Data Science CSCI 3320
Project: Fertility rate

Sarah Abu Khotti, 220202093

About the domain

About the domain

The fertility rate is the average number of children that a woman has over the course of her reproductive years. The fertility rate can have a significant impact on the population size of a country or region.

If the fertility rate is high, the population can grow quickly. This can lead to increased economic growth and development, but it can also put strain on resources and the environment. On the other hand, if the fertility rate is low, the population may not be able to replace itself over time, leading to an aging population and potential decline in the workforce.

There are many factors that can influence fertility rates, including cultural, social, economic, and environmental factors. Some measures that governments and organizations may take to address fertility rates include implementing family planning programs, providing education and access to birth control, and addressing economic and social factors that may affect the decision to have children.

There are several reasons why it might be useful to predict fertility rates:

1.Population planning: Fertility rates are an important factor in population planning and forecasting. By predicting future fertility rates, governments and organizations can make informed decisions about resources and policies related to population growth, such as housing, education, and healthcare.

2.Health and social policy: Fertility rates can have a significant impact on health and social policies, such as family planning and maternal health programs. By predicting fertility rates, policymakers can design and target interventions to address specific population needs.

3.Economic planning: Fertility rates can affect economic indicators such as labor force participation, savings, and consumption patterns. By predicting fertility rates, governments and businesses can make informed decisions about economic policy and investments.

4.Social and demographic research: Fertility rates are an important indicator of social and demographic trends, such as changes in family structure, gender roles, and reproductive health. By predicting fertility rates, researchers can gain insight into these trends and their impacts on society.

5.Overall, predicting fertility rates can help inform decision-making and research in a variety of fields, including public policy, health, economics, and sociology.

Since fertility rate is an important issue for world health organization and also a decline in fertility rates can lead to a number of social, economic, and demographic changes. Some potential impacts of a decline in fertility rates include:

1.Aging population: A decline in fertility rates can lead to an aging population, as there are fewer children being born to replace the aging population. This can have implications for social security, healthcare, and other systems that support the elderly.

2.Changes in family structure: A decline in fertility rates may lead to changes in family structure, such as smaller families or an increase in single-parent households.

3.Labor force changes: A decline in fertility rates can affect the size and composition of the labor force, as fewer people enter the workforce to replace retiring workers. This can have implications for economic growth and productivity.

4.Economic impacts: A decline in fertility rates may have economic impacts, such as changes in consumption patterns, savings, and investments. It may also affect the availability of skilled labor and the demand for certain goods and services.

About the data

About the data

The dataset source:

The World Bank is a global development organization that provides financial assistance and policy advice to countries around the world. As part of its mission, the World Bank collects and disseminates a wide range of data on development indicators, including data on economic, social, and environmental conditions.

The data they collected is sourced from several organizations:

(1) United Nations Population Division. World Population Prospects: 2022 Revision.

(2) Census reports and other statistical publications from national statistical offices.

(3) Eurostat: Demographic Statistics.

(4) United Nations Statistical Division. Population and Vital Statistics Reprot (various years)

(5) U.S. Census Bureau: International Database.

(6) Secretariat of the Pacific Community: Statistics and Demography Programme.

The World Bank's data organization is structured around various themes, such as education, health, agriculture, and infrastructure. The data is organized into databases and datasets, which can be accessed through the World Bank's website or through its APIs (Application Programming Interfaces). The World Bank also publishes a variety of reports and publications that use and analyze the data to provide insights on development issues.
The World Bank's data is widely used by researchers, policymakers, and development practitioners to inform decision-making and to track progress on global development goals.

