Table of Contents

Step 1: Dataset Selection

We chose the **World Happiness Report** dataset.

It has features such as Happiness scored according to economic production, social support, etc.

https://www.kaggle.com/datasets/unsdsn/world-happiness Dataset link on Kaggle

Step 2: Scenario/About Dataset:

This dataset contains global data on happiness scores and factors that contribute to happiness, such as GDP per capita (Economy), family, social support, life expectancy ...

Problem Statement:

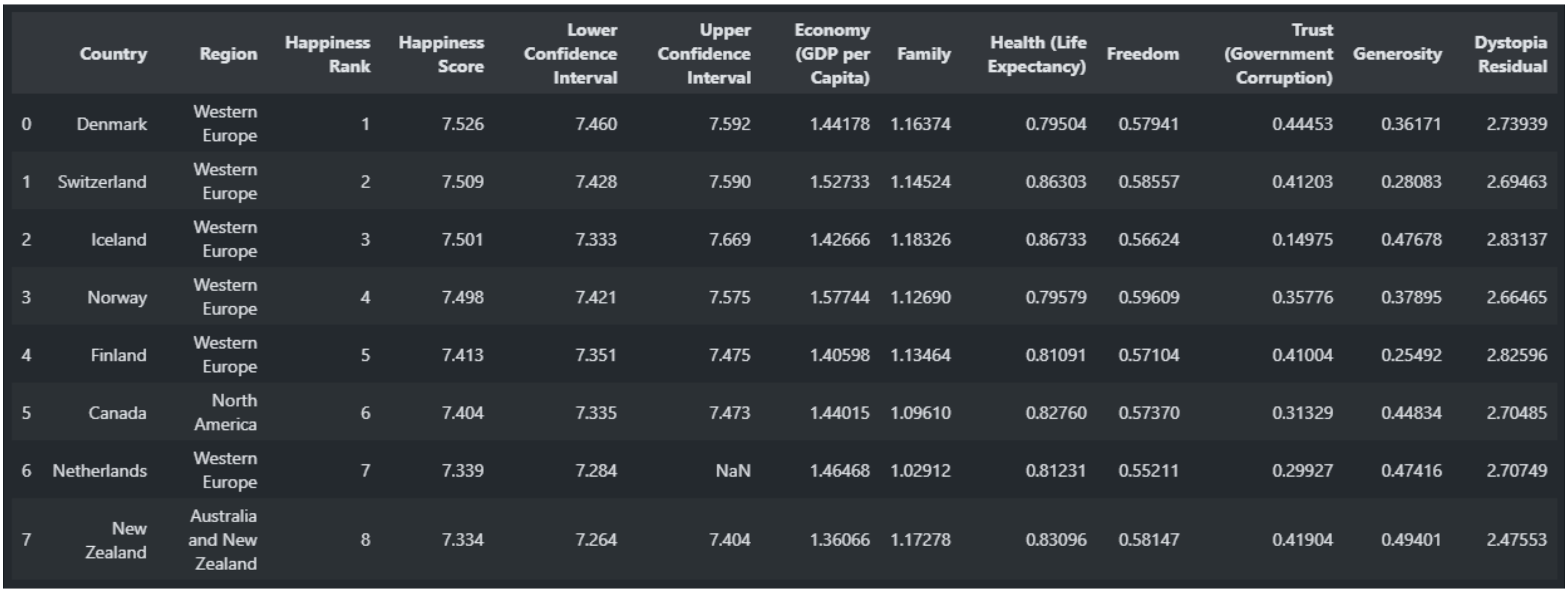
- We want to understand global happiness trends to design effective policy recommendations and allocate resources for improving well-being across different regions. With the World Happiness dataset, which includes metrics such as GDP per capita, social support, life expectancy, freedom, corruption ... Machine Learning will allow us to identify key trends and actionable insights

How Machine Learning is Useful on this dataset:

- **Regression**: To predict happiness scores based on measurable socio-economic indicators, providing a model able to predict happiness scores and simulate the impact of policy changes.
- **Clustering**: To group countries into similar categories based on happiness profiles or socioeconomic factors, revealing hidden patterns regional similarities and global trends
- **Classification**: To categorize countries into predefined happiness levels (e.g., "High Happiness," "Medium Happiness," "Low Happiness") for prioritizing regions needing immediate attention

Step 3: Data Loading

Here is a short view of the dataset after loading it into python.

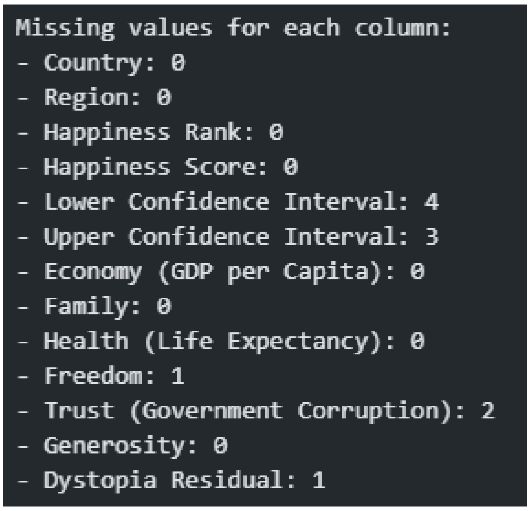


	Country	Region	Happiness Rank	Happiness Score	Lower Confidence Interval	Upper Confidence Interval	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Denmark	Western Europe	1	7.526	7.460	7.592	1.44178	1.16374	0.79504	0.57941	0.44453	0.36171	2.73939
1	Switzerland	Western Europe	2	7.509	7.428	7.590	1.52733	1.14524	0.86303	0.58557	0.41203	0.28083	2.69463
2	Iceland	Western Europe	3	7.501	7.333	7.669	1.42666	1.18326	0.86733	0.56624	0.14975	0.47678	2.83137
3	Norway	Western Europe	4	7.498	7.421	7.575	1.57744	1.12690	0.79579	0.59609	0.35776	0.37895	2.66465
4	Finland	Western Europe	5	7.413	7.351	7.475	1.40598	1.13464	0.81091	0.57104	0.41004	0.25492	2.82596
5	Canada	North America	6	7.404	7.335	7.473	1.44015	1.09610	0.82760	0.57370	0.31329	0.44834	2.70485
6	Netherlands	Western Europe	7	7.339	7.284	NaN	1.46468	1.02912	0.81231	0.55211	0.29927	0.47416	2.70749
7	New Zealand	Australia and New Zealand	8	7.334	7.264	7.404	1.36066	1.17278	0.83096	0.58147	0.41904	0.49401	2.47553

We can see all the features and have an overview of the values for the different countries.

Step 4: Data Wrangling or Data Pre-processing

Handle missing values

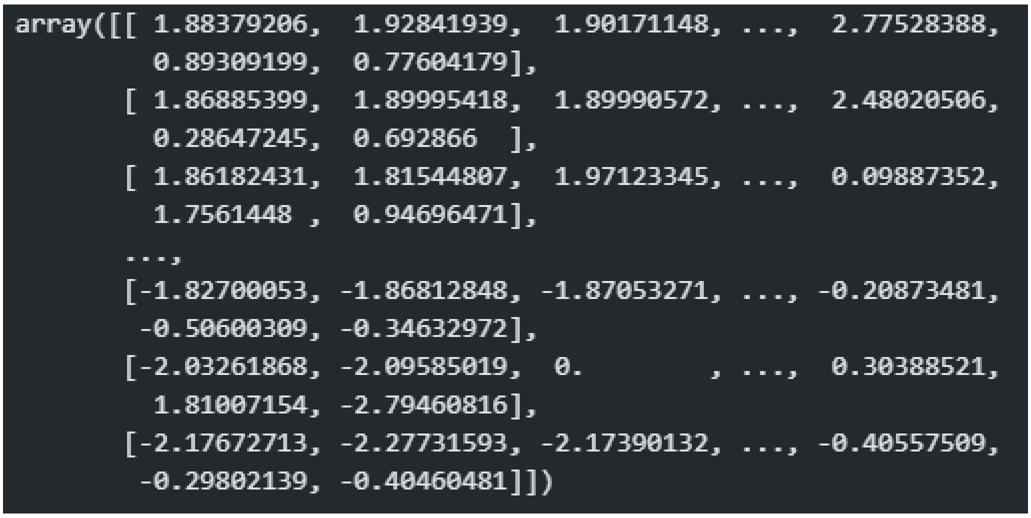


```
Missing values for each column:
- Country: 0
- Region: 0
- Happiness Rank: 0
- Happiness Score: 0
- Lower Confidence Interval: 4
- Upper Confidence Interval: 3
- Economy (GDP per Capita): 0
- Family: 0
- Health (Life Expectancy): 0
- Freedom: 1
- Trust (Government Corruption): 2
- Generosity: 0
- Dystopia Residual: 1
```

With a simple python script, we can see that many values are missing for each column, we will handle them by filling them with the **mean of the column**

Standardize the data

Now that no more values are missing, we can **Standardize the data** after Dropping the useless columns (String values such as “Country”, “Region” and global descending rank “Happiness Rank”)

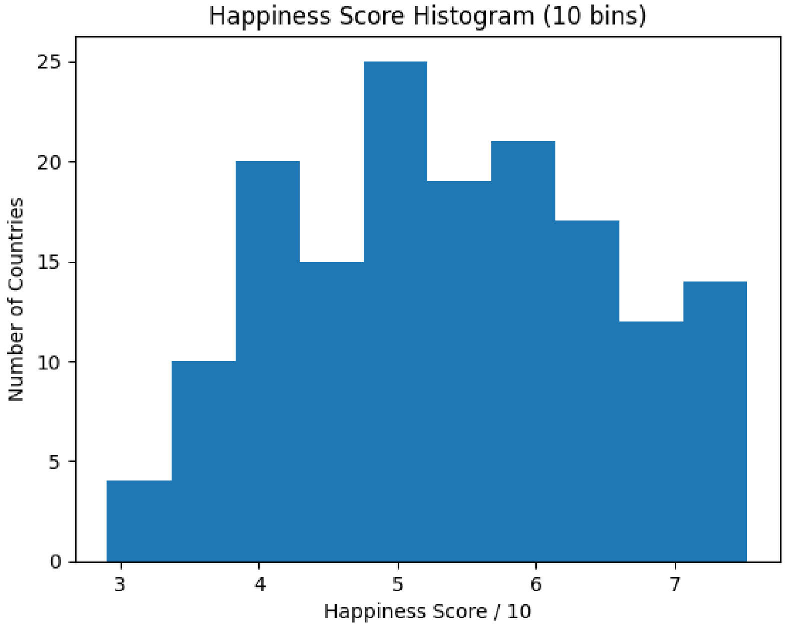


```
array([[ 1.88379206,  1.92841939,  1.90171148, ...,  2.77528388,
         0.89309199,  0.77604179],
       [ 1.86885399,  1.89995418,  1.89990572, ...,  2.48020506,
         0.28647245,  0.692866  ],
       [ 1.86182431,  1.81544807,  1.97123345, ...,  0.09887352,
         1.7561448 ,  0.94696471],
       ...,
       [-1.82700053, -1.86812848, -1.87053271, ..., -0.20873481,
        -0.50600309, -0.34632972],
       [-2.03261868, -2.09585019,  0.        , ...,  0.30388521,
         1.81007154, -2.79460816],
       [-2.17672713, -2.27731593, -2.17390132, ..., -0.40557509,
        -0.29802139, -0.40460481]])
```

Step 5: Exploratory Data Analysis

In order to understand better the dataset and identify relevant features, we will apply various data exploration techniques.

