirstBirthAge
- InterestRate

Feature Engineering

Correlation analysis measures the linear relationships between variables, identifying their direct associations.

→ However, this method cannot capture complex nonlinear relationships between variables. Additionally, analyzing correlations alone may not fully reflect the actual influence or importance of a variable on a model.

→ Additionally, in order to consider the relationships between various features as much as possible for the purpose of our project, we did not think it would be appropriate to exclude features only through correlation analysis.

→ Relying solely on correlation analysis to select features for exclusion is problematic.

To address this, we've utilized RFECV with nonlinear models for more sophisticated feature selection.
This approach allows us to eliminate features which have high correlation and non-essential to model.
It enables feature selection that considers both linear and nonlinear relationships.

Feature Engineering

- RFECV

- Used Non-linear Model

1. RandomForestRegressor
2. XGBoostRegressor
3. DecisionTreeRegressor

- Why did we perform RFECV with various models?

Because each model interprets the structure of the data in a different way and evaluates important characteristics differently, we performed RFECV with various models and interpreted them as comprehensive results.

```python
import pandas as pd
from sklearn.feature_selection import RFECV
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor

X = data.drop(['Year', 'FertilityRate'], axis=1)
y = data['FertilityRate']

models = {
    'RandomForestRegressor': RandomForestRegressor(random_state=0),
    'XGBRegressor': XGBRegressor(random_state=0),
    'DecisionTreeRegressor': DecisionTreeRegressor(random_state=0)
}

feature_rankings = pd.DataFrame()

for model_name, model in models.items():
    rfecv = RFECV(estimator=model, step=1, cv=5, scoring='neg_mean_squared_error')
    rfecv.fit(X, y)

    ranking_df = pd.DataFrame({
        'Feature': X.columns,
        f'Ranking_{model_name}': rfecv.ranking_
    })

    if feature_rankings.empty:
        feature_rankings = ranking_df
    else:
        feature_rankings = feature_rankings.merge(ranking_df, on='Feature')

feature_rankings = feature_rankings.sort_values(by='Ranking_RandomForestRegressor')

feature_rankings
```

Feature Engineering

- RFECV

- We considered only features with high correlation.

- Among the features showing high correlation, select to remove the feature with the lowest overall ranking in the three models.

  - GNI
  - WorkLeisureBalanceIndex

[Feature Importance Ranking]

	Feature	Ranking_RandomForestRegressor	Ranking_XGBRegressor	Ranking_DecisionTreeRegressor
0	FemaleLaborParticipationRate	1	1	1
17	WorkLeisureBalanceIndex	1	4	8
15	TotalLaborParticipationRate	1	1	4
14	EduExpenditureOfGDP	1	1	1
12	GDP	1	1	1
11	GDI	1	1	1
10	UnemploymentExpenditure	1	1	1
18	LaborMarketStability	1	3	6
9	FamilyExpenditure	1	1	1
7	InterestRate	1	7	2
6	HousingPrice	1	1	1
5	UnemploymentRate	1	1	1
4	EmploymentRate	1	1	1
2	FirstBirthAge	1	1	1
1	AvgHoursWorked	1	1	1
19	HousingAffordabilityIndex	1	1	5
3	MarriageRate	2	2	9
16	InflationRate	3	8	1
13	GNI	4	6	7
8	PartTimeRate	5	5	3

Feature Engineering

- Final data

18 Variables

Target Variable → (17 FertilityRate)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 20 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Year                         480 non-null    int64
 1   FemaleLaborParticipationRate 480 non-null    float64
 2   AvgHoursWorked               480 non-null    float64
 3   FirstBirthAge                480 non-null    float64
 4   MarriageRate                 480 non-null    float64
 5   EmploymentRate               480 non-null    float64
 6   UnemploymentRate             480 non-null    float64
 7   HousingPrice                 480 non-null    float64
 8   InterestRate                 480 non-null    float64
 9   PartTimeRate                 480 non-null    float64
 10  FamilyExpenditure            480 non-null    float64
 11  UnemploymentExpenditure      480 non-null    float64
 12  GDI                          480 non-null    float64
 13  GDP                          480 non-null    float64
 14  EduExpenditureOfGDP          480 non-null    float64
 15  TotalLaborParticipationRate  480 non-null    float64
 16  InflationRate                480 non-null    float64
 17  FertilityRate                480 non-null    float64
 18  LaborMarketStability         480 non-null    float64
 19  HousingAffordabilityIndex    480 non-null    float64
dtypes: float64(19), int64(1)
memory usage: 75.1 KB
```

03 Modeling

Train/Valid/Test dataset

- To avoid overfitting by training on a wide historical range and testing on the latest data to ensure the model can predict new trends accurately.

- Train : 1990-2015
- Valid : 2016-2018
- Test : 2019-2021

Hyperparameter Tuning

- Used Model

1. RandomForestRegressor
2. XGBoostRegressor
3. DecisionTreeRegressor

: Comparing RandomForestRegressor, XGBoostRegressor, and DecisionTreeRegressor regressors is sensible because they all handle complex, non-linear data well, like our dataset with many features influencing fertility rates.

Hyperparameter Tuning

1. **RandomForestRegressor**

: a machine learning model that uses an ensemble of decision trees to make predictions, particularly useful for regression tasks. It's known for its high accuracy and ability to handle large datasets with numerous input variables.

- Hyperparameter

- 'n_estimators': The number of trees in the forest
- 'max_depth': The maximum depth of each tree
- 'min_samples_split': The minimum number of samples required to split an internal node
- 'min_samples_leaf': The minimum number of samples required to be at a leaf node

Hyperparameter Tuning

1. **RandomForestRegressor**

- Check all combinations to determine best hyperparameters based on MSE

- Best Hyperparameter
    - 'n_estimators': 100
    - 'max_depth': 20
    - 'min_samples_split': 2
    - 'min_samples_leaf':1

