EPA112A - Programming for Data Science - Group 31

GDP Predictive Modeling Based: a deep-dive into Social, Economic, and Environmental Indicators

Authors:

- *Georges Puttaert - 4686160*
- *Thijs Roolvink - 4961382*
- *Gijs de Werd - 4717775*

Introduction

The Gross Domestic Product (GDP) of a country is a fundamental measure of its economic well-being and prosperity. Understanding the factors that influence GDP is of importance to governments, policymakers, and economists. In our research project, we analyze some of these critical drivers by comparing the economies and their GDP over the years of the Netherlands, Germany, Greece, and Ireland.

These four countries have been selected for their distinct economic sizes and profiles, making them ideal candidates for our analysis. The Netherlands and Germany represent two of the largest and most robust economies in Europe. At the same time, Greece and Ireland, while smaller in scale, offer insights into the economic characteristics of Southern and Western Europe. By analyzing a diverse set of economies, we aim to draw valuable insights into the factors that shape GDP.

Our analyses focus on three categories of indicators: Environmental, Social (economy), and Governance (ESG). Each category includes a set of indicators that may have a significant influence on a country's GDP. Our project will not only examine the correlation between these indicators and GDP but also employ machine-learning models for GDP prediction, considering these diverse indicators. Moreover, visualization techniques will be applied to illustrate these relationships and provide a better understanding of the complex interplay between social, economic, and environmental factors.

The research question of this report is as following:

To what extent do Environmental, Social, and Governance (ESG) indicators influence a country's GDP, and which machine learning model offers the most accurate GDP predictions based on these indicators?

Followed by three corresponding subquestions:

- How significantly do ESG indicators correlate with a country's GDP per capita?
- Among the machine learning models (Random Forest, Linear Regression, and MLP), which model gives the most accurate GDP predictions based on the combined set of indicators?
- Are there significant differences in prediction errors among these machine learning models when applied to each of the four selected countries (the Netherlands, Germany, Greece, and Ireland)?
- What is the relative impact of social, economic, and environmental indicators separately on a country's GDP?

Firstly, the indicators that are being used during this report. After that will be the importation of the useful libraries. In Chapter 1, the data is retrieved from the sources, and it is further preprocessed in a main DataFrame. This gives easy access to the corresponding data needed. Chapter 2 will be about the first visualization of the data per country. Per country, three timeseries (one for each pillar of ESG) followed by a correlation heatmap. Chapter 3 will deep-dive into Machine Learning by analyzing the data with the use of Random Forest, Linear Regression, and Multi-Layer Perceptron (MLP) neural network. By implementing the models, the GDP will be predicted and compared to each other using a timeseries and alpha-lambda plot. The k-fold cross-validation method is used within each model to make the outcome more reliable since the dataset is split into five folds by which one fold is used as test data. The average of the outcome per fold is generated as a result. Chapter 4 visualizes the interim results by which the best machine learning is chosen. This model will dive deep into Chapter 5 in each country's ESG pillar separately to determine which pillar (E, S, or G) will contribute sufficiently to the GDP and which will not. Chapter 6 will be about the conclusion and discussions.

Indicators per Category

For each category, we selected a set of 3/4 indicators that may have a significant influence on a country's GDP, and we have found reliable data for:

Social Indicators:

- *Health Expenditure as a Percentage of GDP:* This metric reflects a nation's commitment to healthcare and the well-being of its citizens.
- *Immunization:* A crucial element of public health, immunization rates can indicate the overall health of a population.
- *Life Expectancy at Birth and Child Mortality:* These metrics offer insights into the longevity and quality of life within a country.
- *Incidence of Diseases:* The prevalence of diseases can impact the productivity and economic output of a nation.

Economic (Governance) Indicators:

- *Export and Import:* These indicators measure a country's involvement in international trade and trade balance.
- *Freight:* Reflecting transportation efficiency, freight indicators provide valuable data for logistical and supply chain analysis.

Environmental Indicators:

- *Renewable Energy Consumption:* This indicator assesses a country's commitment to sustainable and environmentally friendly energy sources.
- *Energy Efficiency:* Measuring the effectiveness of energy use, this indicator speaks to the sustainability of a country's energy policies.
- *Greenhouse Gas (GHG) Emissions:* The environmental impact of GHG emissions is a crucial aspect of sustainability and economic development.

Import Packages

Here, we import essential libraries and packages for our project. These include libraries for data retrieval, such as wbdata, data visualization with plotly, machine learning tools from scikit-learn, data preprocessing with scaling, and additional libraries for data handling and visualization, including country converter and Matplotlib.

```
In [ ]:
# In this cell all the useful librabries are imported.
import wbdata
import plotly
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt
import country_converter as coco
import warnings
from sklearn.preprocessing import MinMaxScaler
from scipy.signal import savgol_filter
from sklearn.decomposition import PCA
import math
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, mean_absolute_error
from sklearn.model_selection import train_test_split

warnings.filterwarnings('ignore')
plotly.offline.init_notebook_mode()
```

1. Data retrieving

In this chapter, we focus on the initial step of our analysis: data retrieval. The data is sourced from multiple repositories, including the World Bank and the European Data Bank, to gather ESG indicators for four countries: The Netherlands, Germany, Greece, and Ireland.

Retrieving the indicators from the World Bank using the Package "wbdata" as DataFrames

The World Bank is an international financial institution that provides financial and technical assistance to developing countries. Established in 1944, its primary focus is on poverty mitigation and sustainable development. The World Bank offers many data sets, ranging from economic indicators like GDP and inflation rates to social metrics such as health expenditure and education levels.