Data context:

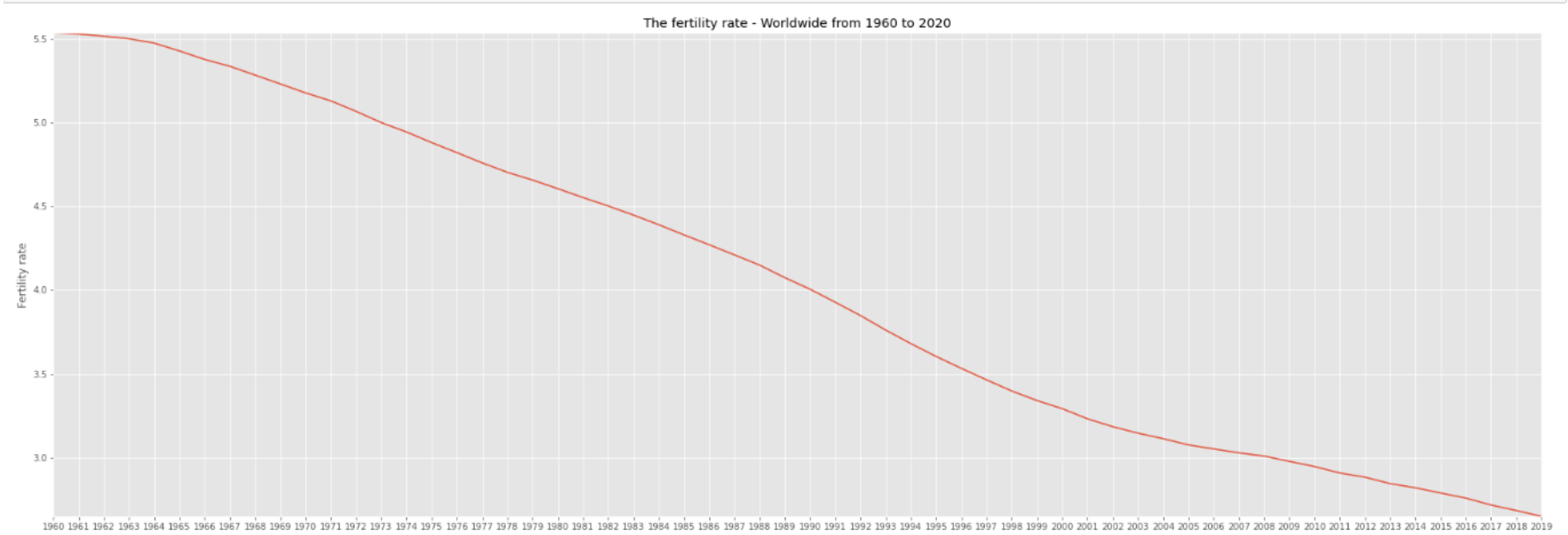
We have a dataset with a number of attributes such as country which takes String(object) values and is a total of 187 countries, and then each year from 1960 to 2020 are also attribute with their values as the fertility rate, there’s 62 rows.