```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}

results_df = pd.DataFrame(columns=["n_estimators", "max_depth", "min_samples_split", "min_samples_leaf", "MSE_train", "MSE_

for n_estimators in param_grid_rf['n_estimators']:
    for max_depth in param_grid_rf['max_depth']:
        for min_samples_split in param_grid_rf['min_samples_split']:
            for min_samples_leaf in param_grid_rf['min_samples_leaf']:
                model = RandomForestRegressor(n_estimators=n_estimators,
                                              max_depth=max_depth,
                                              min_samples_split=min_samples_split,
                                              min_samples_leaf=min_samples_leaf,
                                              random_state=0)
                model.fit(X_train, y_train)
                predictions_train = model.predict(X_train)
                predictions_valid = model.predict(X_valid)
                mse_train = mean_squared_error(y_train, predictions_train)
                mse_valid = mean_squared_error(y_valid, predictions_valid)

                temp_df = pd.DataFrame({
                    "n_estimators": [n_estimators],
                    "max_depth": [max_depth],
                    "min_samples_split": [min_samples_split],
                    "min_samples_leaf": [min_samples_leaf],
                    "MSE_train": [mse_train],
                    "MSE_valid": [mse_valid]
                })

                results_df = pd.concat([results_df, temp_df], ignore_index=True)

best_hyperparams = results_df.loc[results_df["MSE_valid"].idxmin()]
print("-------------Best Hyperparameter------------")
print('Best hyperparameters:', best_hyperparams)
```

Hyperparameter Tuning

2. XGBoostRegressor

: The XGBoostRegressor employs gradient boosting to combine decision trees for precise predictions, excelling in speed and accuracy on intricate datasets, with tunable hyperparameters for performance refinement.

- Hyperparameter

- 'n_estimators': The number of boosting stages to be run
- 'max_depth': The maximum depth of a tree
- 'learning_rate': The step size shrinkage used in updating tree weights

Hyperparameter Tuning

2. XGBoostRegressor

- Check all combinations to determine best hyperparameters based on MSE

- Best Hyperparameter
  - 'n_estimators': 300
  - 'max_depth': 3
  - 'learning rate': 0.1

```
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2]
}

results_df_xgb = pd.DataFrame(columns=["n_estimators", "max_depth", "learning_rate", "MSE_train", "MSE_valid"])

for n_estimators in param_grid_xgb['n_estimators']:
    for max_depth in param_grid_xgb['max_depth']:
        for learning_rate in param_grid_xgb['learning_rate']:
          model = XGBRegressor(n_estimators=n_estimators,
                               max_depth=max_depth,
                               learning_rate=learning_rate,
                               random_state=0)
          model.fit(X_train, y_train)
          predictions_train = model.predict(X_train)
          predictions_valid = model.predict(X_valid)
          mse_train = mean_squared_error(y_train, predictions_train)
          mse_valid = mean_squared_error(y_valid, predictions_valid)
          temp_df_xgb = pd.DataFrame({
                        "n_estimators": [n_estimators],
                        "max_depth": [max_depth],
                        "learning_rate": [learning_rate],
                        "MSE_train": [mse_train],
                        "MSE_valid": [mse_valid]
                    })
          results_df_xgb = pd.concat([results_df_xgb, temp_df_xgb], ignore_index=True)

best_hyperparams_xgb = results_df_xgb.loc[results_df_xgb["MSE_valid"].idxmin()]

print("------------Best Hyperparameter-----------")
print('Best hyperparameters for XGBRegressor:', best_hyperparams_xgb)
```

Hyperparameter Tuning

3. DecisionTreeRegressor

: This model is a single decision tree and can be used as a baseline to compare the more complex ensemble methods. Decision trees are easy to interpret and can model complex decision boundaries.

- Hyperparameter

  - 'max_depth': The maximum depth of tree
  - 'min_samples_split': The minimum number of samples required to split an internal node
  - 'min_samples_leaf': The minimum number of samples required to be at a leaf node

Hyperparameter Tuning

3. DecisionTreeRegressor

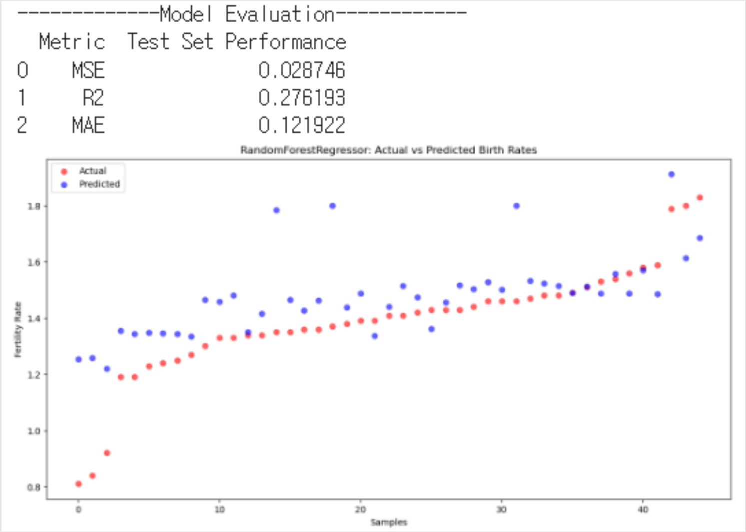
- Check all combinations to determine best hyperparameters based on MSE

- Best Hyperparameter
    - 'max_depth': 10
    - 'min_samples_split': 2
    - 'min_samples_leaf':4

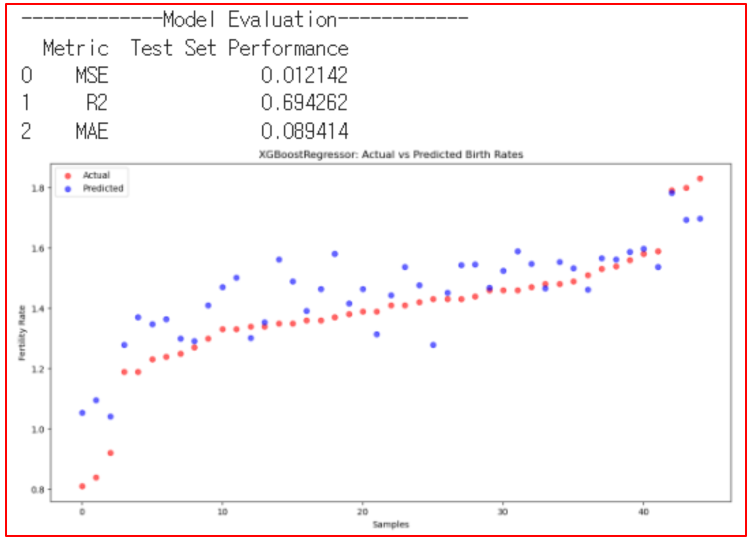
```
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}