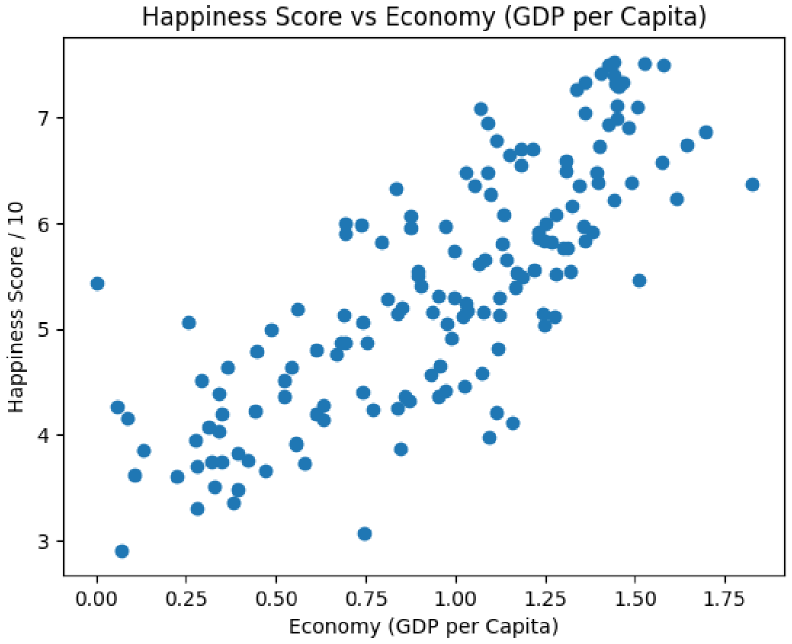
Happiness Score Histogram



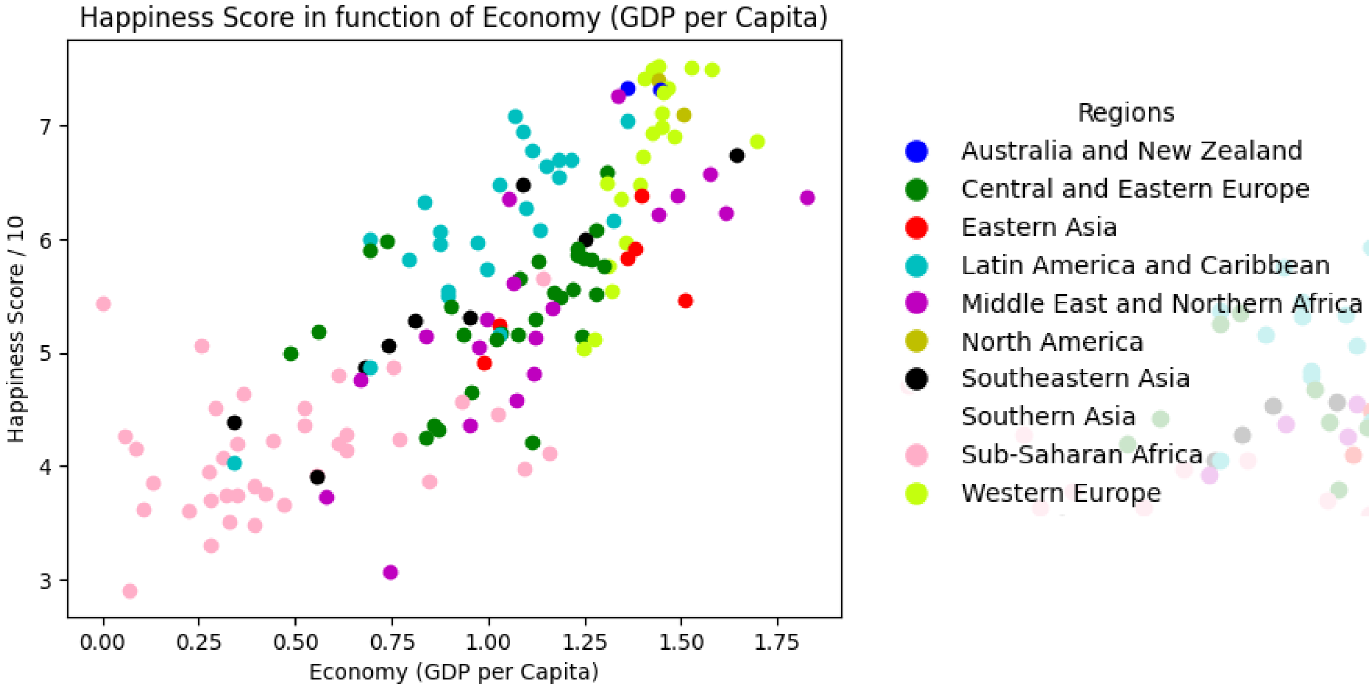
This histogram reveals the average World happiness. We can see many countries manage to have a Happiness score above 7/10 although **the most frequent values are between 4 and 5 /10**.

A first intuition would be to think that **many variables are linearly linked with the Happiness Score**. Let’s see if this intuition is correct.

Happiness Score vs Economy (GDP per Capita)



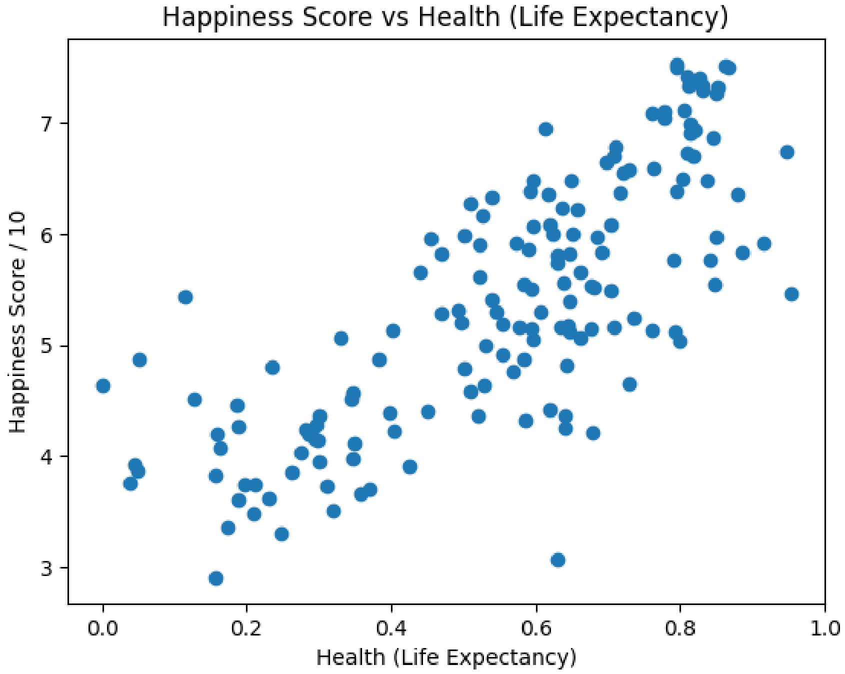
This graph seems to show that the more money a country produces, the happier its inhabitants will be.



Here is the same graph as above but colored by which region the countries are in.

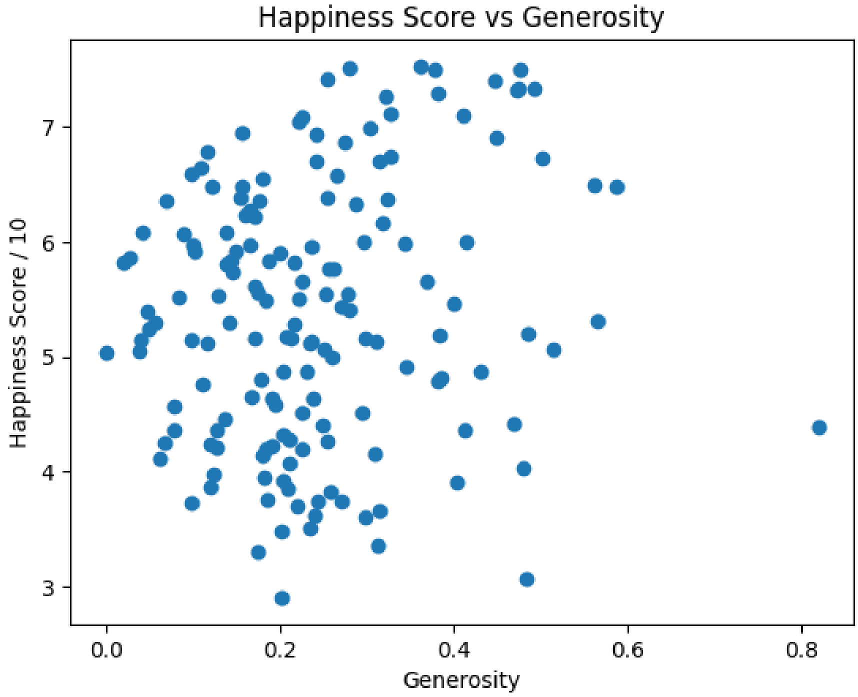
We can see that **the countries in the same region tend to have similar happiness and economy scores,** which is a very important insight for our future analysis.

Happiness Score vs Health (Life Expectancy)



This graph seems to further confirm that many variables are linearly linked with the Happiness Score as the better the health a country has, the happier its inhabitants will be.

Happiness Score vs Generosity



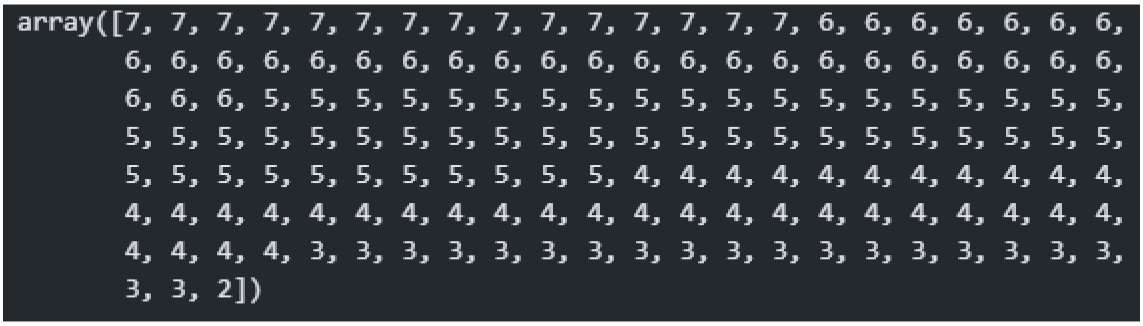
However, it is important to note that not all features behave that way. **Happiness does not seem to depend linearly on the Generosity for example** which was against our first intuition.

Step 6: Model Development

KNN Classification

Define the classes

First, we will retrieve the Happiness score (grade/10) of all countries and use these values as the classes for the KNN classifier.



```
array([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 2])
```

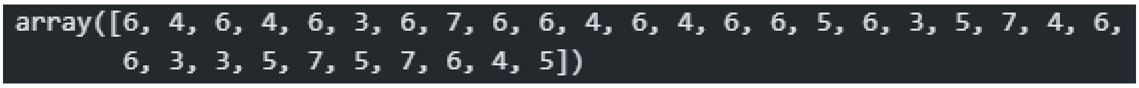
Split the data

Then, to test the KNN algorithms, we will split the data into training and testing sets using **20%** of the dataset for testing.

Fit the model and classify the test data

Using `sklearn.neighbors KNeighborsClassifier` we can easily fit the model to our dataset and see that the model is indeed classifying the countries of the test set into categories (happiness score/10).

Here are the classes found for our test data:



```
array([6, 4, 6, 4, 6, 3, 6, 7, 6, 6, 4, 6, 4, 6, 6, 5, 6, 3, 5, 7, 4, 6,
       6, 3, 3, 5, 7, 5, 7, 6, 4, 5])
```

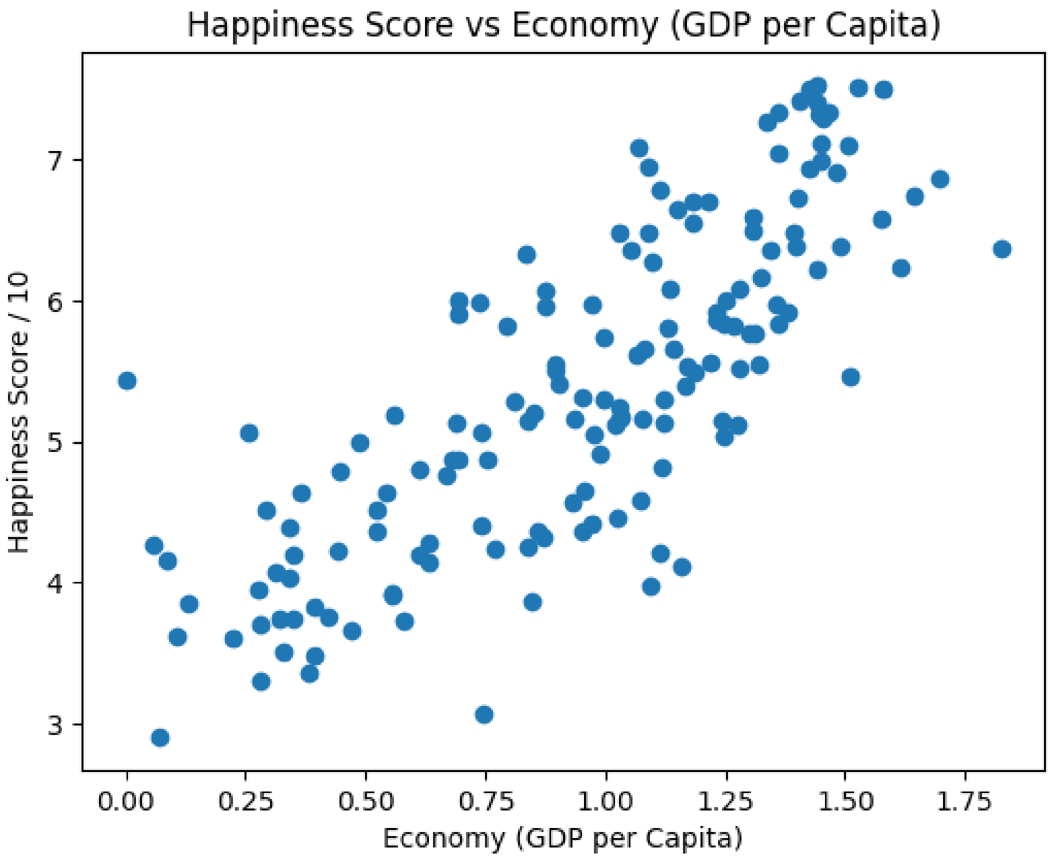
We will test the accuracy of the model in the Model Evaluation step and interpret the results in the Model Refinement step

Linear Regression

Selecting features

For this Linear Regression, we will choose "Economy (GDP per Capita)" as our independent variable and "Happiness Score" as our dependent variable.