The indicators that are retrieved from the World Bank are:

- Gross Domestic Product per Capita

- Social Indicators:
    - *Health Expenditure as a Percentage of GDP*
    - *Immunization*
    - *Life Expectancy at Birth and Child Mortality*
    - *Incidence of Diseases*
- Economic (Governance) Indicators:
    - *Export and Import*
    - *Freight*
- Environmental Indicators:
    - *Renewable Energy Consumption*

In [ ]:
```
# Define indicators Inequality and Social Welfare
health_indicators = {'SH.XPD.CHEX.GD.ZS': "Health Expenditure as a Percentage of GDP", "SH.IMM.IDPT": "Immunization"}
GDP_indicator = {'NY.GDP.PCAP.CD': 'gdppc'}
life_exp_indicator = {'SP.DYN.LE00.IN': 'Life Expectancy at Birth', 'SH.DYN.MORT': 'Child Mortality'}
disease_indicator = {'SH.TBS.INCD': 'Indicence of Diseases'}

# Define indicators Import & Export
export_indicator = {'NE.EXP.GNFS.KD.ZG': 'Export'}
import_indicator = {'NE.IMP.GNFS.KD.ZG': 'Import'}
freight_indicator = {'IS.AIR.GOOD.MT.K1': 'Freight'}

# Define indicators Evironmental
renewable_energy_indicator = {'EG.FEC.RNEW.ZS': 'Renewable energy consumption (% of total final energy consumption)'}
```

In [ ]:
```
countries = ['NLD', 'DEU', 'GRC', 'IRL']

# Dataframes Inequality and Social Welfare
df_health = wbdata.get_dataframe(health_indicators, country=countries, convert_date=True)
df_gdp = wbdata.get_dataframe(GDP_indicator, country=countries, convert_date=True)
df_life_exp = wbdata.get_dataframe(life_exp_indicator, country=countries, convert_date=True)
df_diseases = wbdata.get_dataframe(disease_indicator, country=countries, convert_date=True)

# Dataframes Import & Export
df_export = wbdata.get_dataframe(export_indicator, country=countries, convert_date=True)
df_import = wbdata.get_dataframe(import_indicator, country=countries, convert_date=True)
df_freight = wbdata.get_dataframe(freight_indicator, country=countries, convert_date=True)

# Dataframes Environmental
df_renewable = wbdata.get_dataframe(renewable_energy_indicator, country=countries, convert_date=True)
```

In [ ]:
```
# Resetting index of the dataframes
df_health = df_health.reset_index()
df_gdp = df_gdp.reset_index()
df_life_exp = df_life_exp.reset_index()
df_diseases = df_diseases.reset_index()
df_export = df_export.reset_index()
df_import = df_import.reset_index()
df_freight = df_freight.reset_index()
df_renewable = df_renewable.reset_index()

# Formatting the dates column of the indicators DataFrames to year format
df_health['date'] = df_health['date'].dt.year
df_health = df_health.drop(['Immunization'], axis = 1)
df_gdp['date'] = df_gdp['date'].dt.year
df_life_exp['date'] = df_life_exp['date'].dt.year
df_diseases['date'] = df_diseases['date'].dt.year
df_export['date'] = df_export['date'].dt.year
df_import['date'] = df_import['date'].dt.year
df_freight['date'] = df_freight['date'].dt.year
df_renewable['date'] = df_renewable['date'].dt.year
```

Retrieving the indicators from the European Data Bank by importing the csv of the individual datasets

The European Data Bank is a less centralized entity than the World Bank but serves a similar purpose within the context of the European Union. It collects data from various European institutions and offers specific indicators that are highly relevant to the EU's policy objectives. This is relevant for the scope of this study, which looks at four countries that are members of the European Union.

The indicators that are retrieved from the European Data Bank are:

- Environmental Indicators:
    - *Air GHG* - https://ec.europa.eu/eurostat/databrowser/view/sdg_13_10__custom_8184934/default/table?lang=en
    - *Energy Efficiency* - https://ec.europa.eu/eurostat/databrowser/view/nrg_ind_eff__custom_8226911/default/table?lang=en

In [ ]:
```
# Importing the csv files of the two indicators retrieved from the European Data Bank
df_emissions = df = pd.read_csv('Datasets/sdg_13_10_linear.csv')
df_efficiency =  pd.read_csv('Datasets/nrg_ind_eff_linear.csv')

# Converting ISO-2 country codes to ISO-3
cc = coco.CountryConverter()
df_emissions['geo'] = df_emissions['geo'].replace('EL', 'GR')
iso3_codes_emissions = cc.pandas_convert(series=df_emissions['geo'], to='ISO3')

# Removing columns that will not be used and renaming columns to merge DataFrame later with Dataframes from wbdata indicators
df_emissions['geo_3'] = iso3_codes_emissions
GHG = df_emissions[df_emissions['geo_3'].isin(countries) & (df_emissions['airpol'] == 'GHG') & (df_emissions['unit'] == 'T_HAB') & (df_emissions['src_crf'] == 'TOTXMEMONIA')]
GHG = GHG[['geo_3', 'OBS_VALUE', 'TIME_PERIOD']]
country_names = cc.pandas_convert(series = GHG['geo_3'], to = 'name_short')
GHG['country'] = country_names
GHG = GHG.drop('geo_3', axis = 1)
GHG = GHG.rename(columns = {'OBS_VALUE' : 'GHG', 'TIME_PERIOD' : 'date'})

# Converting ISO-2 country codes to ISO-3
df_efficiency['geo'] = df_efficiency['geo'].replace('EL', 'GR')
iso3_codes_efficiency = cc.pandas_convert(series=df_efficiency['geo'], to='ISO3')
df_efficiency['geo_3'] = iso3_codes_efficiency

# Removing columns that will not be used and renaming columns to merge DataFrame later with Dataframes from wbdata indicators
efficiency = df_efficiency[df_efficiency['geo_3'].isin(countries) & (df_efficiency['nrg_bal'] == 'PEC2020-2030') & (df_efficiency['unit'] == 'MTOE')]
efficiency = efficiency[['geo_3', 'OBS_VALUE', 'TIME_PERIOD']]
country_names = cc.pandas_convert(series = efficiency['geo_3'], to = 'name_short')
efficiency['country'] = country_names
efficiency = efficiency.drop('geo_3', axis = 1)
efficiency = efficiency.rename(columns = {'OBS_VALUE' : 'Energy efficiency', 'TIME_PERIOD' : 'date'})
```

```
WARNING:country_converter.country_converter:EU27_2020 not found in regex
WARNING:country_converter.country_converter:EU27_2020 not found in regex
```