results_df_dt = pd.DataFrame(columns=["max_depth", "min_samples_split", "min_samples_leaf", "MSE_train", "MSE_valid"])

for max_depth in param_grid_dt['max_depth']:
    for min_samples_split in param_grid_dt['min_samples_split']:
        for min_samples_leaf in param_grid_dt['min_samples_leaf']:
            model = DecisionTreeRegressor(max_depth=max_depth,
                                          min_samples_split=min_samples_split,
                                          min_samples_leaf=min_samples_leaf,
                                          random_state=0)
            model.fit(X_train, y_train)
            predictions_train = model.predict(X_train)
            predictions_valid = model.predict(X_valid)
            mse_train = mean_squared_error(y_train, predictions_train)
            mse_valid = mean_squared_error(y_valid, predictions_valid)

            temp_df_dt = pd.DataFrame({
                "max_depth": [max_depth],
                "min_samples_split": [min_samples_split],
                "min_samples_leaf": [min_samples_leaf],
                "MSE_train": [mse_train],
                "MSE_valid": [mse_valid]
            })

            results_df_dt = pd.concat([results_df_dt, temp_df_dt], ignore_index=True)

best_hyperparams_dt = results_df_dt.loc[results_df_dt["MSE_valid"].idxmin()]
print("-------------Best Hyperparameter------------")
print('Best hyperparameters for DecisionTreeRegressor:', best_hyperparams_dt)
```

Best Model

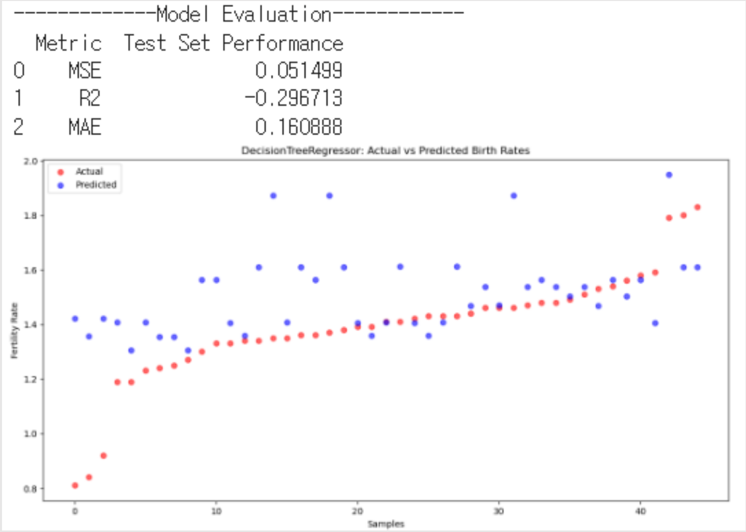
- Compare Performance



[RandomForestRegressor]



[XGBoostRegressor]



[DecisionTreeRegressor]

Model Selection

•XGBoostRegressor : Based on the results, the XGBoostRegressor is the best choice among the three models. It achieves the lowest mean squared error on the test set and the highest R^2 score.

04 Conclusion

Result Interpretation

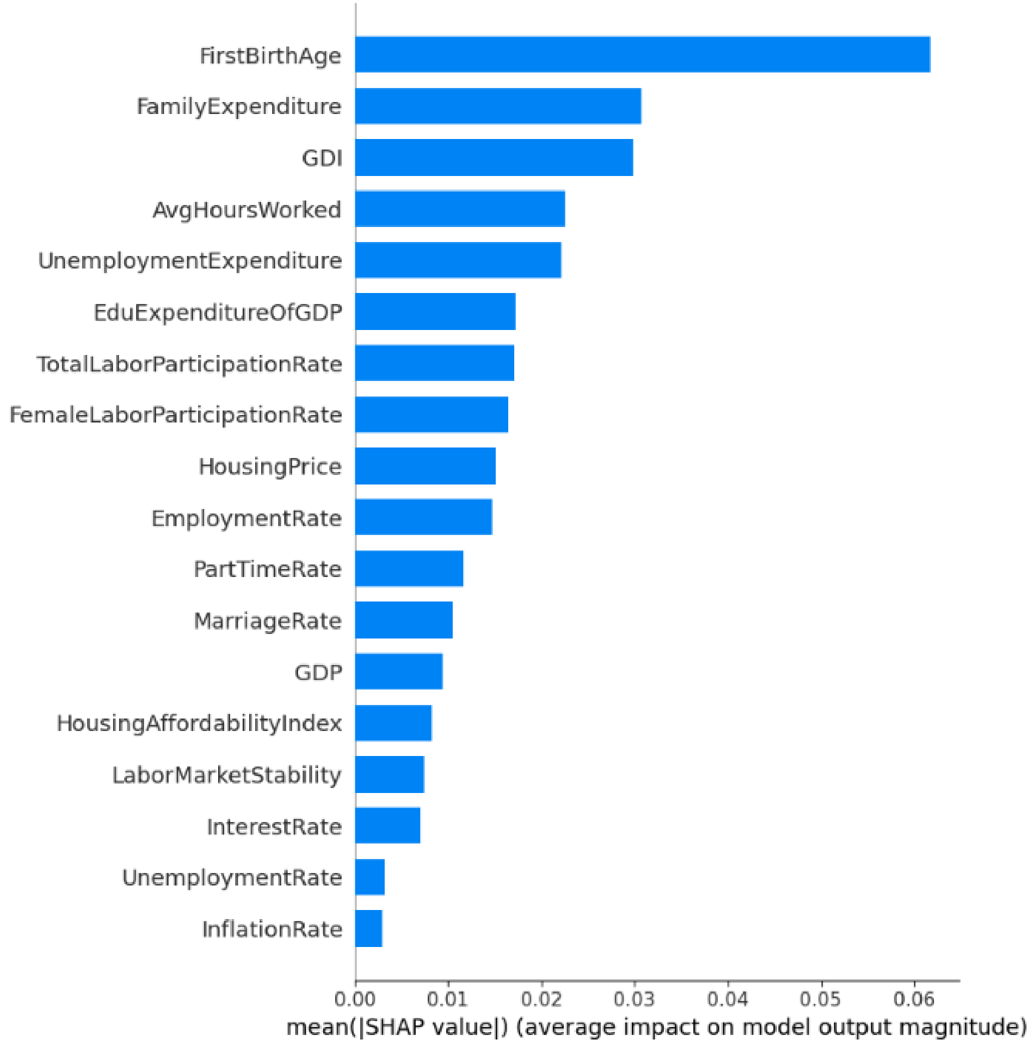
- What is "SHAP"

: Shapley Additive exPlanations (SHAP) is a conceptual framework for describing the predictions of machine learning models. SHAP values represent how much each feature contributed to the model's predictions, and help explain the model's predictions

- Why we used "SHAP"