Here we recall the graph of the Happiness Score in function of the Economy (GDP per Capita)



Split the data

Then, to test our algorithm, we will split our data into training and testing sets using **20%** of the dataset for testing.

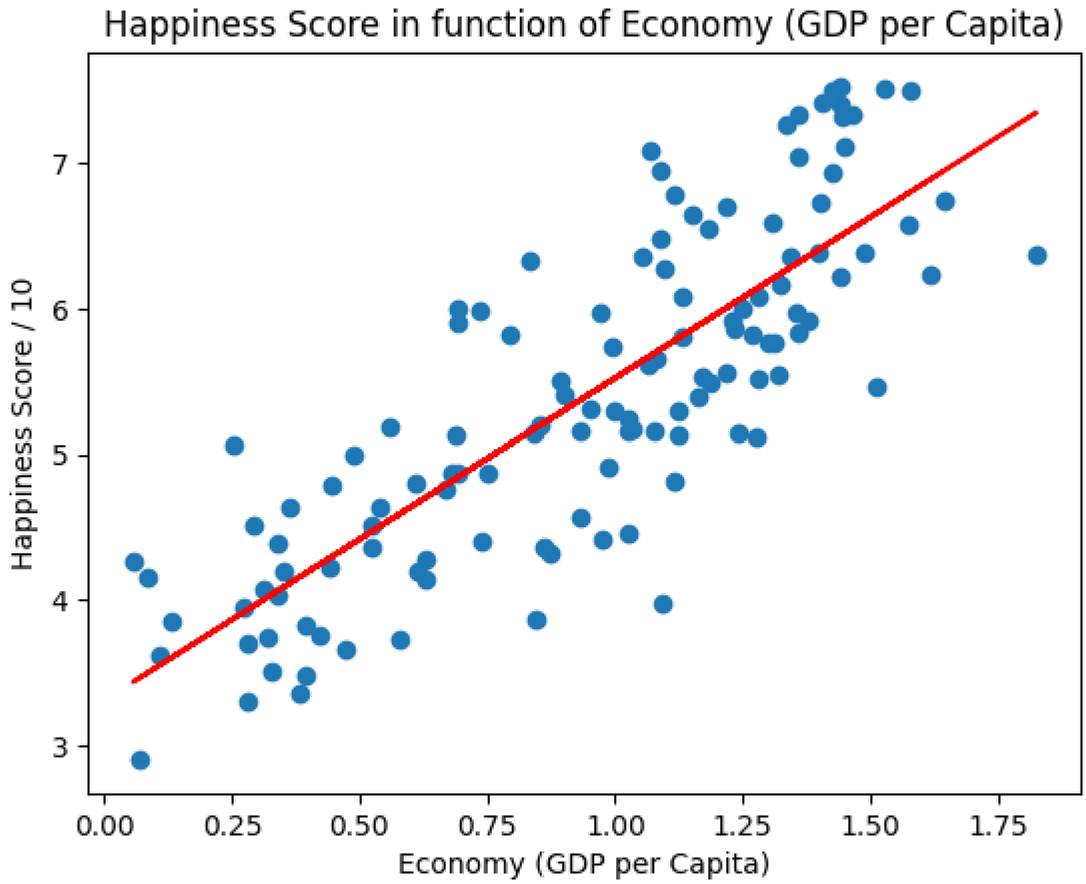
Fit and Obtain the coefficients of the model

Using `sklearn.linear_model LinearRegression`, we can easily implement our Linear Regression algorithm and fit our data.

Then, we obtain the coefficients of the model and are able to plot the curve of equation

$$y = ax + b$$

Plot the curve



This model clearly fits the data really well. The Happiness score and the Economy (GDP per Capita) are very linearly dependent.

We will test the accuracy of the model in the Model Evaluation step and interpret the results in the Model Refinement step

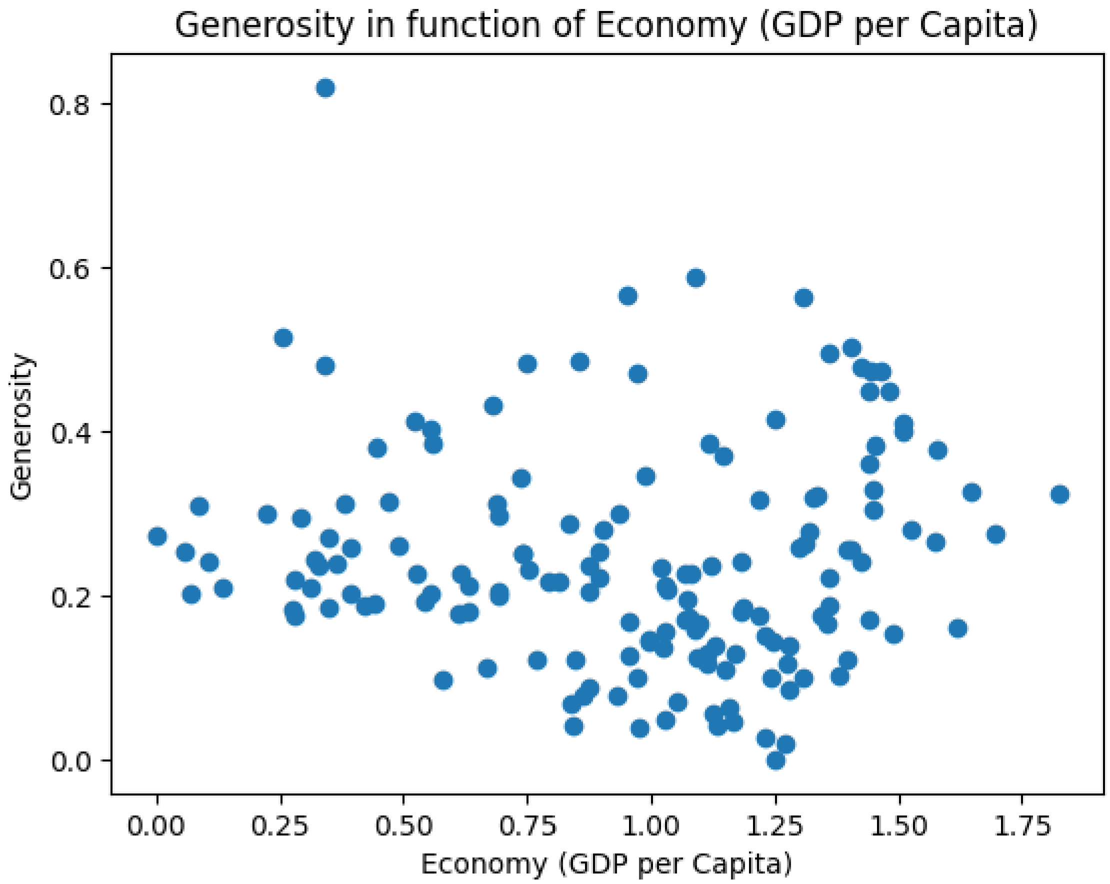
K-Means clustering

Selecting features

For this K-Means clustering, we will choose "Economy (GDP per Capita)" as our independent variable and " Generosity" as our dependent variable.

These variables are really interesting because, unlike what we might think, they are not linearly dependent. This Clustering should allow us to understand the moral values of the countries based on their economy.

Here we recall the graph of the Happiness Score in function of the Economy (GDP per Capita)



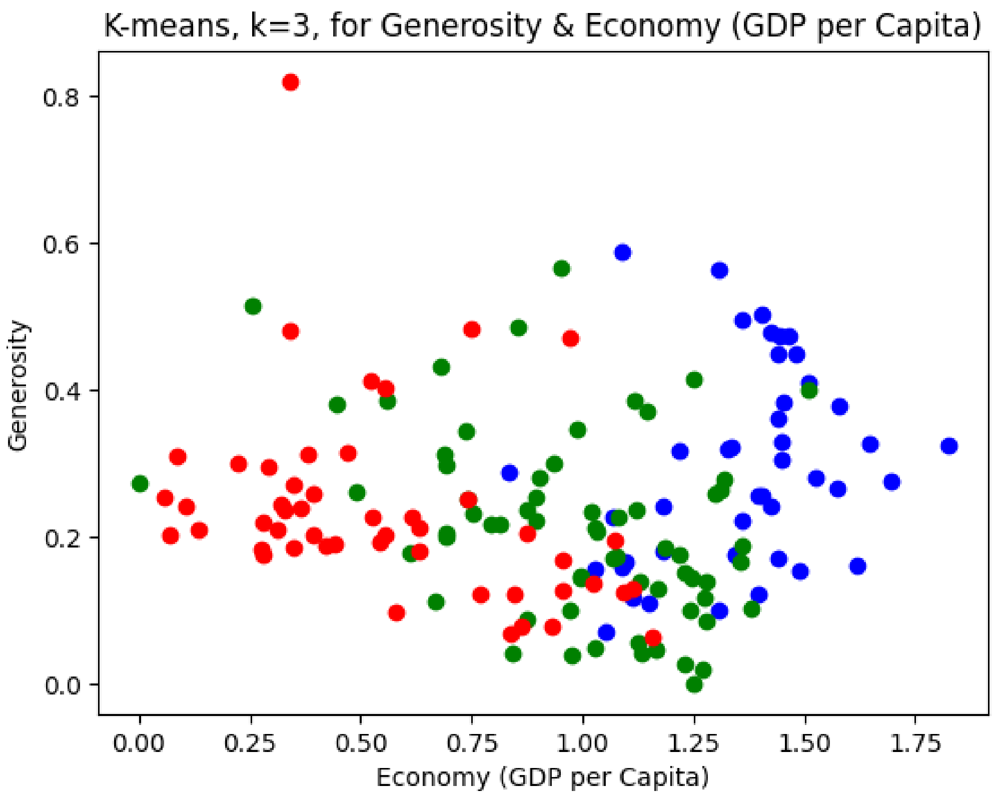
Fit the data

Using `sklearn.cluster KMeans` we can easily implement our K-Means clustering algorithm and fit our data.

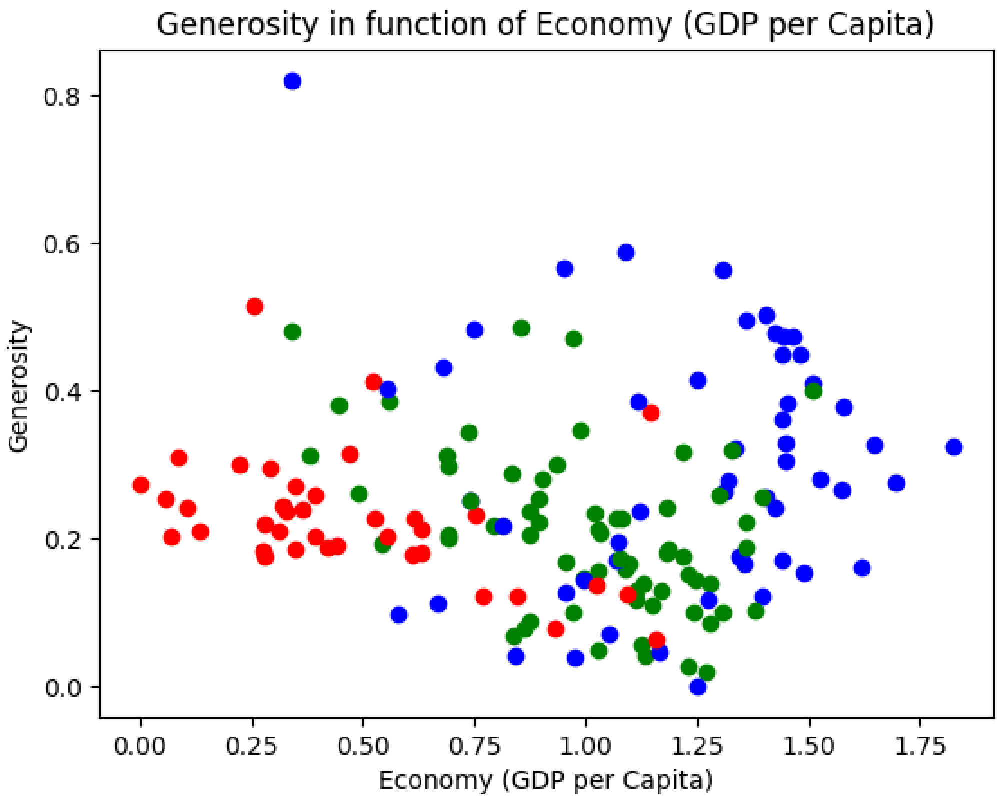
Plot the clusters

We can then plot the clusters using our 2 variables.