In [ ]:
```
# Indicators in a list for easy search
indicators_social = ['Health Expenditure', 'Life Expectancy at Birth', 'Child Mortality', 'Indicence of Diseases']
indicators_env = ['GHG', 'Renewable', 'Energy efficiency']
indicators_economy = ['Export', 'Import', 'Freight']
```

Merging the indicators for all four countries to a main DataFrame named ESG and Setting up timeframe (2001 - 2020)

For this study, the timeframe selected is from the year 2001 to 2020. This decision is made in terms of the data availability across all chosen indicators from both the World Bank and the European Data Bank. The 20-year period offers enough data to perform Machine Learning methods later in the study.

In [ ]:
```
# Making of the main Dataframe in order to easily do an analysis
ESG = df_gdp
ESG = pd.merge(ESG, df_health, how = 'inner')
ESG = pd.merge(ESG, df_life_exp, how = 'inner')
ESG = pd.merge(ESG, df_diseases, how = 'inner')
ESG = pd.merge(ESG, df_export, how = 'inner')
ESG = pd.merge(ESG, df_import, how = 'inner')
ESG = pd.merge(ESG, df_freight, how = 'inner')
ESG = pd.merge(ESG, df_renewable, how = 'inner')
ESG = pd.merge(ESG, GHG, how = 'inner')
ESG = pd.merge(ESG, efficiency, how = 'inner')
ESG = ESG.rename(columns= {'Renewable energy consumption (% of total final energy consumption)': 'Renewable', 'Health Expenditure as a Percentage of GDP': 'Health Expenditure'})

ESG = ESG[(ESG['date'] >= 2001) & (ESG['date'] <= 2020)]
ESG.head()
```

Out[ ]:

	country	date	gdppc	Health Expenditure	Life Expectancy at Birth	Child Mortality	Indicence of Diseases	Export	Import	Freight	Renewable	GHG	Energy efficiency
1	Germany	2020	46772.825351	12.822489	81.041463	3.6	5.3	-9.274737	-8.502678	9166.371283	18.60	9.0	262.10
2	Germany	2019	46793.686762	11.696230	81.292683	3.7	6.1	1.265017	2.859892	7763.619214	17.07	9.8	285.24
3	Germany	2018	47939.278288	11.457275	80.892683	3.8	7.0	2.223462	3.992724	7969.863640	16.04	10.5	291.95
4	Germany	2017	44652.589172	11.324208	80.992683	3.9	7.1	4.898995	5.225381	7901.652344	15.22	10.9	298.12
5	Germany	2016	42136.120791	11.232638	80.990244	3.9	7.7	2.470000	4.490000	6942.706332	14.24	11.1	297.63

2: Data Visualization and Correlation

In this chapter, we visualize the data of every indicator for each country and indicate the correlation to the GDP per capita.

Exploring Indicator Relationships to a country's GDP:

Firstly, we use Plotly to create a series of data visualizations, categorizing them into three sections: Social, Economic, and Environmental indicators. Every section visualizes the relationships between the indicators for the different categories and a country's Gross Domestic Product (GDP).

In the Social indicators plot (the first figure), we dive into critical social indications (focussed on health) and try to see their connection with GDP per capita. Including a black dashed line representing GDP per capita acts as a reference point, helping us see the influence of social factors on economic trends.

In the Economic indicators plot (the second figure), we examine trends in economic data alongside GDP per capita. This analysis provides insights into how economic factors correspond to changes in economic trends.

The Environmental indicators plot (the third figure) gives the trends of the environmental indicators alongside the GDP per capita. This visualization contributes to understanding the relationship between environmental sustainability and economic development.

Analyzing Indicator Correlations:

After the visualization, we use Pandas to calculate a correlation matrix for every country. The resulting correlation matrix is translated into an interactive heatmap (fig), where color intensity reflects the strength and direction of correlations. This visualization is instrumental in identifying potential patterns and dependencies between indicators.

By reading the correlation matrix, we get valuable insights into how different factors correlate with the GDP per capita. This offers a deeper understanding of which indicators may have positive or negative impacts on the GDP of a country.

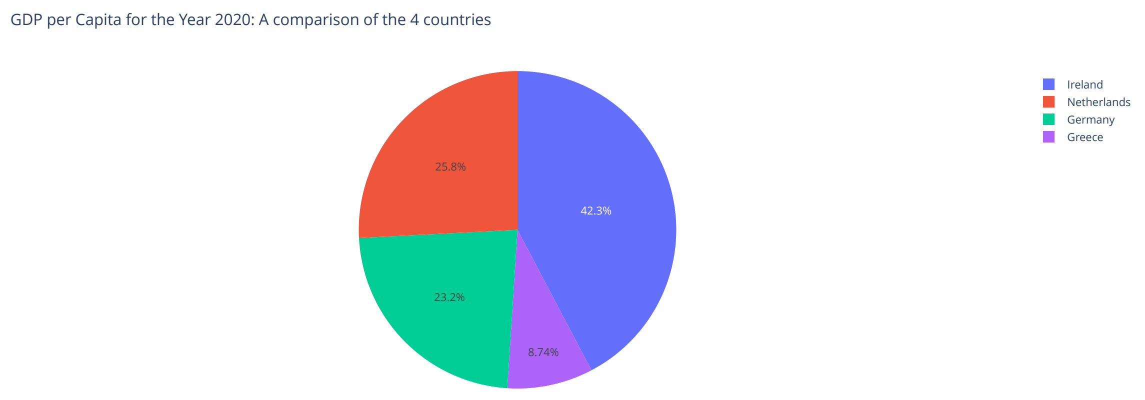
Note: 'gdppc' stand for Gross Domestic Product (GDP) per capita

Proportions of the GDP per capita for every country compared

What can be seen, is that Ireland has a relative high GDP per capita and Greece much smaller. Germany and the Netherlands are almost identical.