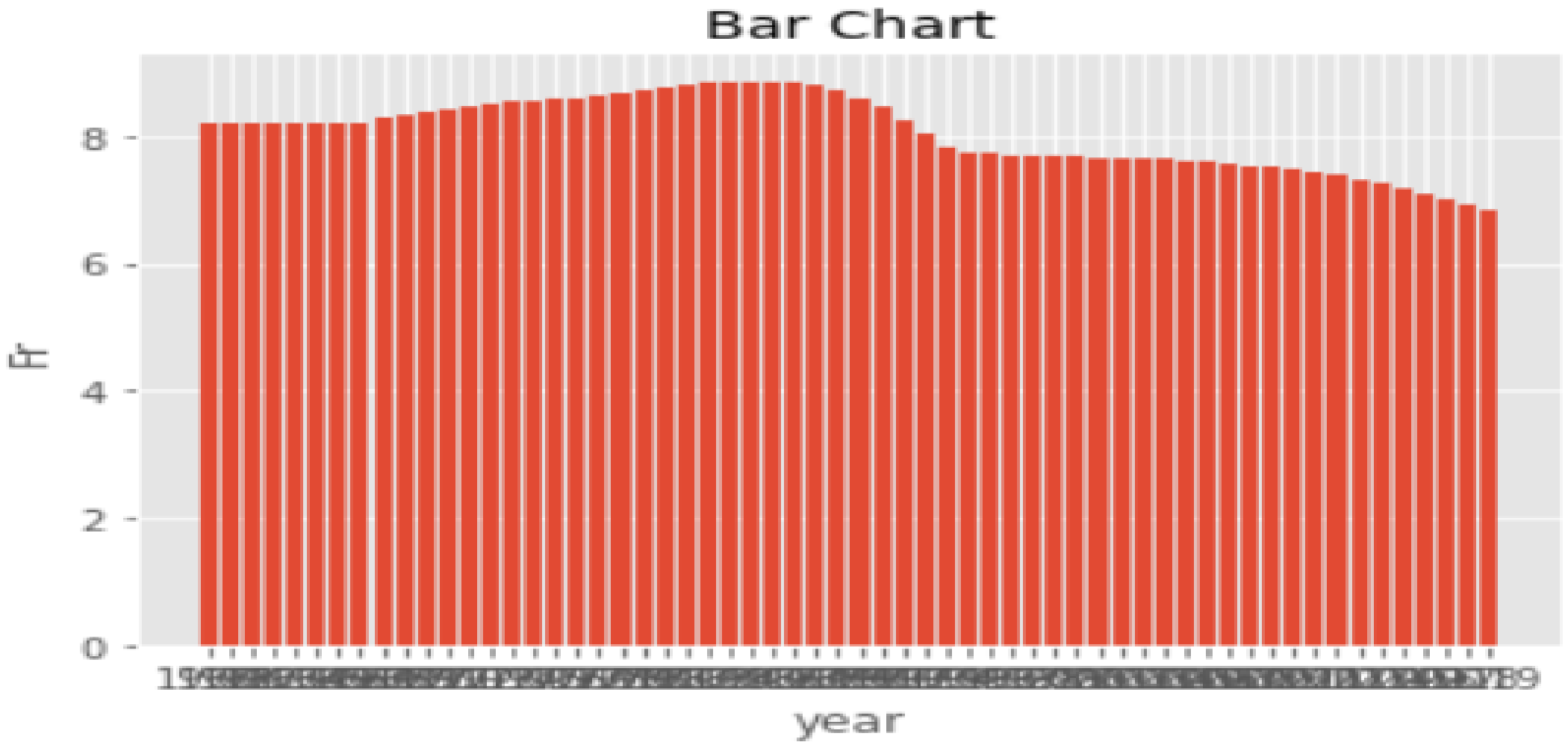
Data Exploration

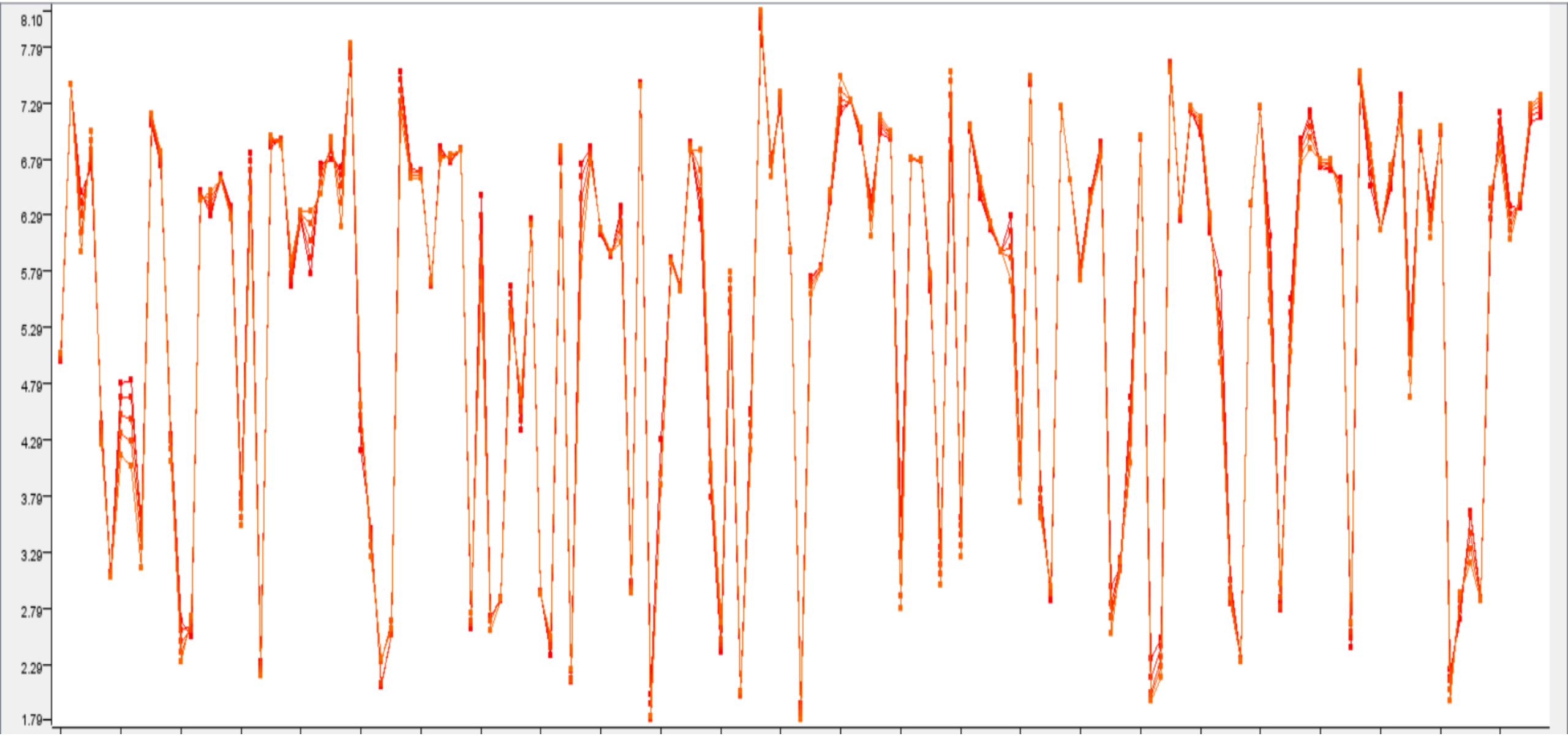
Data exploration

There’s no missing values in the data set, I used descriptive statistics and found the min, max, maiden, mean, count.

Data visualization:




Bar Chart
Fr
0
2
4
6
8
year


8.10
7.79
7.29
6.79
6.29
5.79
5.29
4.79
4.29
3.79
3.29
2.79
2.29
1.79

Data Preparation

Data preparation

We first handle missing data using python pandas by exploring the data I found no missing values so there was no need for the process of handling missing values, to be sure I changed the data set formation for easier access to data so all the year values are under one year attributes(transformation), all will be in the screenshots of the python code and the python code sent.

By reshaping data using the **pandas.melt()** function to reshape data from wide to long format, then Grouping and aggregating data: using the pandas.groupby() function to group data by one or more variables, and then apply an aggregation function (e.g. mean, sum, count) to compute summary statistics for each group.

Part	# of records	%
Train	X:8976 *Y:8976	80
Test	X:2224*y:2244	20

Descriptive Analytics

Descriptive analytics