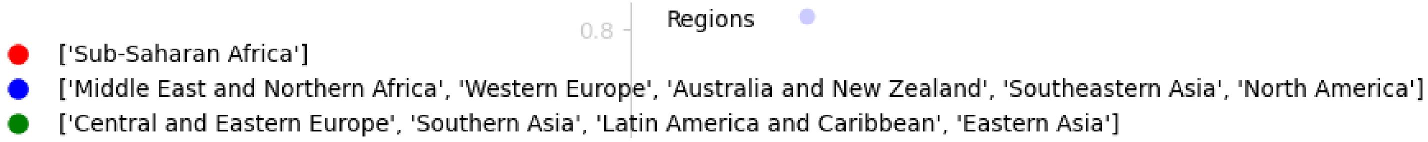
To really understand the meaning of the clusters, we also plotted the un-clustered graph but with different colors for regions (manual classifying). The interpretation of the results is in in the Model Refinement step



Comparison with the graph colored by regions (see graph legend):



Graph legend:



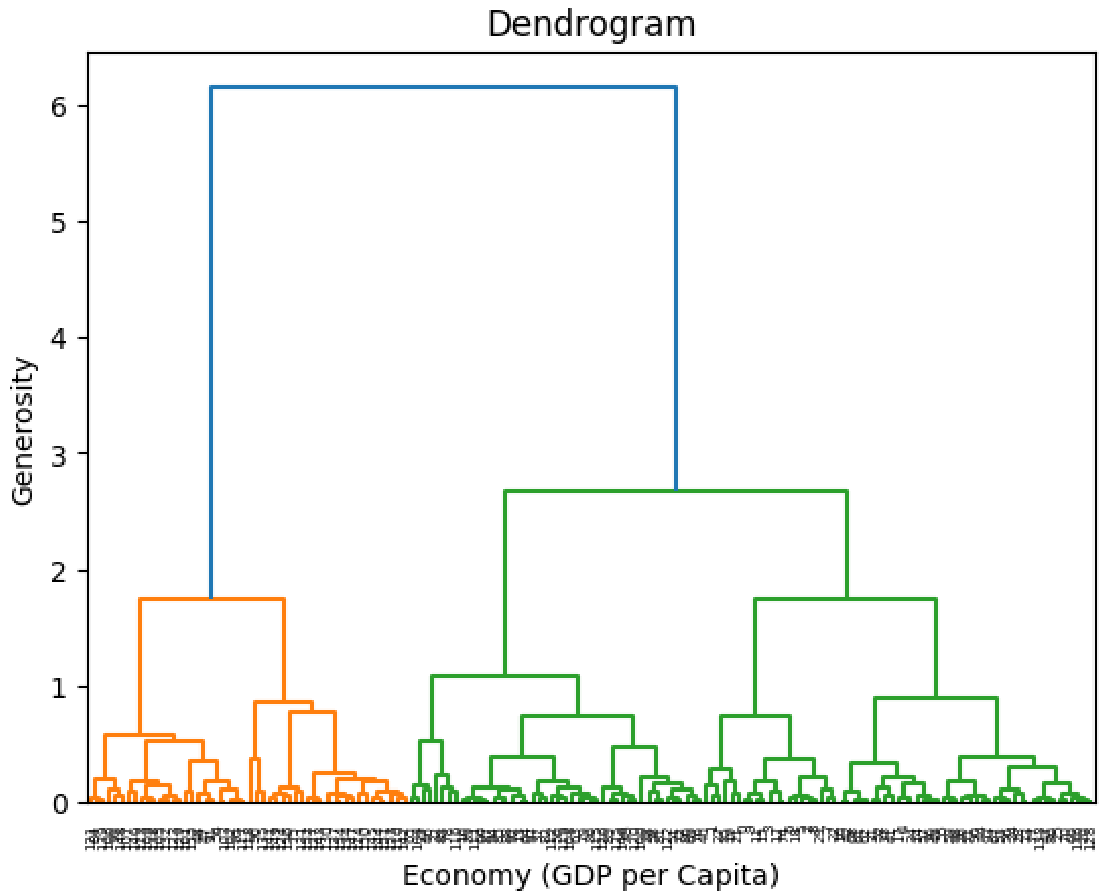
HCA

Selecting features

Let’s choose the same variables as the K-Means clustering. These variables are relevant for clustering and we will be able to compare the 2 models.

Dendrogram

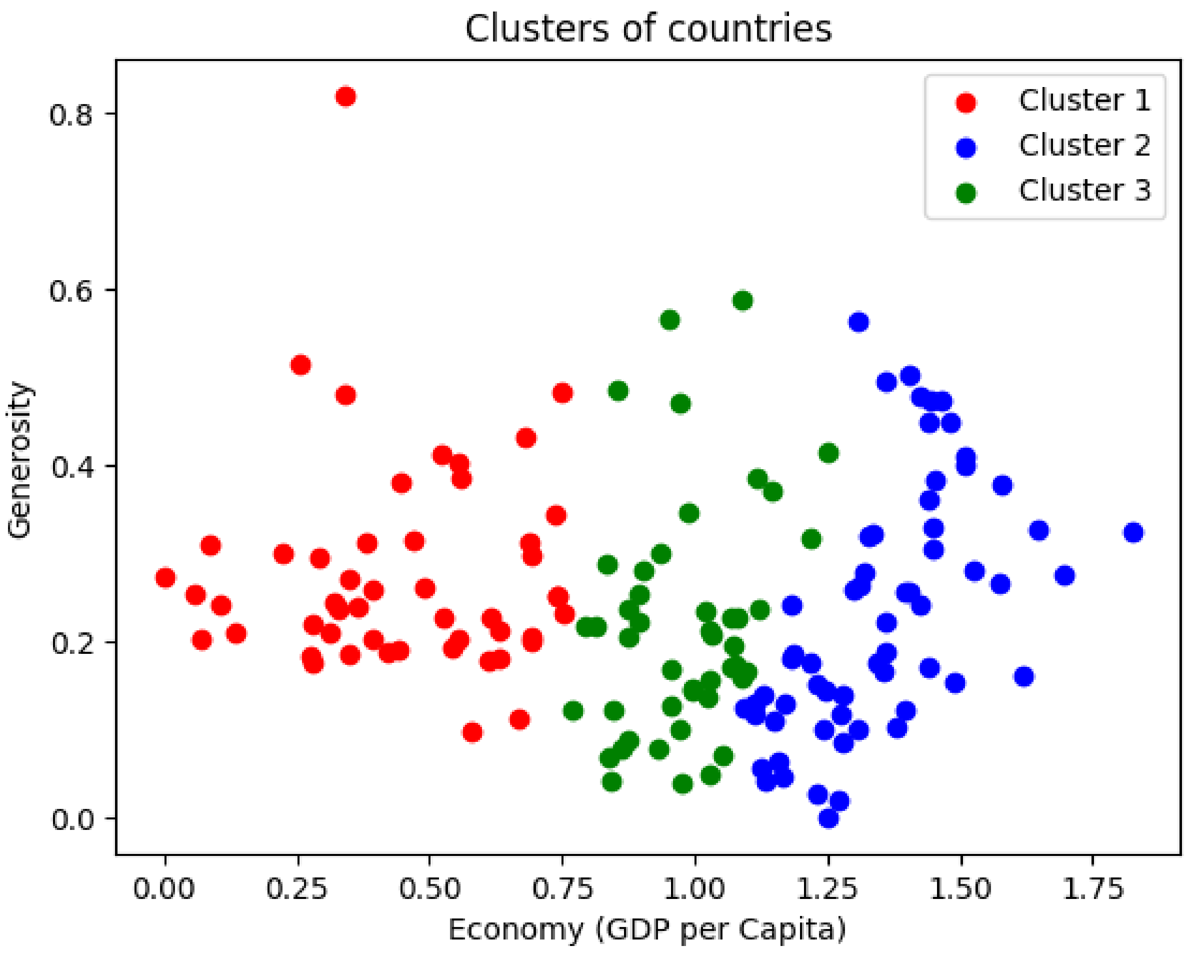
Using `scipy.cluster.hierarchy` we can easily create a Dendrogram representation of HCA.



We can see that 3 seems to be a very fitting number of clusters, similarly to K-Means.

Plot the clusters

Then, using `sklearn.cluster AgglomerativeClustering` we can implement our HCA clustering algorithm and fit our data.



The clusters are better defined since the model is not sensitive to noise and are less spherical. However, as a consequence, they also correspond less to the classes we found in the graph colored by region above, making these clusters a bit **meaningless to our data**. The interpretation of the results is in in the Model Refinement step.

PCA

PCA is a dimensionality reduction technique to **simplify datasets** with many features while retaining as much variability (information) as possible. It works by:

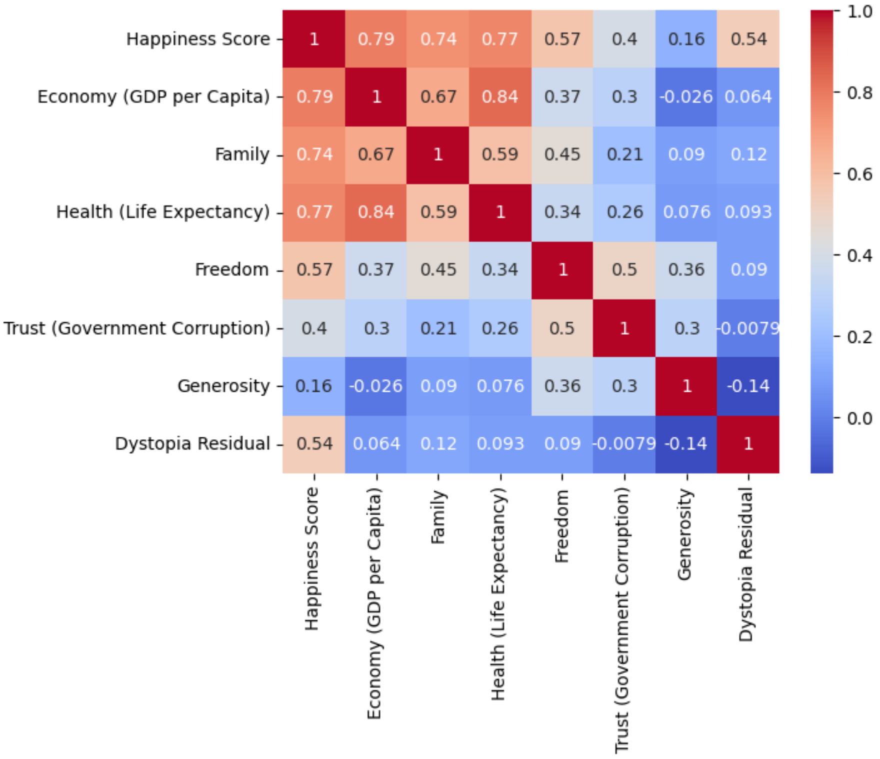
- Identifying the directions (principal components) along which the data varies the most (most “powerful” features).
- Transforming the original features into a new set of uncorrelated features (principal components) ordered by the amount of variance they capture.
- Reducing the dataset to **fewer dimensions** by selecting the top principal components

Correlation Matrix

The correlation matrix is the table showing the pairwise correlation coefficients between all variables in our dataset. Each cell in the matrix contains the correlation value (ranging from -1 to 1) for two variables:

- 1: Perfect positive correlation.
- 0: No correlation.
- -1: Perfect negative correlation.