In [ ]:

```
# Making of a piechart
df_bar = ESG[ESG['date'] == 2020]
fig = px.pie(df_bar, names='country', values='gdppc',
             title='GDP per Capita for the Year 2020: A comparison of the 4 countries', hole=0, width = 1300)

fig.show()
```



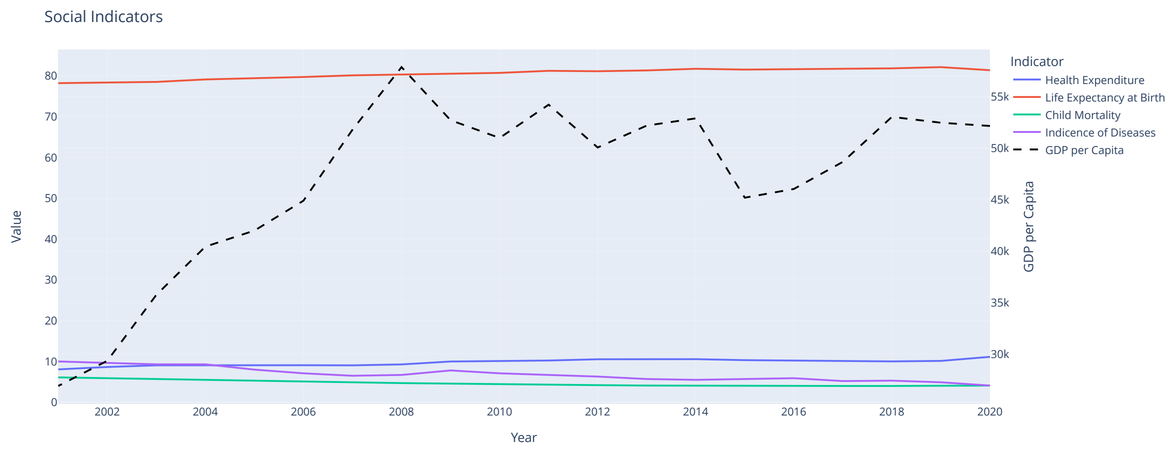
The Netherlands

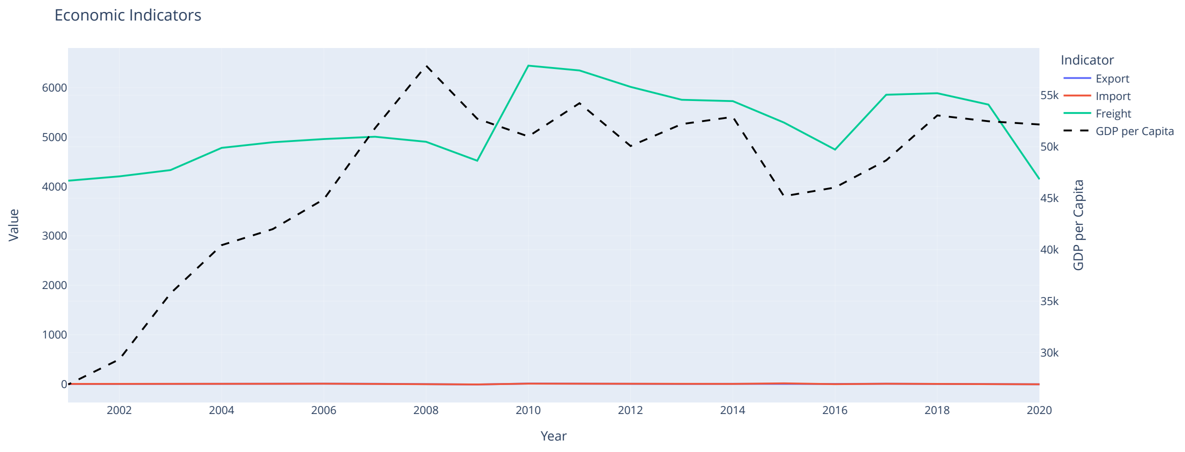
The Netherlands Social, Economic and Environmental indicators, and GDP per capita plotted:

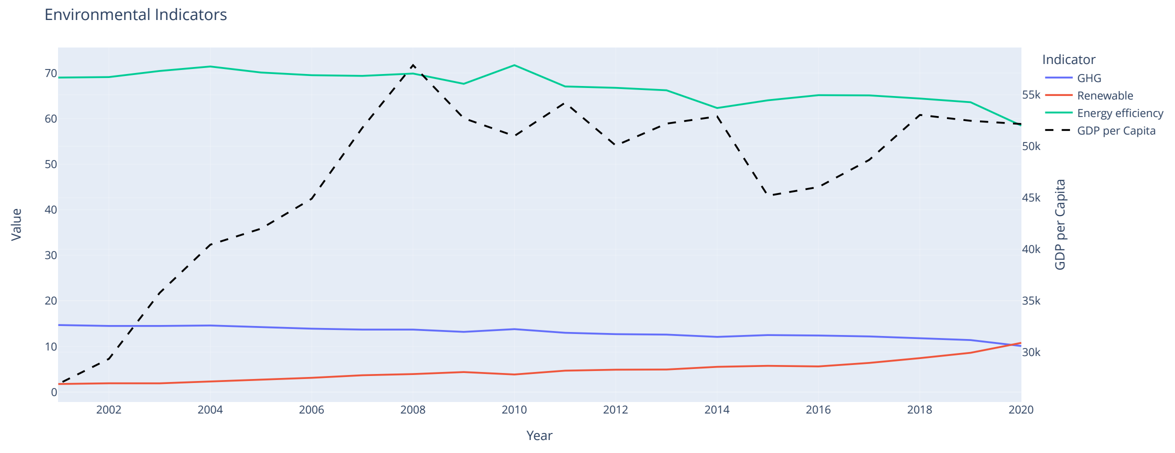
In [ ]:

```
# Making of figuers for country Netherlands for the social indicators
fig1 = px.line(ESG[ESG['country'] == 'Netherlands'], x='date', y=indicators_social, width=1300)
fig1.add_scatter(x=ESG[ESG['country'] == 'Netherlands']['date'],
                 y=ESG[ESG['country'] == 'Netherlands']['gdppc'],
                 name='GDP per Capita', yaxis="y2",
                 line=dict(color='black', dash='dash'))
fig1.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                   title='Social Indicators',
                   legend_title=dict(text='Indicator'),
                   yaxis1=dict(title='Value'),
                   xaxis=dict(title='Year'))
fig1.show()

# Making of figuers for country Netherlands for the economy indicators
fig2 = px.line(ESG[ESG['country'] == 'Netherlands'], x='date', y=indicators_economy, width=1300)
fig2.add_scatter(x=ESG[ESG['country'] == 'Netherlands']['date'],
                 y=ESG[ESG['country'] == 'Netherlands']['gdppc'],
                 name='GDP per Capita', yaxis="y2",
                 line=dict(color='black', dash='dash'))
fig2.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                   title='Economic Indicators',
                   legend_title=dict(text='Indicator'),
                   yaxis1=dict(title='Value'),
                   xaxis=dict(title='Year'))
fig2.show()

# Making of figuers for country Netherlands for the environmental indicators
```

```
fig3 = px.line(ESG[ESG['country'] == 'Netherlands'], x='date', y=indicators_env, width=1300)
fig3.add_scatter(x=ESG[ESG['country'] == 'Netherlands']['date'],
                 y=ESG[ESG['country'] == 'Netherlands']['gdppc'],
                 name='GDP per Capita', yaxis="y2",
                 line=dict(color='black', dash='dash'))
fig3.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                   title='Environmental Indicators',
                   legend_title=dict(text='Indicator'),
                   yaxis1=dict(title='Value'),
                   xaxis=dict(title='Year'))
fig3.show()
```





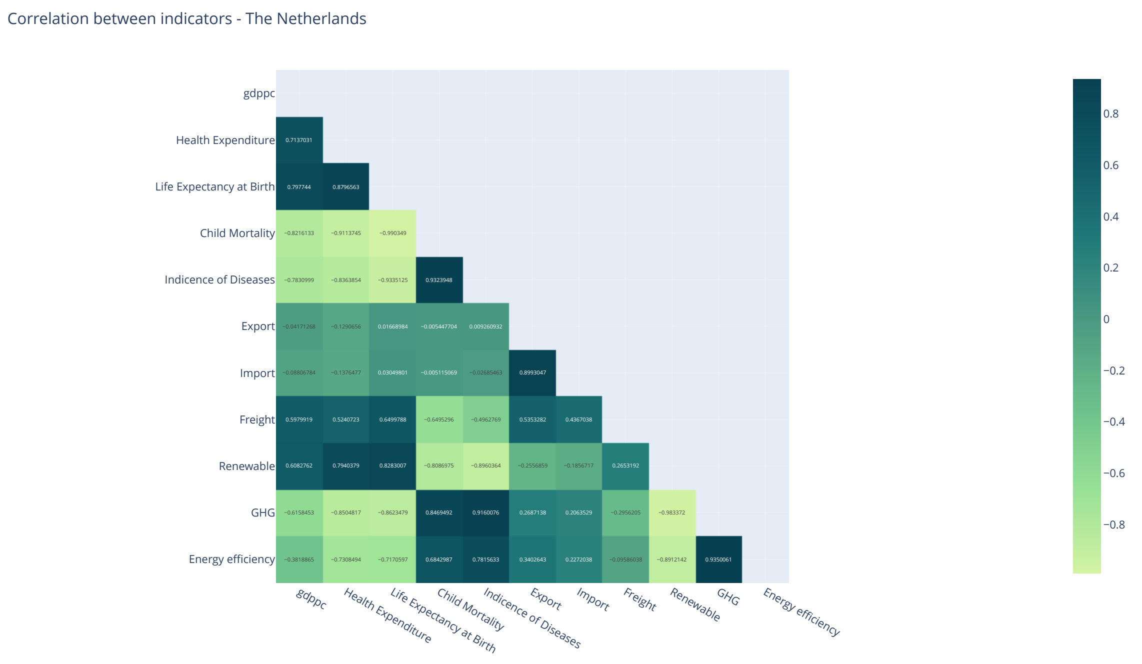


Correlation between all indicators and GDP per capita:

The correlation coefficient is a statistical measure that quantifies the strength and direction of the linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no linear relationship. A positive value implies that as one variable increases, the other tends to increase, while a negative value suggests the opposite. On the right side of the figure below, the correlation coefficients of indicators with GDP are given, indicating their relationship.

Look at the heatmap below. The social indicators focussed on health all have a strong correlation to the GDP per capita. The expectation is that the social indicators, therefore, give a good prediction of the GDP. For instance, the correlation between export and GDP is much weaker.