I used K-mean clustering to analyze data and used Elbow Curve to get the right number of Cluster as will as silhouette score and found out the best number of clusters would be 3 or 2 so I went with 3 which is my initial choice since k- mean id an unsupervised learning algorithm I performed clustering on the initial data set.

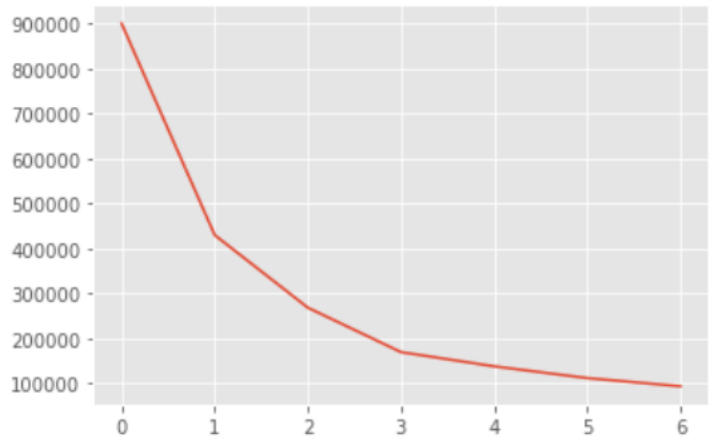
Association rule wouldn’t be of use for our data set since it only consists of one variable that is calculated and that doesn’t offer any ways of finding values that can be categorized. Since association rule algorithm is a technique for discovering relationships or patterns between items in a dataset. It is commonly used in market basket analysis to identify items that are frequently purchased together and that’s not what our data needs.

```
# Create the k-means model
kmeans = KMeans(n_clusters=3, max_iter=40).fit(clus)
```

```
# Elbow Curve to get the right number of Cluster
ssd = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=40)
    kmeans.fit(clus)

    ssd.append(kmeans.inertia_)

# plot the SSDs for each n_clusters
plt.plot(ssd)
```




```
# Silhouette analysis
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]

for num_clusters in range_n_clusters:

    # intialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(clus)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = silhouette_score(clus, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