We need it to assess relationships between the variables (prove that **there is redundancy** → PCA is relevant)



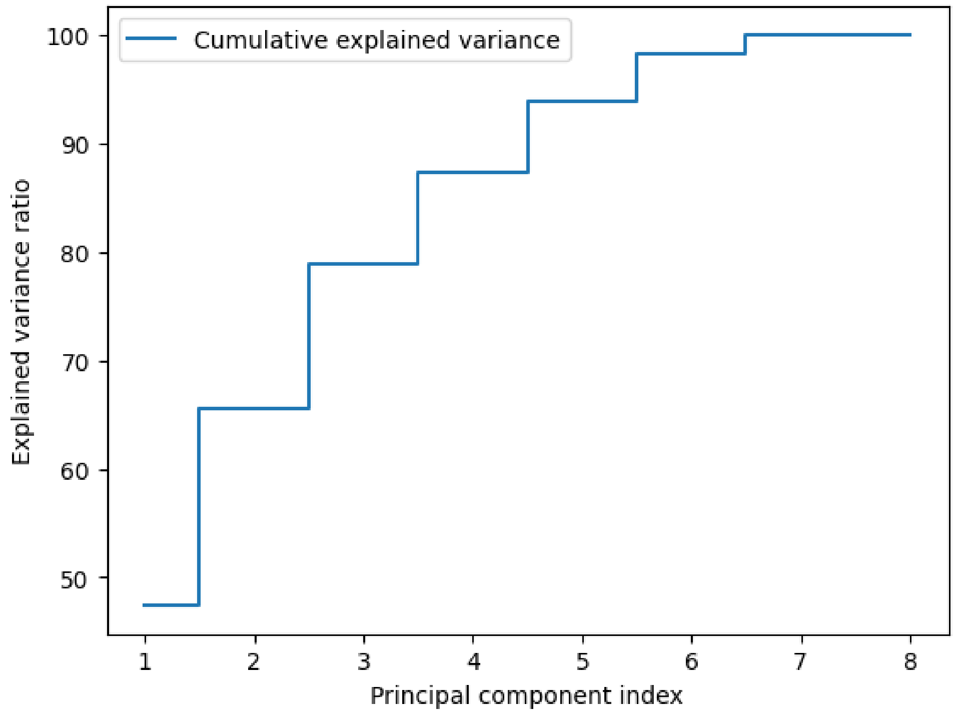
	Happiness Score	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
Happiness Score	1	0.79	0.74	0.77	0.57	0.4	0.16	0.54
Economy (GDP per Capita)	0.79	1	0.67	0.84	0.37	0.3	-0.026	0.064
Family	0.74	0.67	1	0.59	0.45	0.21	0.09	0.12
Health (Life Expectancy)	0.77	0.84	0.59	1	0.34	0.26	0.076	0.093
Freedom	0.57	0.37	0.45	0.34	1	0.5	0.36	0.09
Trust (Government Corruption)	0.4	0.3	0.21	0.26	0.5	1	0.3	-0.0079
Generosity	0.16	-0.026	0.09	0.076	0.36	0.3	1	-0.14
Dystopia Residual	0.54	0.064	0.12	0.093	0.09	-0.0079	-0.14	1

We consider strong correlations values above 0.5 or below -0.5 (the red on the graph)

We can See That there is strong correlation between multiple variables, therefore The PCA is relevant to reduce the dimensionality, and the computation power needed for a sensible analysis

Determine the number of components (new features) we will use

By diagonalizing the correlation matrix, we can deduce how many components are needed.

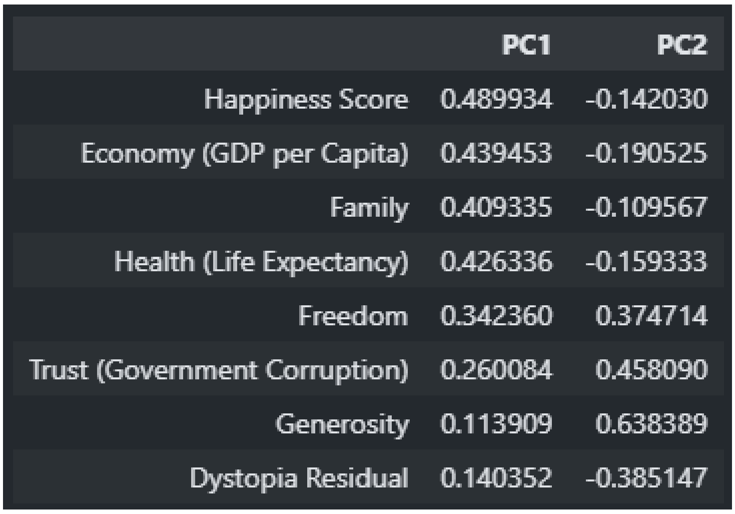


We can see that keeping 2 principal components for the analysis retains about 70% of the information of the original data<br>

This is a good compromise between dimensionality reduction (and computation power) and information loss

Features / Component Correlation

We verify that our features are represented well by the components

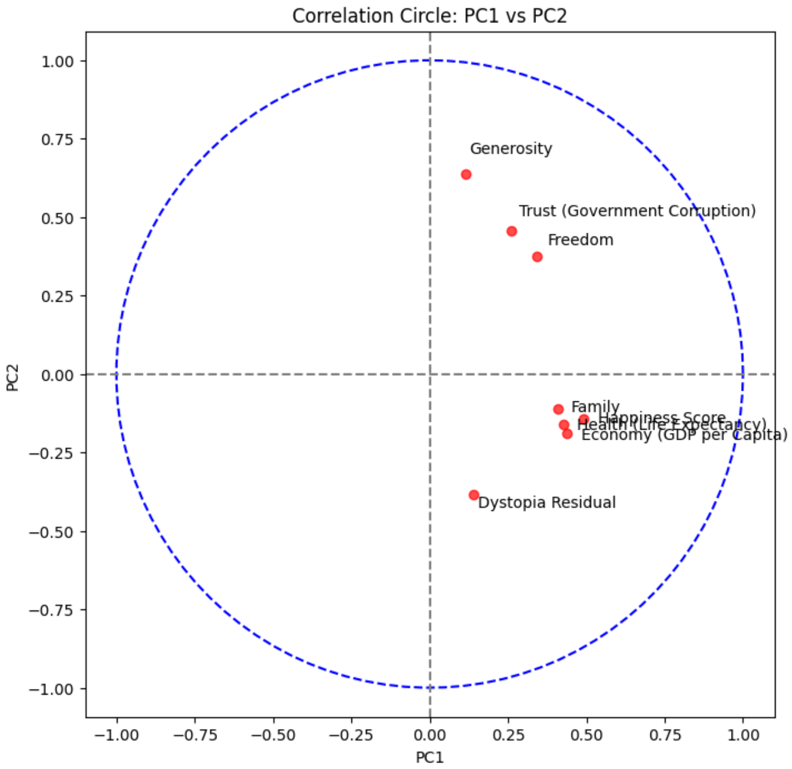


	PC1	PC2
Happiness Score	0.489934	-0.142030
Economy (GDP per Capita)	0.439453	-0.190525
Family	0.409335	-0.109567
Health (Life Expectancy)	0.426336	-0.159333
Freedom	0.342360	0.374714
Trust (Government Corruption)	0.260084	0.458090
Generosity	0.113909	0.638389
Dystopia Residual	0.140352	-0.385147

We can see that the components are strongly correlated with the original features

Correlation circle

This is a plot of our old features based on their correlation with the new Components that we defined.



From the correlation circle, we can understand what each axis represents:

Axis 1 : Material and Economic Factors

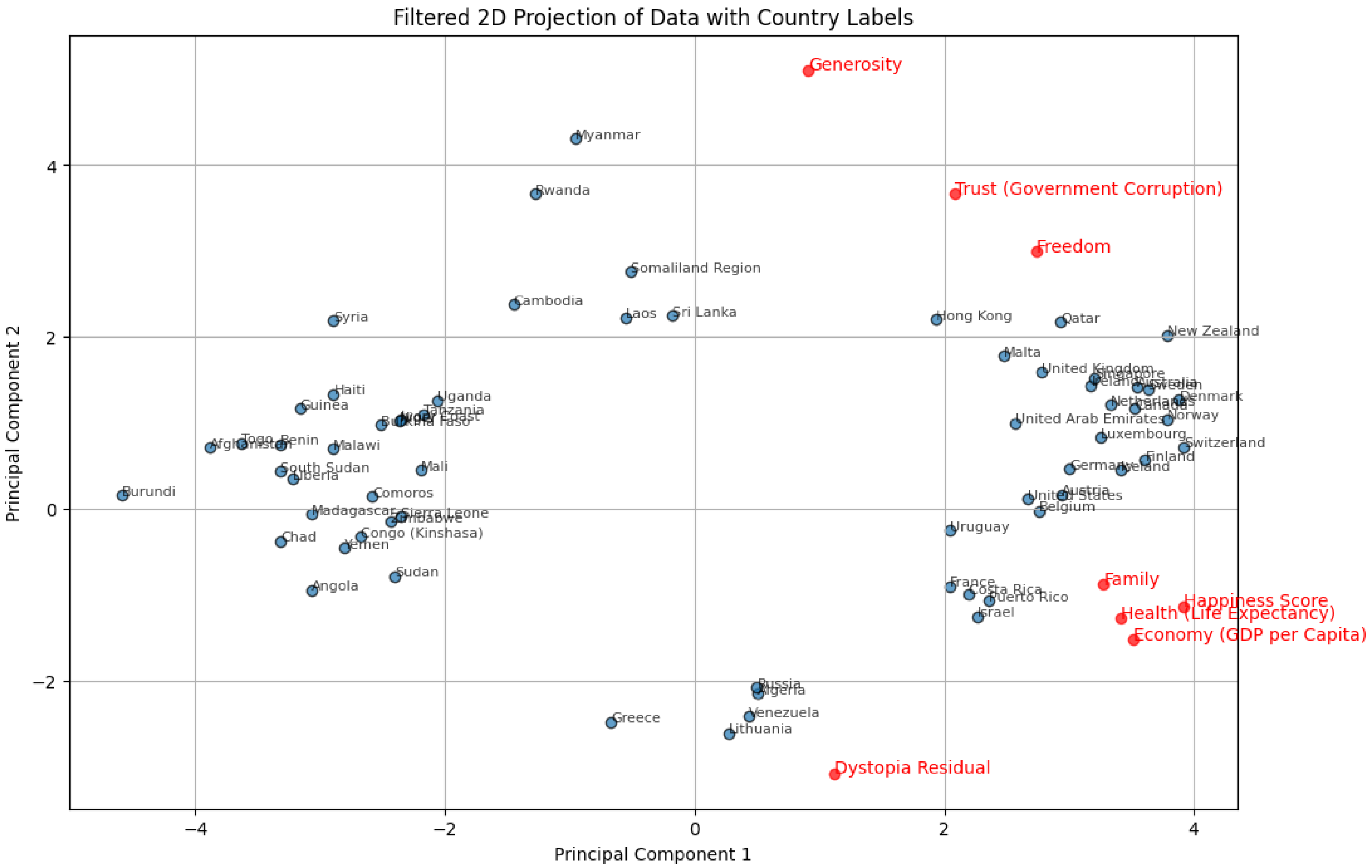
- Family, health, economy

Axis 2 : Social and Moral Factors

- Generosity, Trust, freedom,

Final plot

Now we will plot again the countries and the features but using the 2 principal components as the x and y axis