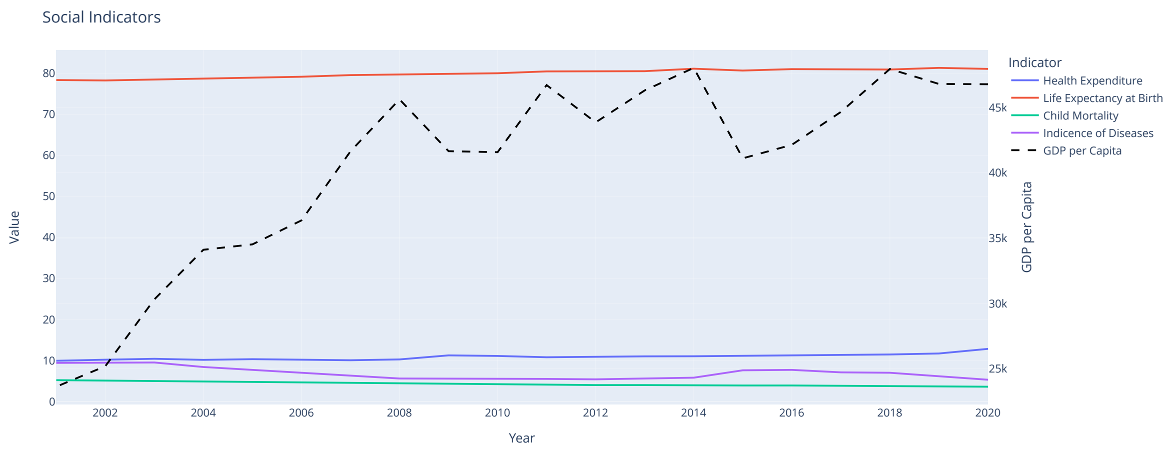
```
In [ ]: # Making of correlation heatmap for the Netherlands with using the corr. function
        df_corr_nld = ESG[ESG['country'] == 'Netherlands'].drop(['date', 'country'], axis = 1)
        corr = df_corr_nld.corr()
        mask = np.triu(np.ones_like(corr, dtype=bool))
        corr.where(~mask, inplace=True)

        fig = px.imshow(corr, text_auto=True, color_continuous_scale= px.colors.sequential.Emrld, width= 1300, height=750, title = 'Correlation between indicators - The Netherlands')

        fig.show()
```