```
For n_clusters=2, the silhouette score is 0.6058000067770053
For n_clusters=3, the silhouette score is 0.5533000931314114
For n_clusters=4, the silhouette score is 0.5150829546727067
For n_clusters=5, the silhouette score is 0.5291659232241898
For n_clusters=6, the silhouette score is 0.4955393372353631
For n_clusters=7, the silhouette score is 0.45821658896500644
For n_clusters=8, the silhouette score is 0.45065151195239134
```

```
# final kmean model

# Create the k-means model
kmeans = KMeans(n_clusters=3, max_iter=40).fit(clus)
```

```
kmeans.labels_
```

```
array([2, 2, 2, ..., 1, 1, 1])
```

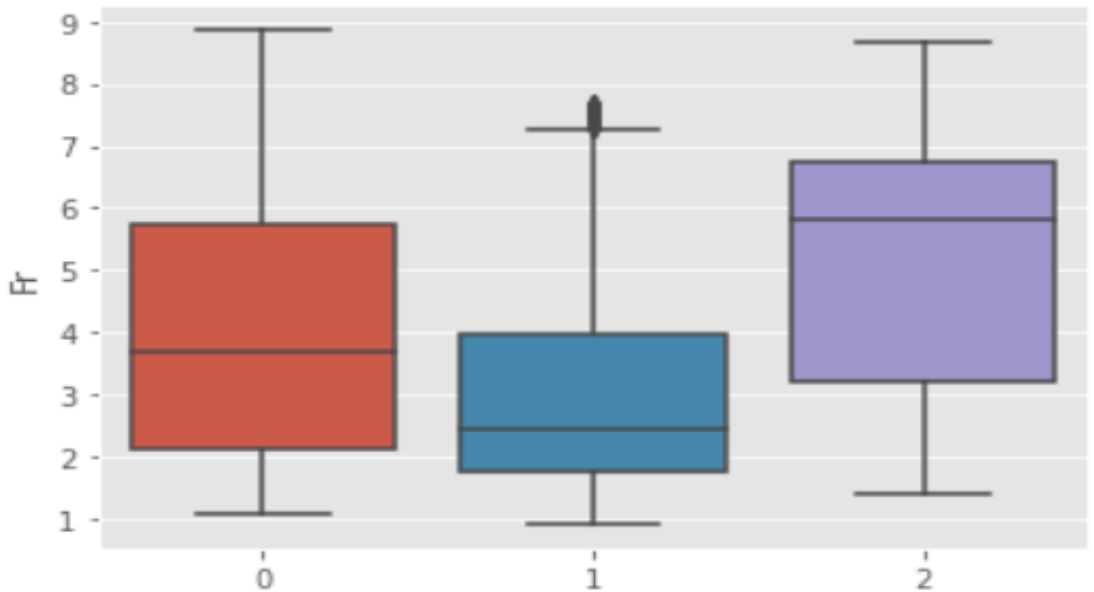
```
 # assign the label
clus['Cluster_Id'] = kmeans.labels_
clus.head()
```

	year	Fr	Cluster_Id
0	1960	4.98	2
1	1960	7.45	2
2	1960	6.49	2
3	1960	7.52	2
4	1960	6.71	2

```
# Box plot to visualize Cluster Id  Fr

sns.boxplot(x='Cluster_Id', y='Fr', data=clus)
```

```
<AxesSubplot:xlabel='Cluster_Id', ylabel='Fr'>
```



Predictive Analytics

Predictive analytics

Linear Regression:

I used linear regression since it’s appropriate for handling a continuous target variable (such as fertility rate) and a small number of numerical features, we could use linear regression to model the relationship between the features and the target variable.

Linear regression is a simple and interpretable model that can be used to make predictions and understand the importance of different features.