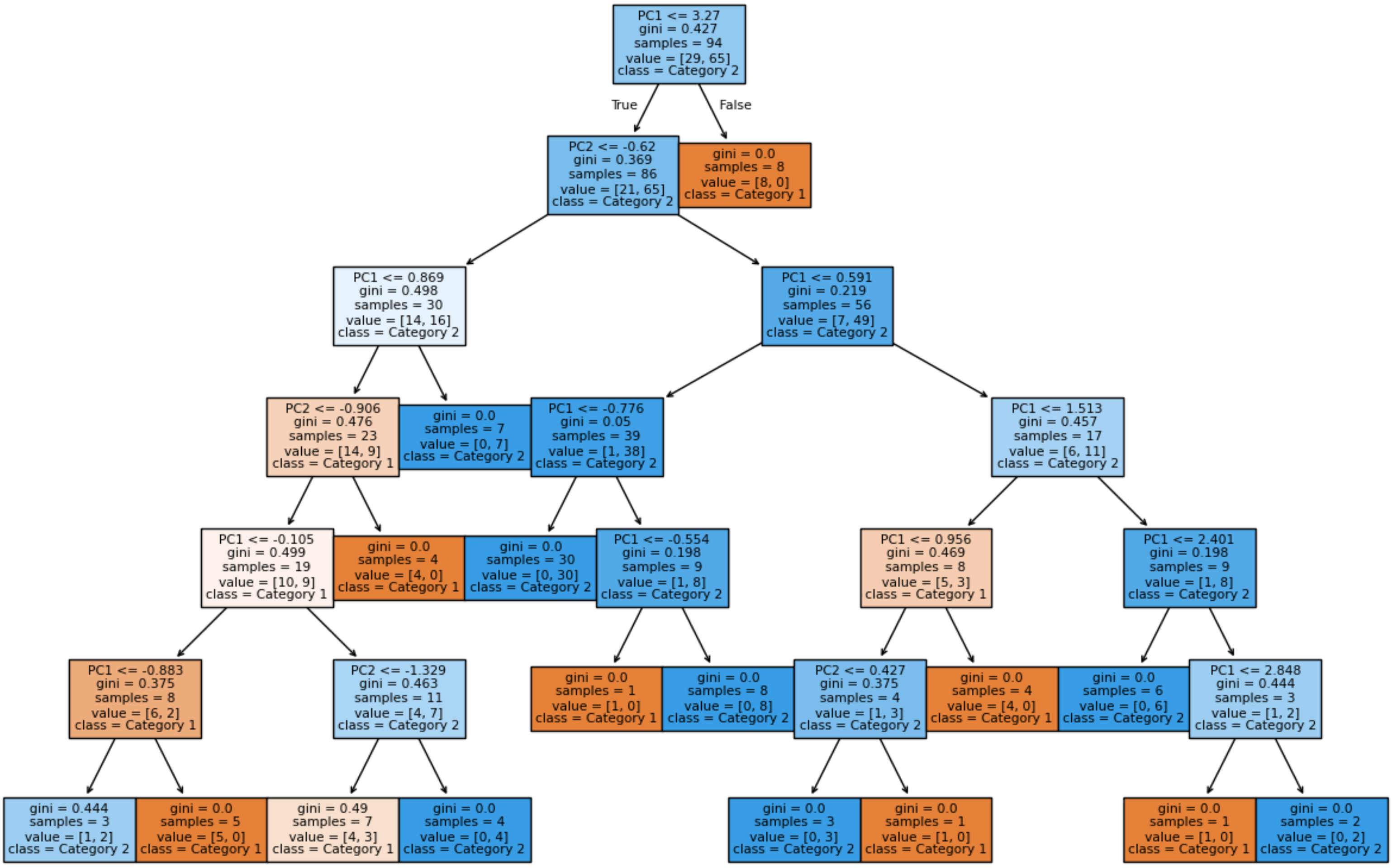
With a high enough treshold, we can clearly see two groups of counties. The PC1 component is linked to the countries wealth & PC2 component is linked to the Social/Moral Factors

Decision Tree Classifier on the PCA’s data

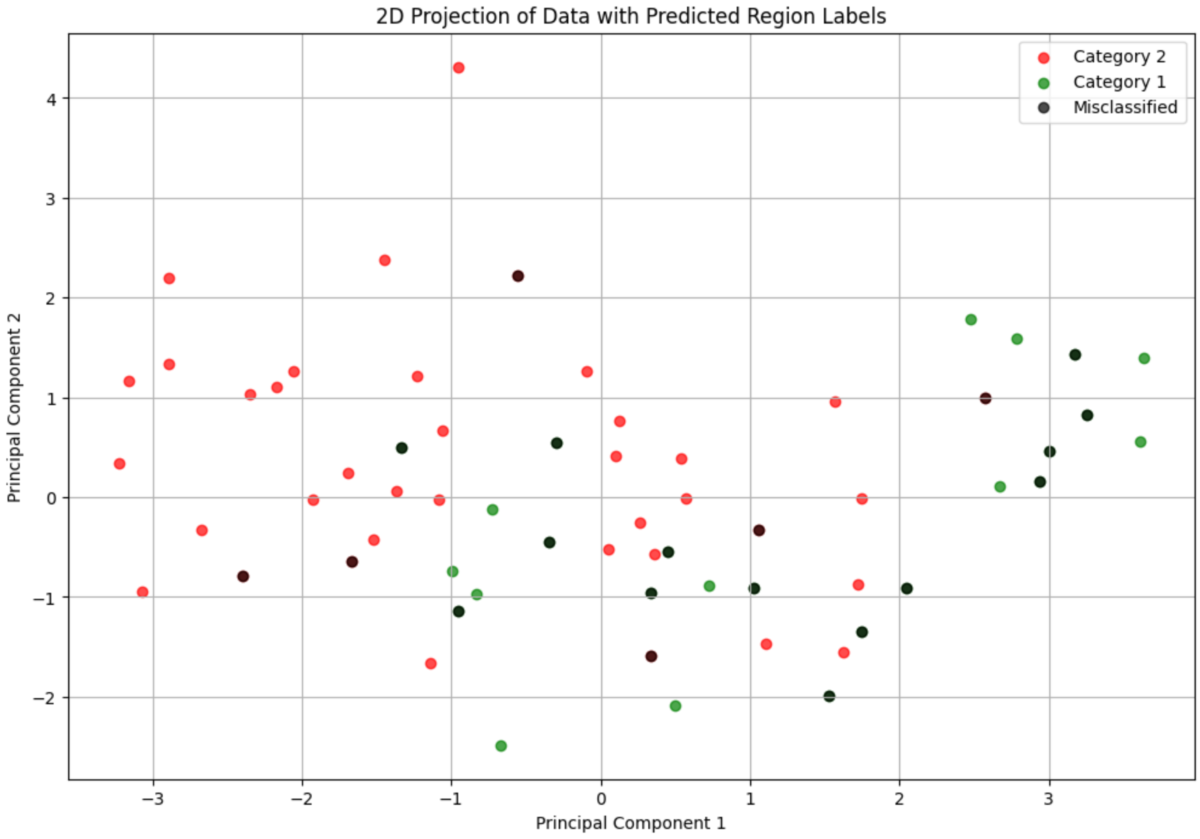
Based on the PCA results, we can try to group our results, in this case, we will try to find out, if we can separate the occidental countries, from the rest of the world

First, we have to find the best depth for the classifier, then we can create the tree

- At each step, the algorithm evaluates all features to determine the best one to split the data.
- The "best" feature is ch osen based on a criterion that measures the quality of the split (e.g., Gini impurity or Entropy in Information Gain).


PC1 <= 3.27 gini = 0.427 samples = 94 value = [29, 65] class = Category 2
True
False
PC2 <= -0.62 gini = 0.369 samples = 86 value = [21, 65] class = Category 2
gini = 0.0 samples = 8 value = [8, 0] class = Category 1
PC1 <= 0.869 gini = 0.498 samples = 30 value = [14, 16] class = Category 2
PC1 <= 0.591 gini = 0.219 samples = 56 value = [7, 49] class = Category 2
PC2 <= -0.906 gini = 0.476 samples = 23 value = [14, 9] class = Category 1
gini = 0.0 samples = 7 value = [0, 7] class = Category 2
PC1 <= -0.776 gini = 0.05 samples = 39 value = [1, 38] class = Category 2
PC1 <= 1.513 gini = 0.457 samples = 17 value = [6, 11] class = Category 2
PC1 <= -0.105 gini = 0.499 samples = 19 value = [10, 9] class = Category 1
gini = 0.0 samples = 4 value = [4, 0] class = Category 1
gini = 0.0 samples = 30 value = [0, 30] class = Category 2
PC1 <= -0.554 gini = 0.198 samples = 9 value = [1, 8] class = Category 2
PC1 <= 0.956 gini = 0.469 samples = 8 value = [5, 3] class = Category 1
PC1 <= 2.401 gini = 0.198 samples = 9 value = [1, 8] class = Category 2
PC1 <= -0.883 gini = 0.375 samples = 8 value = [6, 2] class = Category 1
PC2 <= -1.329 gini = 0.463 samples = 11 value = [4, 7] class = Category 2
gini = 0.0 samples = 1 value = [1, 0] class = Category 1
gini = 0.0 samples = 8 value = [0, 8] class = Category 2
PC2 <= 0.427 gini = 0.375 samples = 4 value = [1, 3] class = Category 2
gini = 0.0 samples = 4 value = [4, 0] class = Category 1
gini = 0.0 samples = 6 value = [0, 6] class = Category 2
PC1 <= 2.848 gini = 0.444 samples = 3 value = [1, 2] class = Category 2
gini = 0.444 samples = 3 value = [1, 2] class = Category 2
gini = 0.0 samples = 5 value = [5, 0] class = Category 1
gini = 0.49 samples = 7 value = [4, 3] class = Category 1
gini = 0.0 samples = 4 value = [0, 4] class = Category 2
gini = 0.0 samples = 3 value = [0, 3] class = Category 2
gini = 0.0 samples = 1 value = [1, 0] class = Category 1
gini = 0.0 samples = 1 value = [1, 0] class = Category 1
gini = 0.0 samples = 2 value = [0, 2] class = Category 2

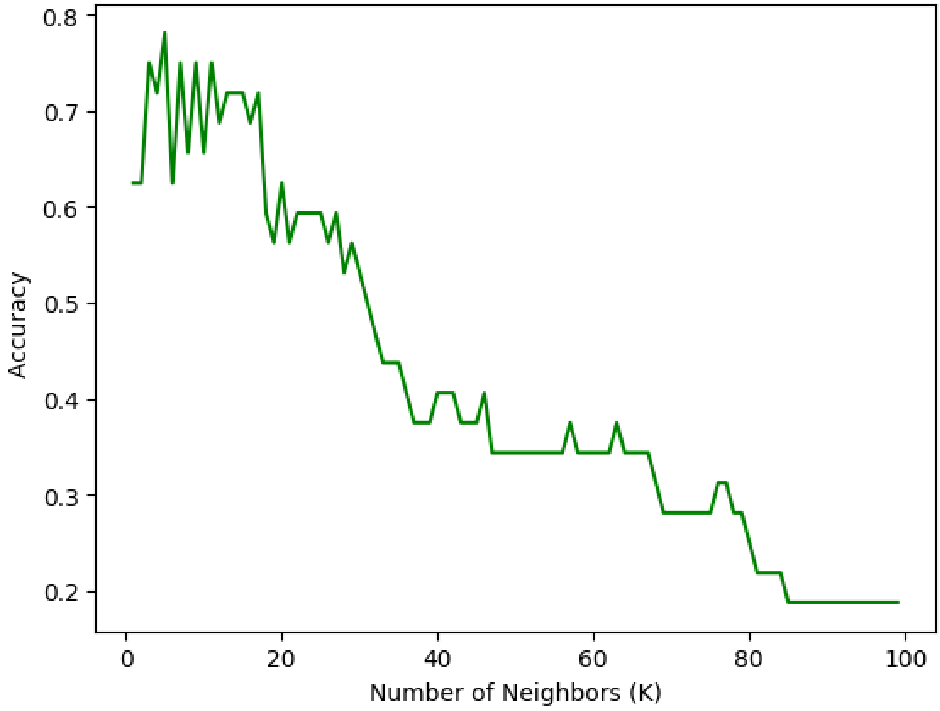
Finally, we can plot again the PCA's data using these newly defined classes.



On the final graph, we can see that the algorithm,managed to separate the occidental countries from the rest

KNN Classification

Accuracy Evaluation



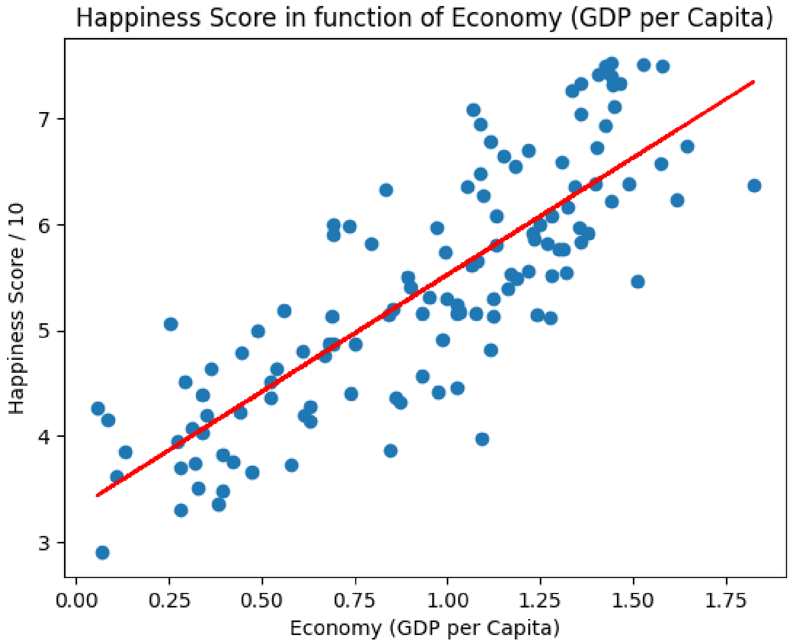
This graph shows us the accuracy of the model for different numbers of neighbors.

The best Test set accuracy was 0.78125 with k = 5

We can conclude that the best number of neighbors is 5 as it gives the highest accuracy for the test set while keeping a high accuracy for the training set.