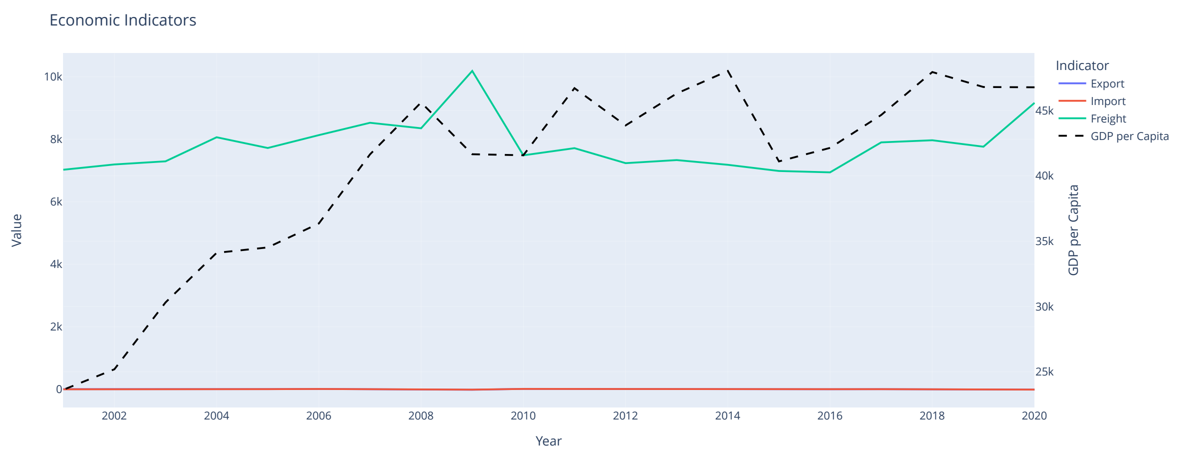


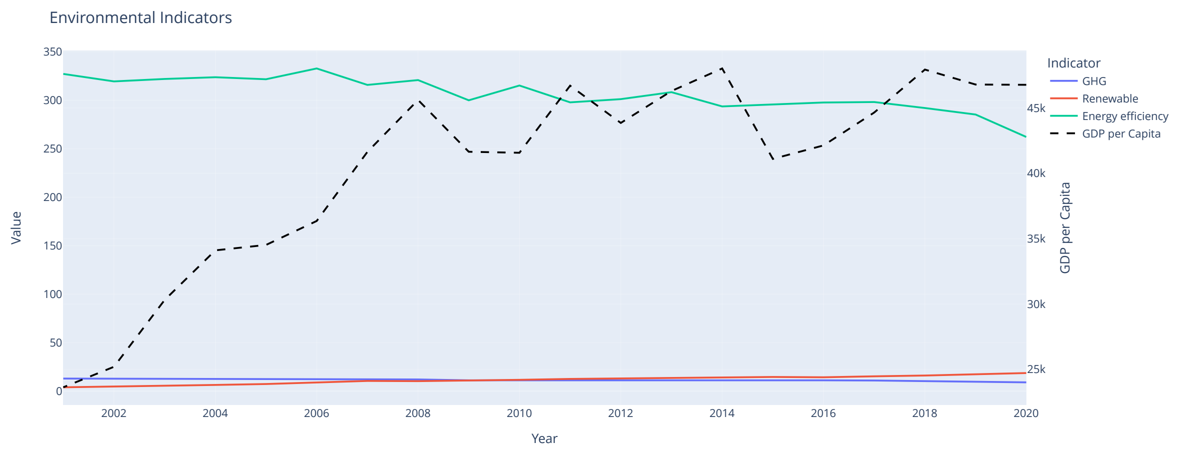
Germany

Germany's Social, Economic and Environmental indicators, and GDP per capita plotted:

```
In [ ]: # Making of figuers for country Germany for the social indicators
        fig1 = px.line(ESG[ESG['country']=='Germany'], x='date', y=indicators_social, width = 1300)
        fig1.add_scatter(x=ESG[ESG['country']=='Germany']['date'],
                         y=ESG[ESG['country']=='Germany']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig1.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                           title='Social Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig1.show()

        # Making of figuers for country Germany for the economy indicators
        fig2 = px.line(ESG[ESG['country']=='Germany'], x='date', y=indicators_economy, width = 1300)
        fig2.add_scatter(x=ESG[ESG['country']=='Germany']['date'],
                         y=ESG[ESG['country']=='Germany']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig2.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                           title='Economic Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1=dict(title='Value'), xaxis = dict(title = 'Year'))
        fig2.show()

        # Making of figuers for country Germany for the environmental indicators
        fig3 = px.line(ESG[ESG['country']=='Germany'], x='date', y=indicators_env, width = 1300)
        fig3.add_scatter(x=ESG[ESG['country']=='Germany']['date'],
                         y=ESG[ESG['country']=='Germany']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig3.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right', ),
                           title='Environmental Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig3.show()
```



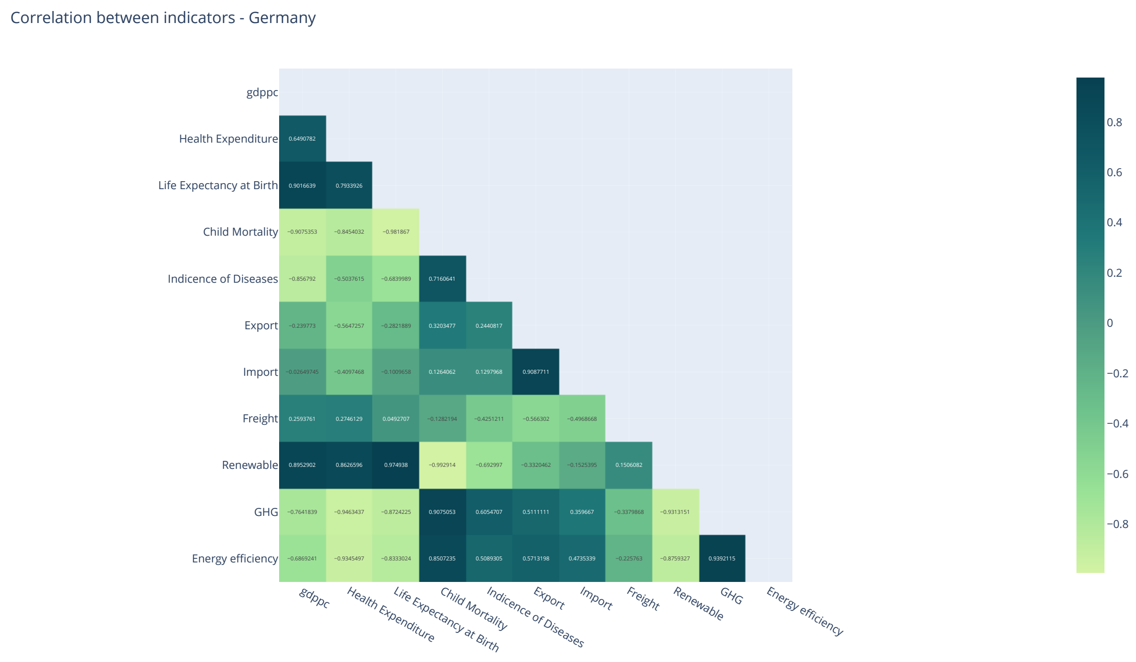




Correlation between all indicators and GDP per capita:

Looking at the correlation heatmap of Germany's ESG indicators and the GDP per capita, the social indicators strongly correlate with the GDP. Also, renewable energy consumption is strongly correlated with the GDP of Germany.

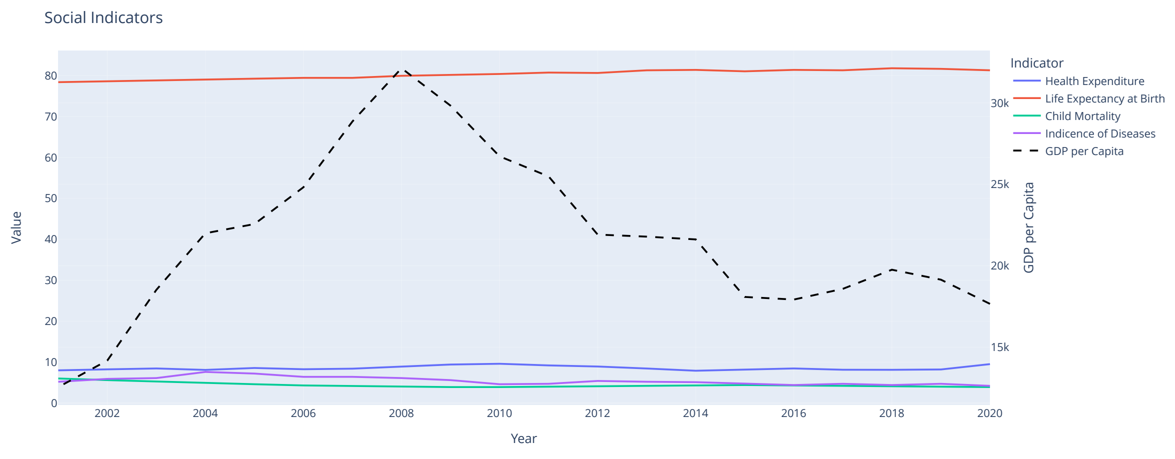
```
In [ ]: # Making of correlation heatmap for Germany with using the corr. function
        df_corr_deu = ESG[ESG['country'] == 'Germany'].drop(['date', 'country'], axis = 1)
        corr = df_corr_deu.corr()
        mask = np.triu(np.ones_like(corr, dtype=bool))
        corr.where(~mask, inplace=True)

        fig = px.imshow(corr, text_auto=True, color_continuous_scale= px.colors.sequential.Emrld, width= 1300, height=750, title = 'Correlation between indicators - Germany')

        fig.show()
```