: To get a rough idea of which variables are most important to the model, you can check the results by plotting the SHAP - values of all the variables for all the samples.

Result Interpretation



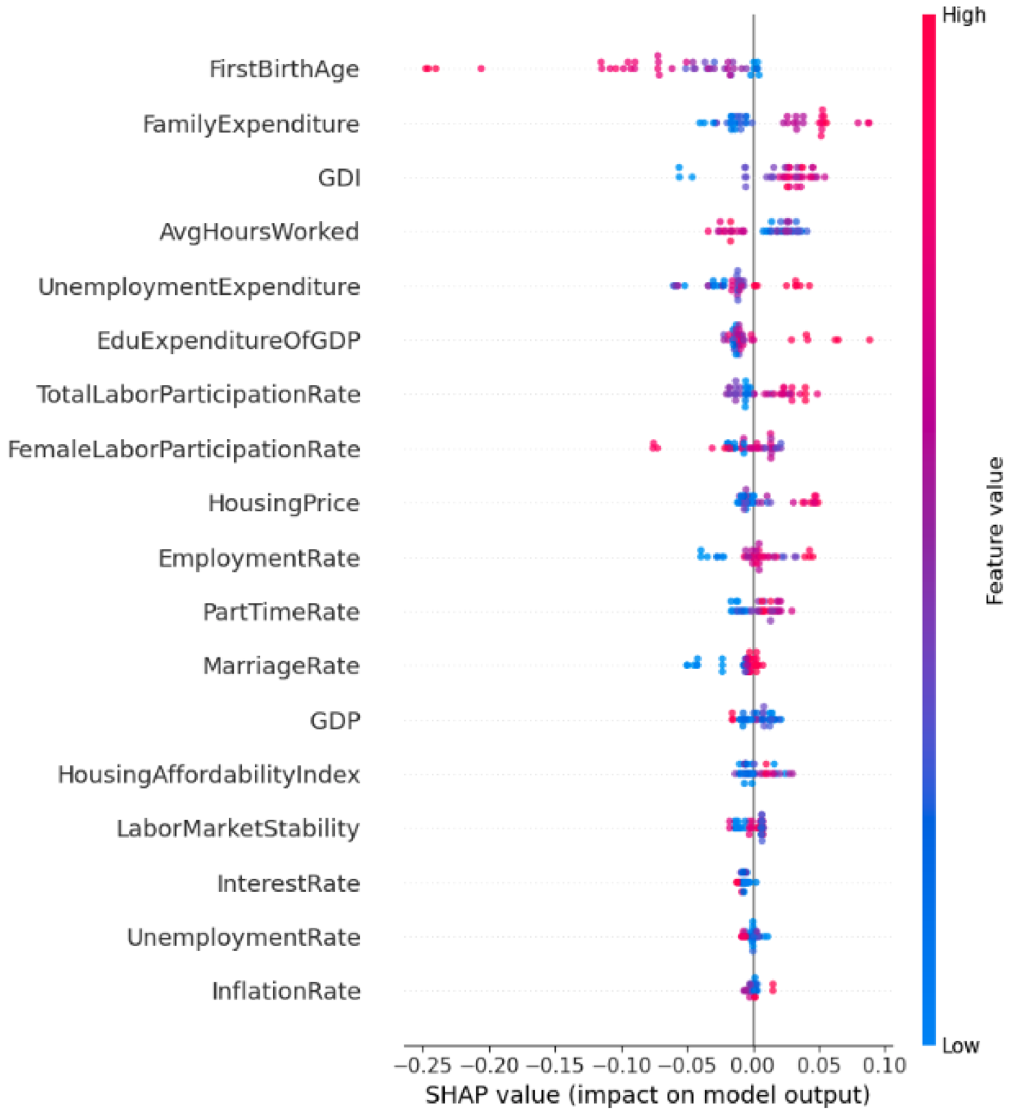
- The graph is a bar chart displaying the mean SHAP (SHapley Additive exPlanations) values, which indicate the average impact of each variable on the model's output.

- Large Impact: FirstBirthAge, FamilyExpenditure, GDI
- Small Imapact: InflationRate, UnemploymentRate

Result Interpretation

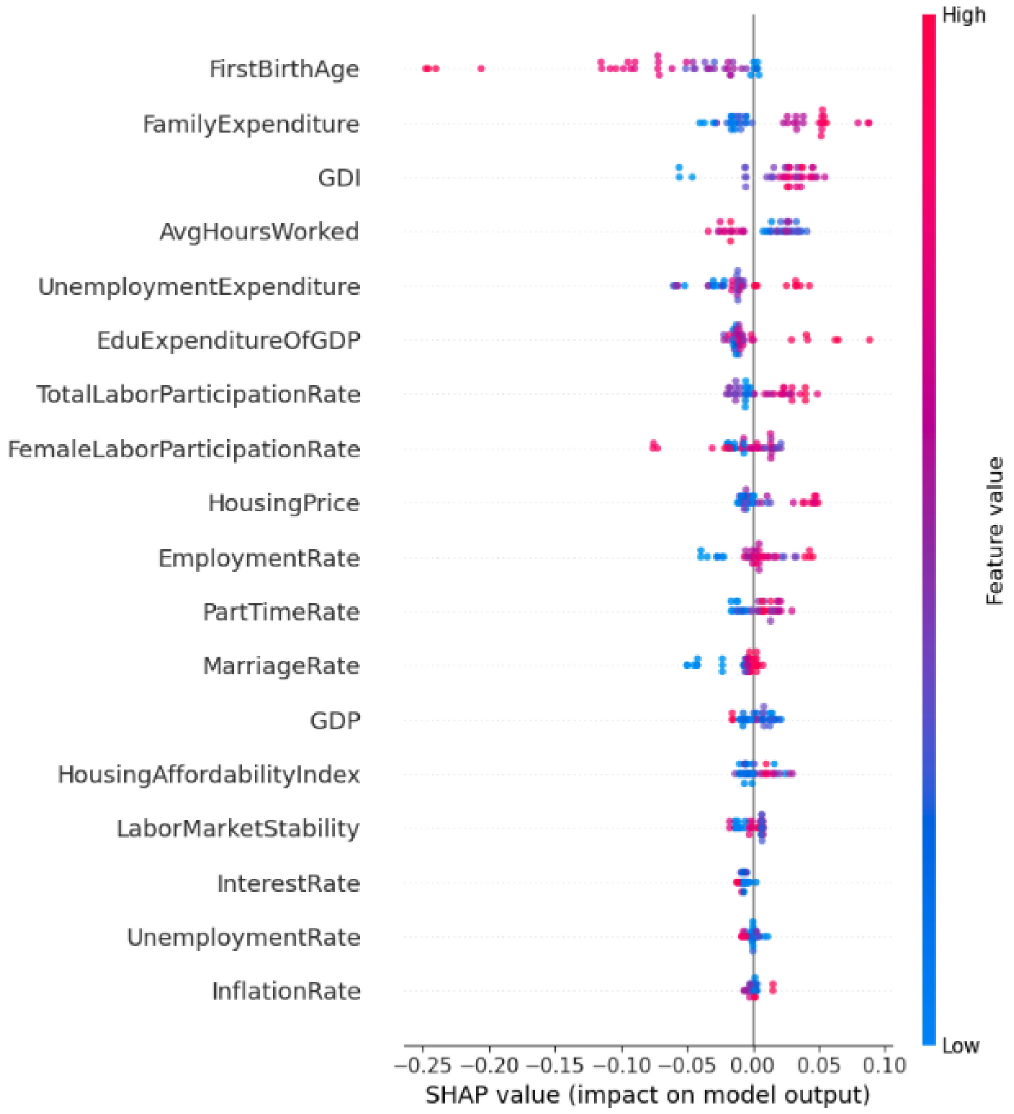
- This graph is a SHAP value plot, showing how different features influence the output of a predictive model. Each dot represents the effect of a feature for an individual prediction, with red dots indicating a high feature value and blue dots a low value.

- For example, it shows that a high 'FirstBirthAge' lowers the expected fertility rate.



Result Interpretation

	Feature Value	Expected Result
FirstBirthAge	↑	↓
FamilyExpenditure	↑	↑
GDI	↑	↑
AvgHoursWorked	↑	↓
UnemploymentExpenditure	↑	↑
EduExpenditureOfGDP	↑	↑
TotalLaborParticipationRate	↑	↑

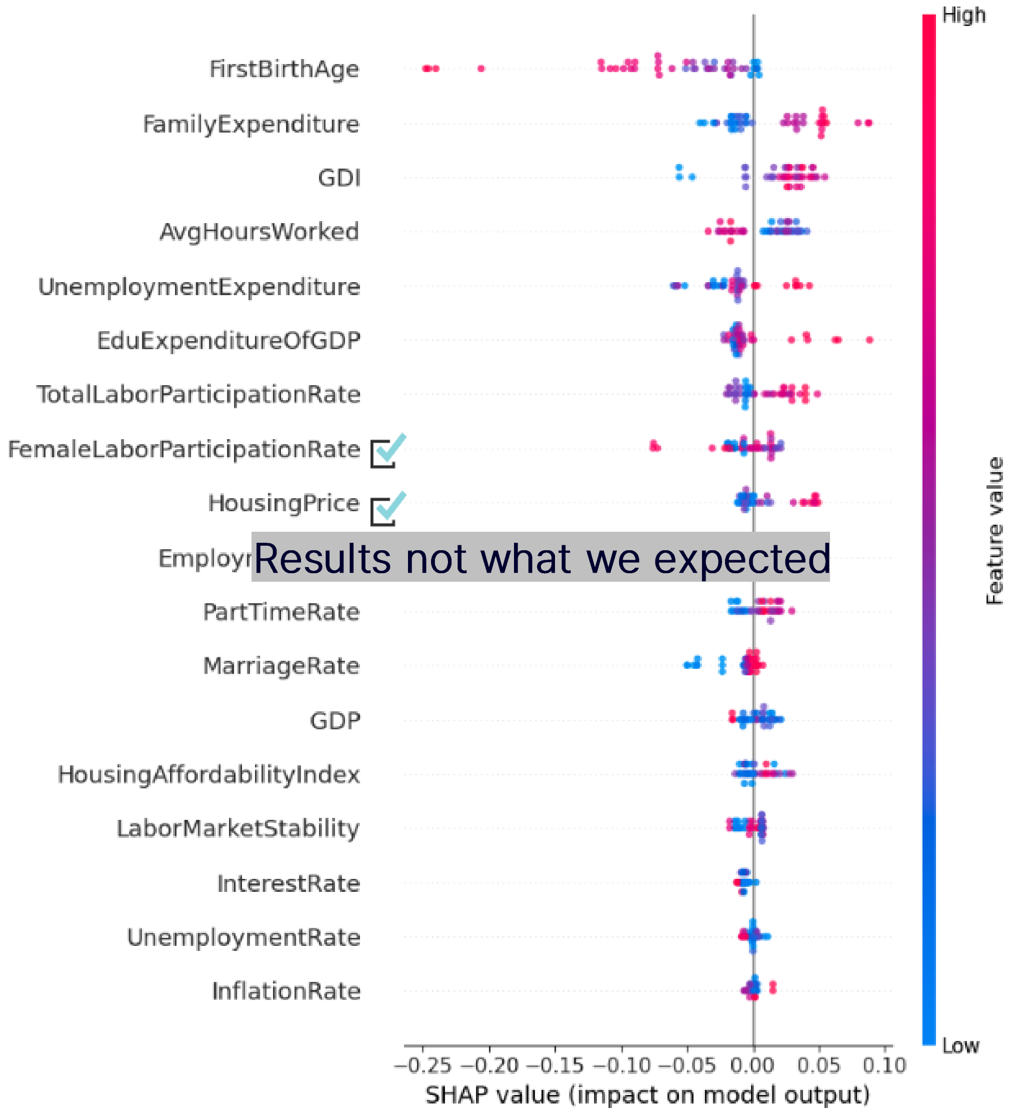