```
# linear ragrission prediction model
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
LinearRegression()
```

```
#Retrieve the intercept
print(regressor.intercept_)
```

```
113.0353365894169
```

```
#Retrieve the slop
print(regressor.coef_)
```

```
[-0.05478376]
```

```
#Comparing the predicted value to the actual value
y_pred = regressor.predict(X_test)
y_pred = pd.DataFrame(y_pred, columns= ["predicted"]
y_pred
```

	predicted
0	4.892190
1	3.358245
2	4.563488
3	3.577380
4	4.892190
...	...
2239	4.837406
2240	4.618271
2241	5.549595
2242	4.289569
2243	4.344353

2244 rows × 1 columns

```
y_test
```

```
2649    5.83
7867    1.93
3810    3.25
7260    1.13
2711    6.40
       ...
2811    6.65
3732    2.79
481     3.24
4683    2.56
4535    1.40
Name: Fr, Length: 2244, dtype: float64
```

```
# evaluating the algorithm
print('Mean absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regressor.score(X_test, y_test))
```

```
Mean absolute Error: 263.00222816399287
Mean squared Error: 115241.53231862745
Root Mean squared Error: 339.472432339693
Variance score: 0.22
```

Decision trees:

If we have a dataset with a categorical target variable (such as low, medium, or high fertility rate) and a mix of numerical and categorical features, we could use a decision tree model to learn the decision boundaries that separate the different fertility rate categories. Decision trees are easy to interpret and can handle complex feature interactions.

Not the best choice for our dataset.

```
#convert y values to categorical values
y = fertility_rate['Fr']
lab = preprocessing.LabelEncoder()
y_transformed = lab.fit_transform(y)
X = fertility_rate.drop(columns=['Fr','Country'])
X_train, X_test, y_ttrain, y_ttest = train_test_split(X, y_transformed, test_size=0.2, random_state=1)
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_ttrain)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_ttest, y_pred))
```

```
Accuracy: 0.0022281639928698753
```

```
# we can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_ttrain)
```


```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_ttrain)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_ttest, y_pred))
```

```
Accuracy: 0.0022281639928698753
```

```
# we can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_ttrain)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```


```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_ttest, y_pred))
```

```
Accuracy: 0.007575757575757576
```

Random forests

Since we have a large dataset, we could use a random forest model to make predictions. Random forests are an ensemble learning method that combines the predictions of many decision trees to improve the accuracy and stability of the model.

Not the best choice for our dataset.

```
#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_ttrain)

y_pred=clf.predict(X_test)
```

```
# metrics module for accuracy calculation
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_ttest, y_pred))
```

```
Accuracy: 0.001336898395721925
```

```
#Creating a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_ttrain)

# prediction on test set
y_pred=clf.predict(X_test)

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_ttest, y_pred))
```

```
Accuracy: 0.0022281639928698753
```

Comparing the models:

The machine learning model that was the most fitting was the linear regression model since we could give it the data values immediately while the other two we needed to transform the fertility rate values since they don't handle numerical values, also the linear regression is more appropriate for handling a continuous target variable.

is a good choice for datasets with a small number of features and a linear relationship between the features and the target. Linear regression is fast to train and predict, but it is sensitive to outliers and we don't have outliers in our data set.

decision tree is prone to overfitting and may not be as accurate as linear regression or random forests on some datasets.

Linear regression:
Mean absolute Error: 263.00222816399287
Mean squared Error: 115241.53231862745
Root Mean squared Error: 339.472432339693
Variance score: 0.22 (highest accuracy)

Decision tree:
Accuracy: 0.00757575757575757576 (second to highest)

Random Forest:
Accuracy: 0.0022281639928698753 (lowest accuracy)

Conclusions

- since leaning about the declaration of fertility rate made courtiers perform precautionary measures that will help stopping the sharp decline.
- The data set provided lacks more attributes to provide us with more inside knowledge which will help in the overall prediction model accuracy.
- Using k-means provide us with clusters of data that helps the prediction model overall performance, generalization, and interpretability of the model.
- Regression models are the best way of handling data with a little number of attribute and mostly numerical.