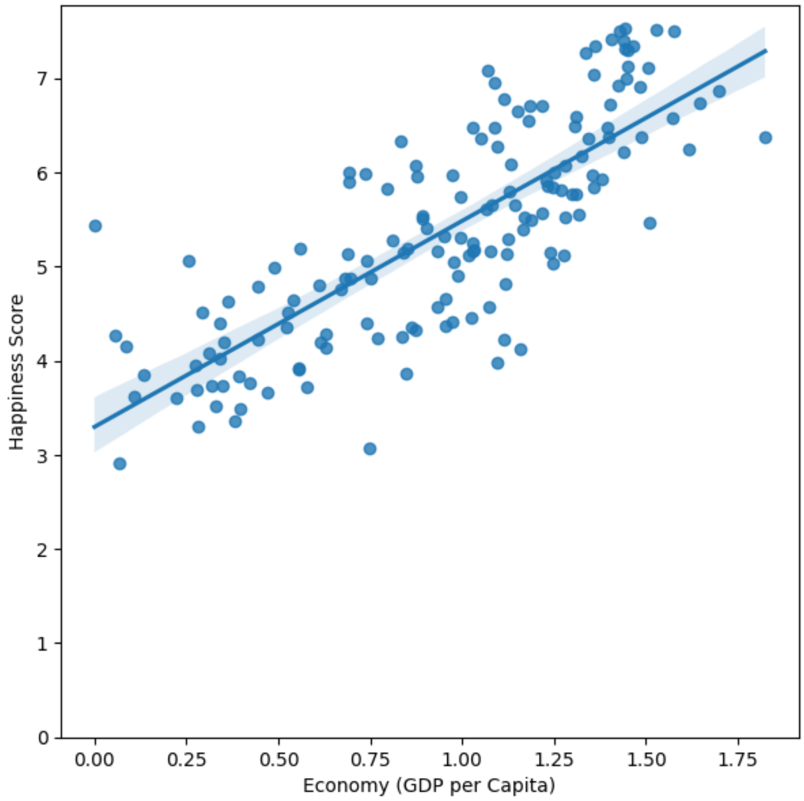
Linear Regression

Recall of the results:



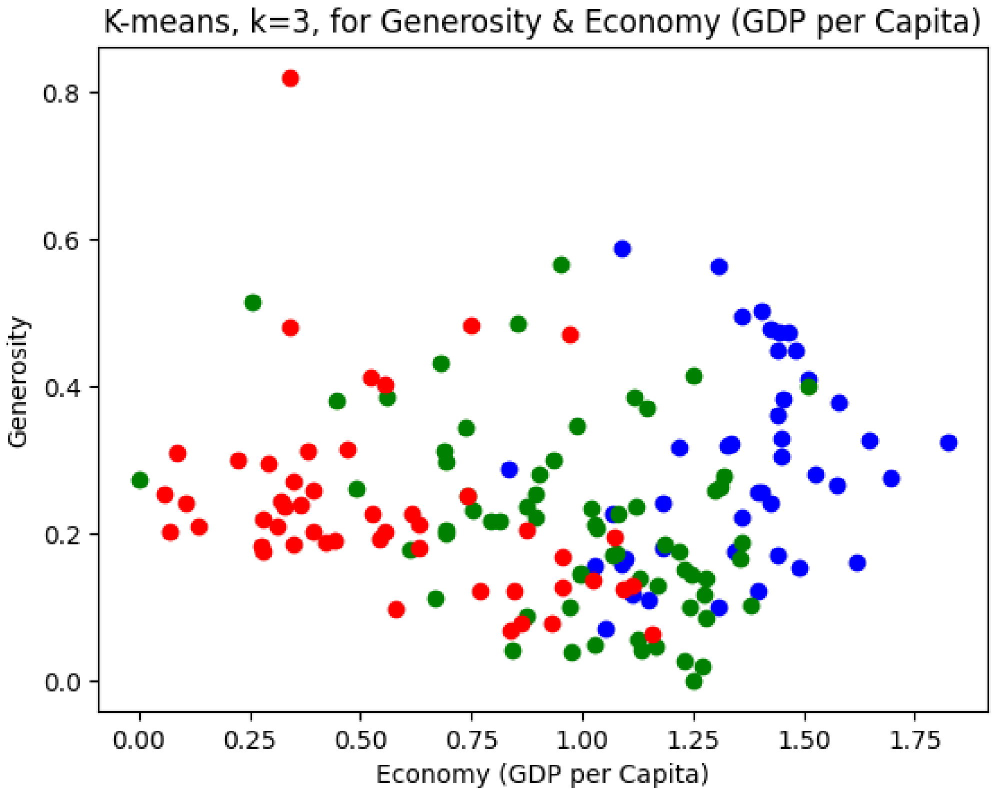
Regression Plot

This plot will show a combination of a scattered data points (a <b>scatterplot</b>), as well as the fitted <b>linear regression</b> line going through the data. This will give us a reasonable estimate of the relationship between the two variables, the strength of the correlation, as well as the direction (positive or negative correlation)

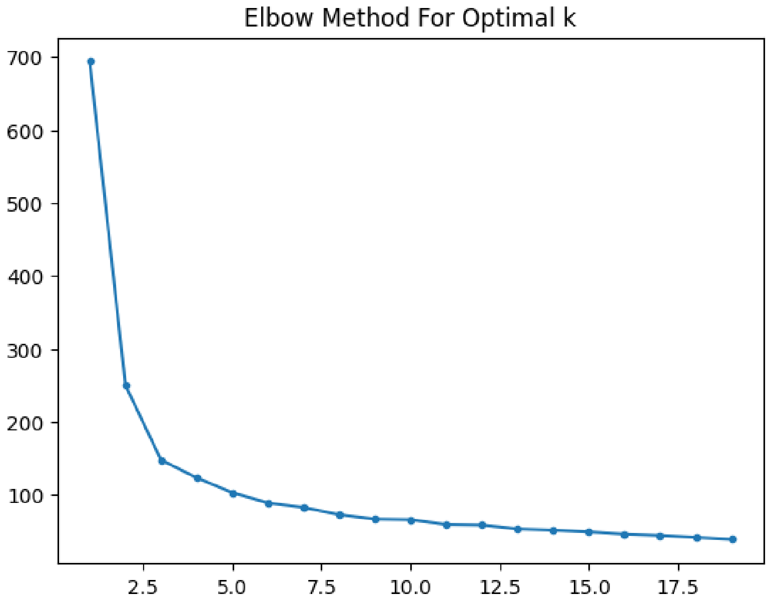


K-Means clustering

Recall of the results obtained:



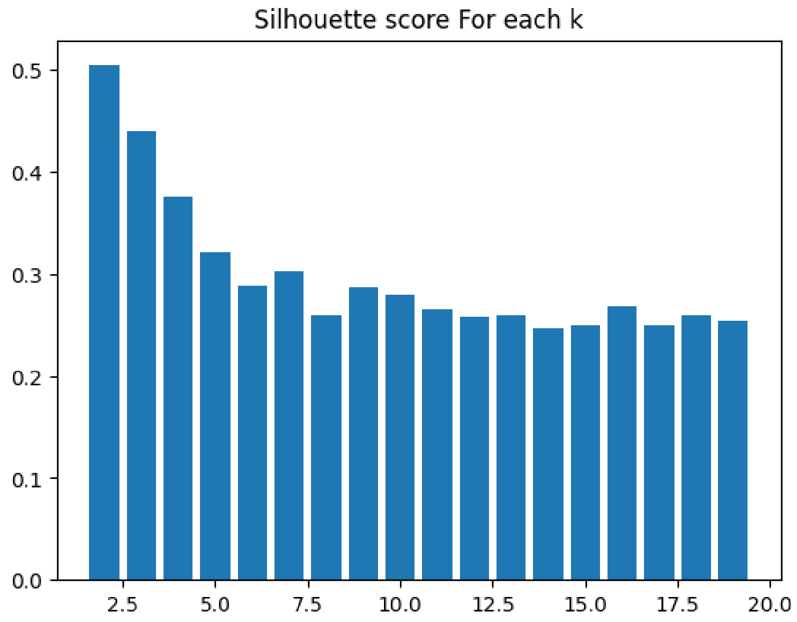
Elbow method



We see a sudden drop in inertia at k=3, then the curve becomes linear.

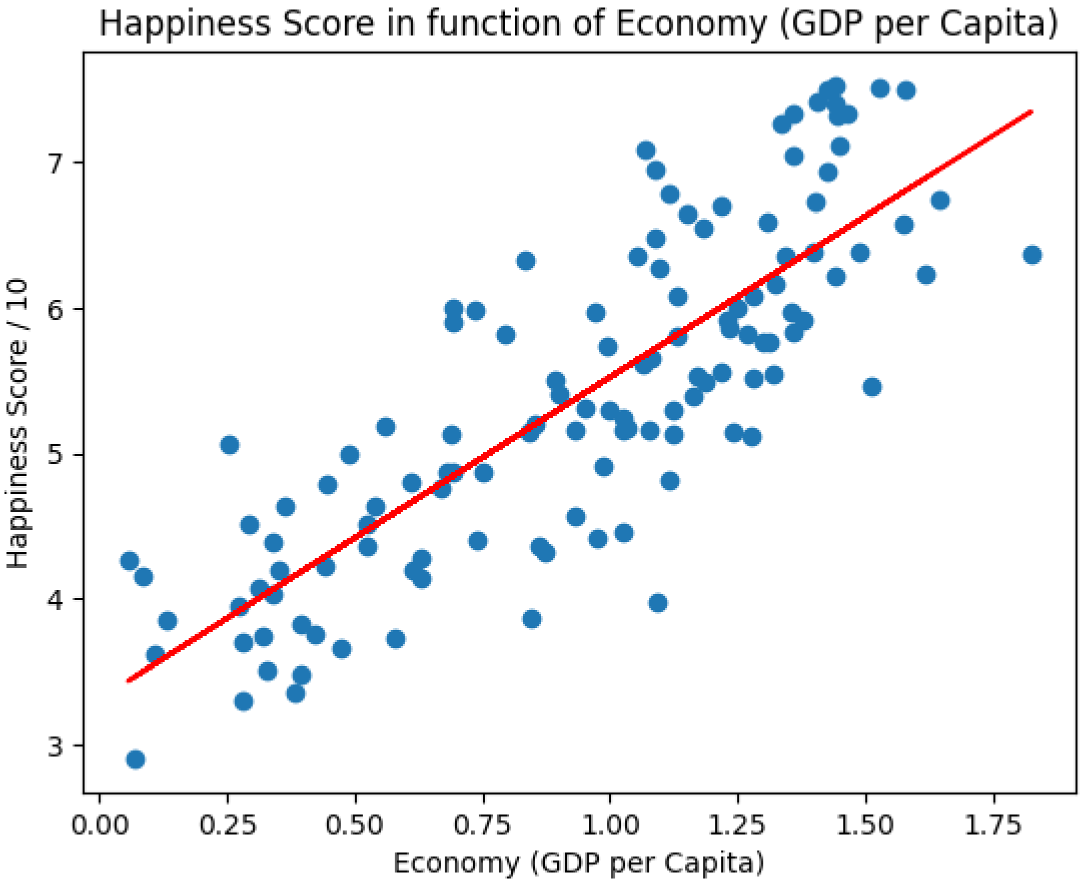
Therefore, we can assume that the optimal number of clusters is 3 as we have previously chosen.

Silhouette score



The silhouette score for k=3 is 0.44, which is not the highest, but is more than acceptable.