Result Interpretation

- 'FemaleLaborParticipationRate'

: A rise in the 'Female Labor Participation Rate' is now linked to a more positive fertility rate.

This unexpected outcome may suggest evolving societal dynamics, possibly indicating that increased female workforce participation is no longer a significant deterrent(억제) to higher fertility rates.

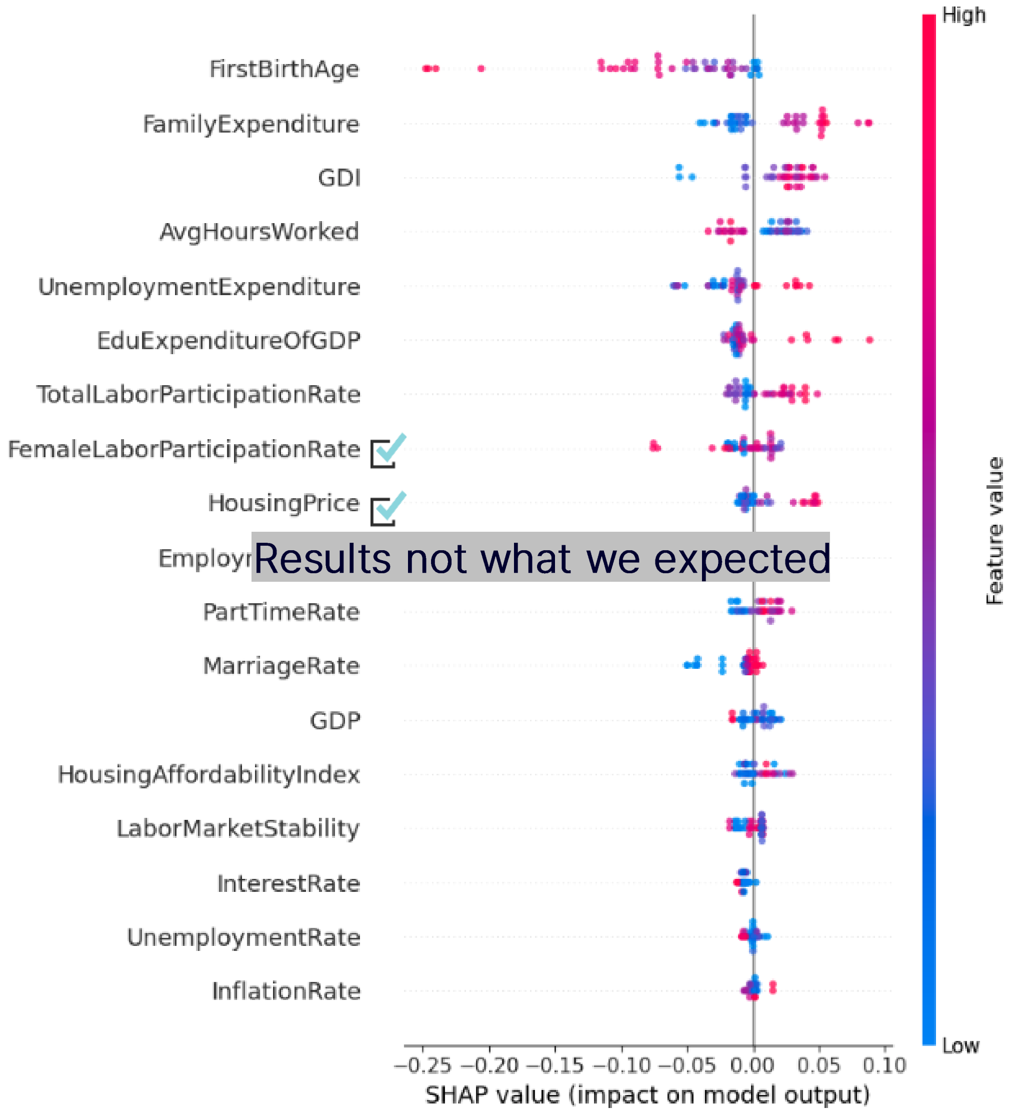


Result Interpretation

- 'HousingPrice'

: The results contradict our conventional wisdom that higher house prices are associated with higher fertility.

This unexpected positive correlation is due to a variety of social factors. For example, the global trend of house price growth.

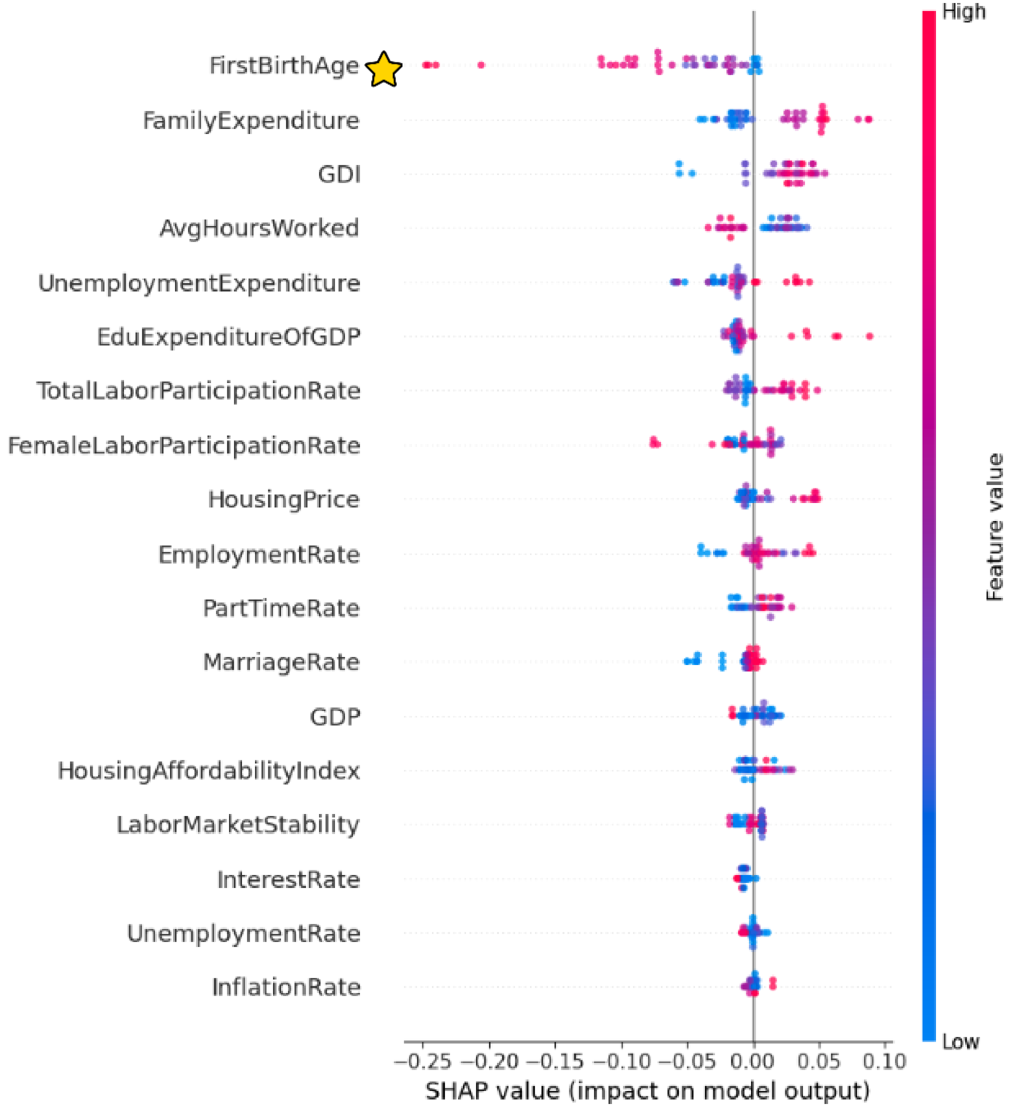


Suggestion

- ‘FirstBirthAge’ : Mean age of women at first childbirth

\- Indicating that the increase in the first birth age is closely related to the decrease in fertility rate.

→ It is important for the country to understand why the first childbirth is delayed. Policies or programs that support young parents, such as family planning education and childcare support, can increase the fertility rate.