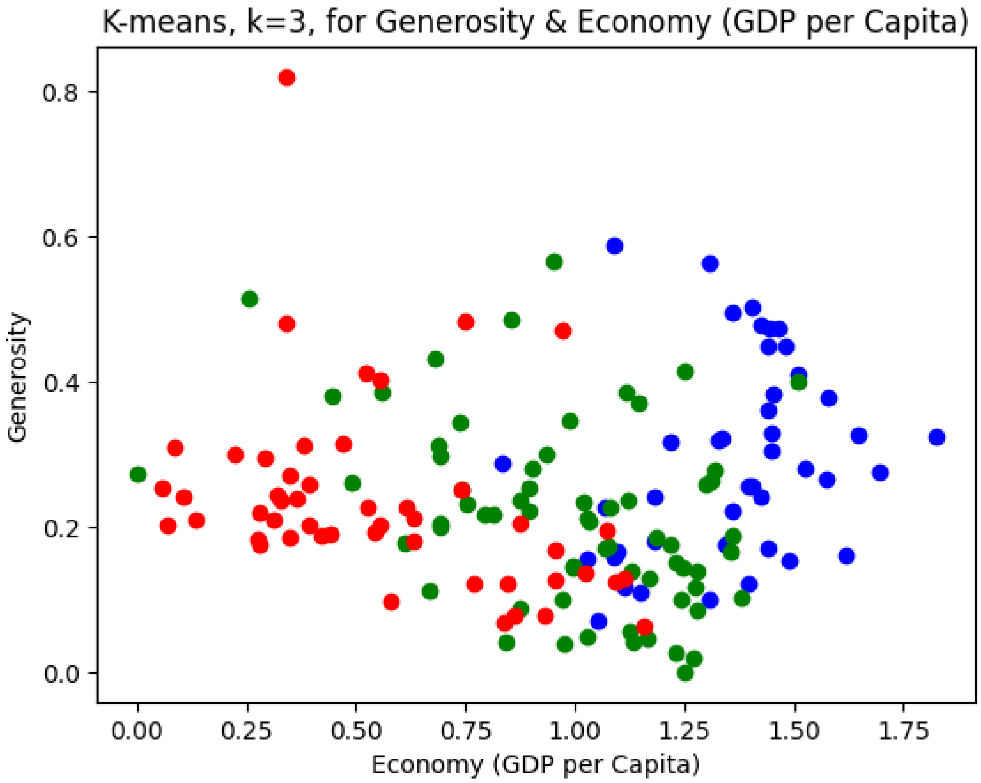
Linear Regression



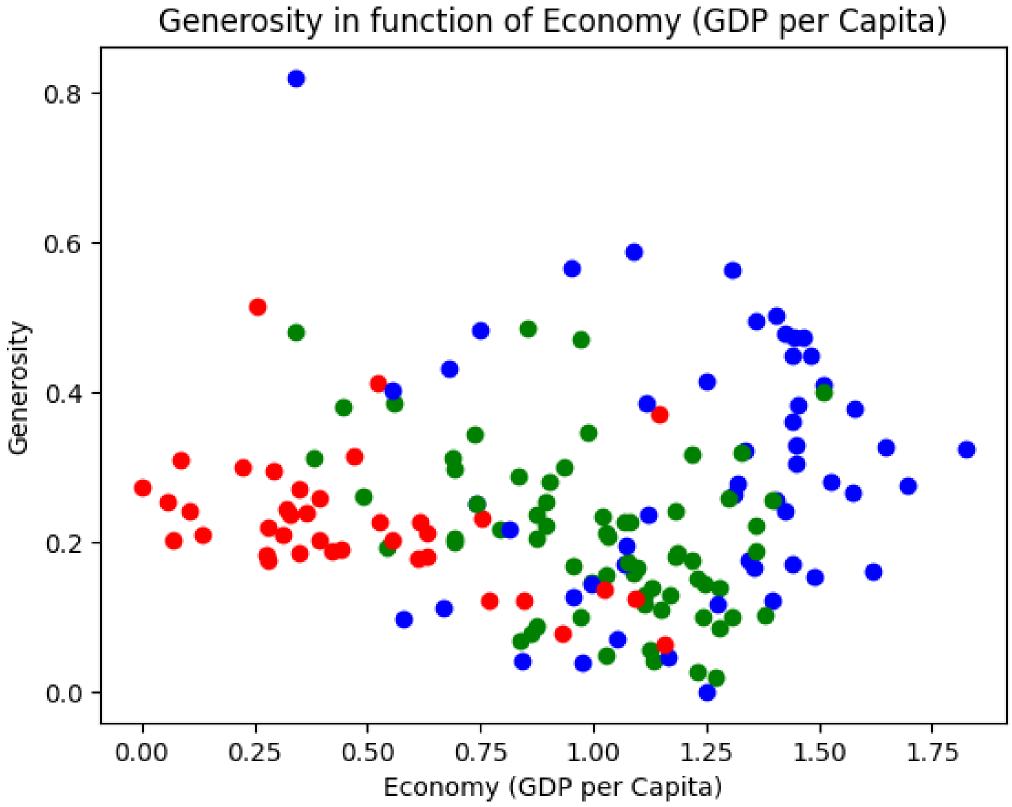
This model further confirms our first intuition that **the more money a country produces, the happier its inhabitants will be**.

And since many other features are linearly dependent with the Happiness score, such as “Health (Life Expectancy)”, we can safely deduct the most important variable (those that have the higher impact on the happiness score)

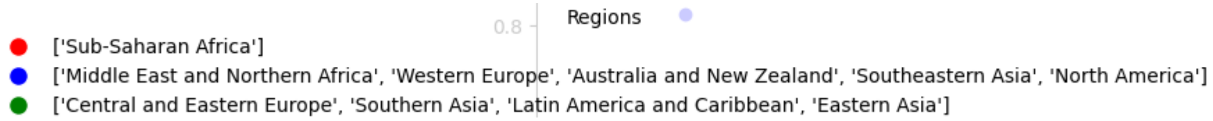
K-Means clustering



Graph colored by regions:



Graph legend:



🔴 ['Sub-Saharan Africa']

🔵 ['Middle East and Northern Africa', 'Western Europe', 'Australia and New Zealand', 'Southeastern Asia', 'North America']

🟢 ['Central and Eastern Europe', 'Southern Asia', 'Latin America and Caribbean', 'Eastern Asia']

With this graph, we can **explain better the clusters** generated by the K-Means algorithm:

Category 1 (red): "Emerging Regions"

- Includes: Sub-Saharan Africa
- Rationale: Sub-Saharan Africa is often characterized by developing economies and unique socio-political challenges, making it distinct from the other regions.

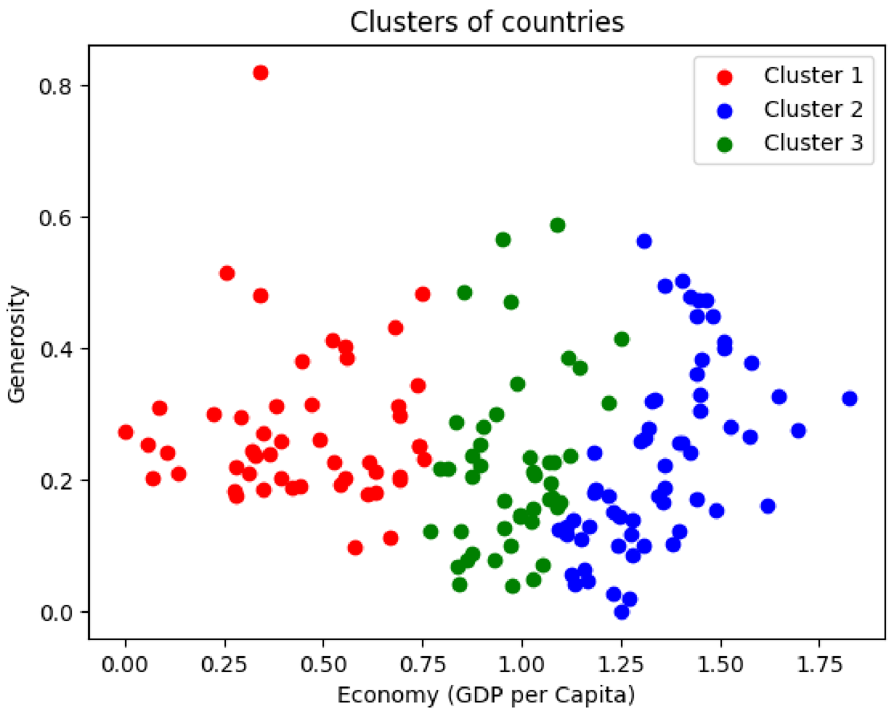
Category 2 (blue): "Developed and Diverse Economies"

- Includes: Middle East and Northern Africa, Western Europe, Australia and New Zealand, Southeastern Asia, North America
- Rationale: These regions represent a mix of developed economies and regions with significant economic diversity. They encompass high-income countries (e.g., Western Europe, North America, Australia and New Zealand) and rapidly developing areas with economic integration (e.g., Southeastern Asia, parts of the MiddleEast).

Category 3 (green): "Transitional and Emerging Economies"

- Includes: Central and Eastern Europe, Southern Asia, Latin America and Caribbean, Eastern Asia
- Rationale: These regions are characterized by a mix of developing and transitional economies, with many countries undergoing rapid industrialization and socio-economic changes.

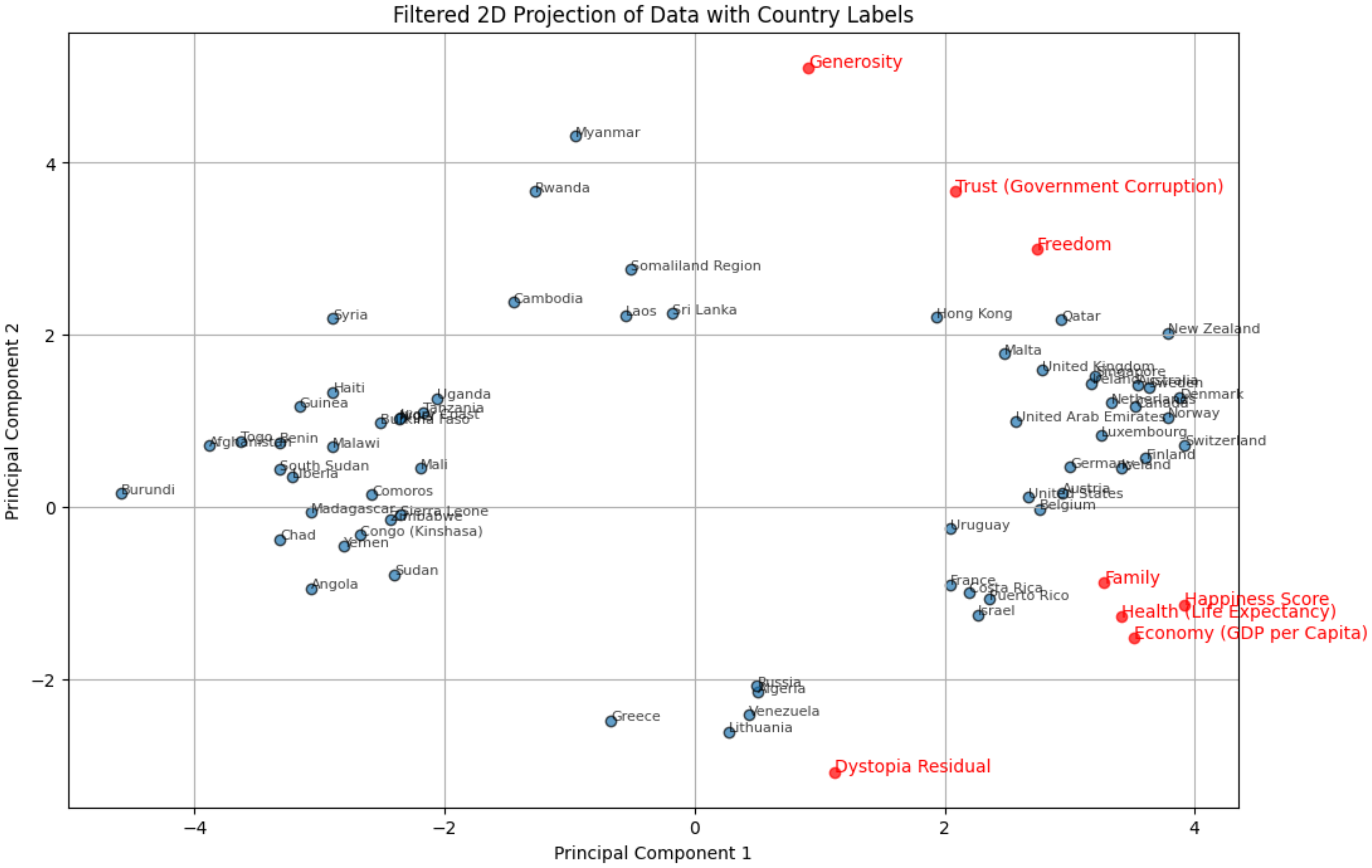
Comparison with HCA



The model is not sensitive enough to sound to accurately cluster the countries/regions which is a **big limitation** in our case as the clusters are clear but meaningless regarding our data.

PCA

Recall of the result:



PCA allowed us to:

- Identifying the directions (principal components) along which the data varies the most.
- Transforming the original features into a new set of uncorrelated features (principal components) ordered by the amount of variance they capture.
- Reducing the dataset to fewer dimensions by selecting the top principal components.

Decision Tree Classifier on the PCA's data

The Decision Tree Classifier provides a very insightful view

Advantages:

- Interpretable: The tree structure is intuitive and easy to understand.
- Non-linear: Can capture non-linear relationships between features and the target.
- Handles Mixed Data Types: Works with both numerical and categorical data.
- Requires Minimal Preprocessing: No need for feature scaling or centering.

However, when modifying parameters or updating the PCA, we realized the Decision Tree Classifier has many drawbacks:

- Overfitting: Decision trees can overfit the training data, capturing noise instead of the underlying pattern.
- Bias to Imbalanced Data: May perform poorly with imbalanced datasets unless balanced criteria are used.
- Instability: Small changes in data can lead to significantly different trees