Suggestion

- 'FamilyExpenditure ' : Public spending on family benefits, including financial support that is exclusively for families and children.

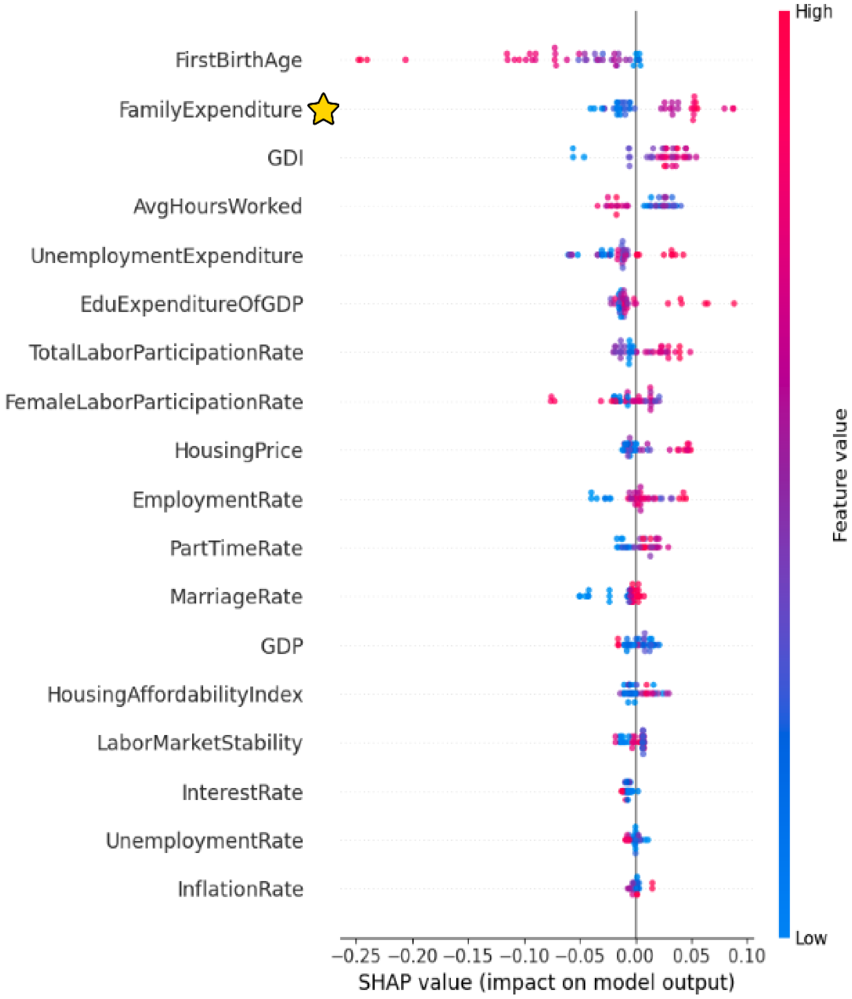
– Indicating that the increase in the public spending on family benefit is closely related to the increase in fertility rate.

Case in France)

- One of the countries that has solved the birth rate problem

- Differences in FamilyExpenditurein Korea and France

[France]
- 1M~2.5M Won per child per month(under the age of 20)
[Korea]
- 1M Won per child per month(under the age of 8)



→ Therefore, various family support policies such as support for parenting time, child care and education services, including cash support such as child allowance, should be implemented steadily.

Limitation

- **Complex social and cultural influences**: Fertility rates are strongly influenced by social and cultural factors. While XGBoost models can learn patterns in data, predicting complex social behavior can be difficult. In particular, Fertility rates can be influenced not only by policy and economic factors, but also by culture and family structure.

- **Data limitations**: The data period we were able to collect was from 1990 to 2021, but the amount of data was insufficient, and the missing data was handled by time interpolation, so it is difficult to trust the accuracy of the forecast 100% due to insufficient quantity and quality.

- **Policy selection problem**: Features like 'GDI(3rd highest)' are important, but they don't change easily as policies change. Furthermore, selecting and proposing a policy with a small difference in outcome for SHAP is a challenging problem. ('AvgHoursWorked' vs. 'unEmploymentExpenditure')

Github Link

- https://github.com/oosedus/BirthratePrediction

Appendix

- https://www.worldbank.org/en/home
- https://ourworldindata.org/
- https://data.oecd.org/
- https://stats.oecd.org/
- https://kosis.kr/index/index.do
- https://m.blog.naver.com/marron-glace-paris/223253810198
- https://www.kiri.or.kr/report/downloadFile.do?docId=29
- https://www.bokjiro.go.kr/ssis-tbu/twataa/wlfareInfo/moveTWAT52011M.do?wlfareInfoId=WLF00001171&wlfareInfoReldBztpCd=01

Q & A