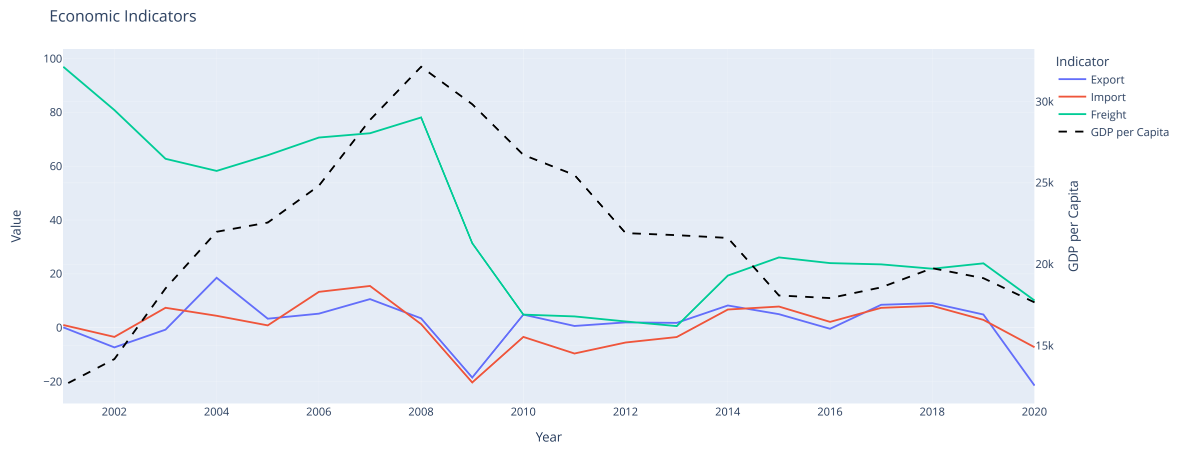


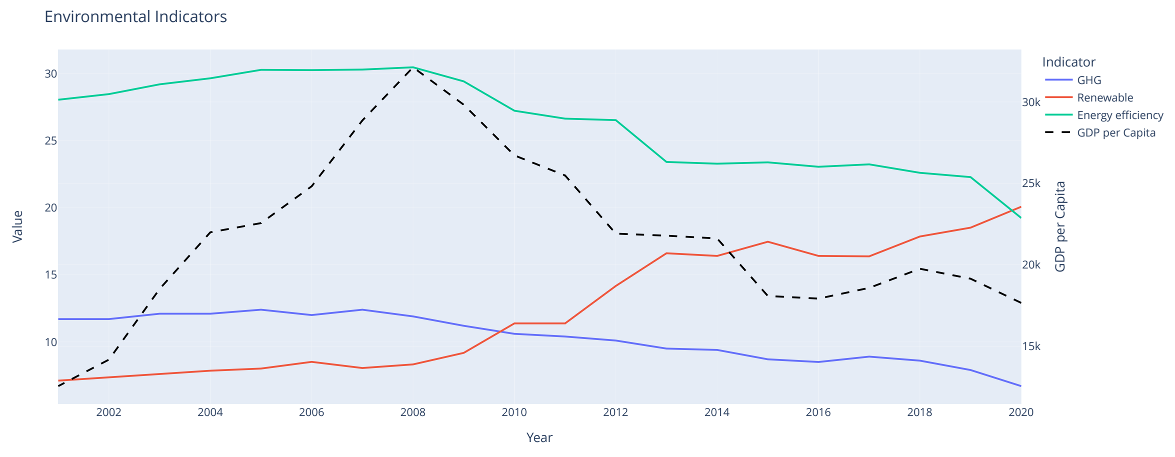
Greece

Greece's Social, Economic and Environmental indicators, and GDP per capita plotted:

```
In [ ]: # Making of figuers for country Greece for the social indicators
        fig1 = px.line(ESG[ESG['country']=='Greece'], x='date', y=indicators_social, width = 1300)
        fig1.add_scatter(x=ESG[ESG['country']=='Greece']['date'],
                         y=ESG[ESG['country']=='Greece']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig1.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                           title='Social Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig1.show()

        # Making of figuers for country Greece for the economy indicators
        fig2 = px.line(ESG[ESG['country']=='Greece'], x='date', y=indicators_economy, width = 1300)
        fig2.add_scatter(x=ESG[ESG['country']=='Greece']['date'],
                         y=ESG[ESG['country']=='Greece']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig2.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                           title='Economic Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1=dict(title='Value'), xaxis = dict(title = 'Year'))
        fig2.show()

        # Making of figuers for country Greece for the environmental indicators
        fig3 = px.line(ESG[ESG['country']=='Greece'], x='date', y=indicators_env, width = 1300)
        fig3.add_scatter(x=ESG[ESG['country']=='Greece']['date'],
                         y=ESG[ESG['country']=='Greece']['gdppc'],
                         name='GDP per Capita', yaxis="y2",
                         line = dict(color = 'black', dash = 'dash'))
        fig3.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right', ),
                           title='Environmental Indicators', legend_title = dict(text = 'Indicator'),
                           yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig3.show()
```





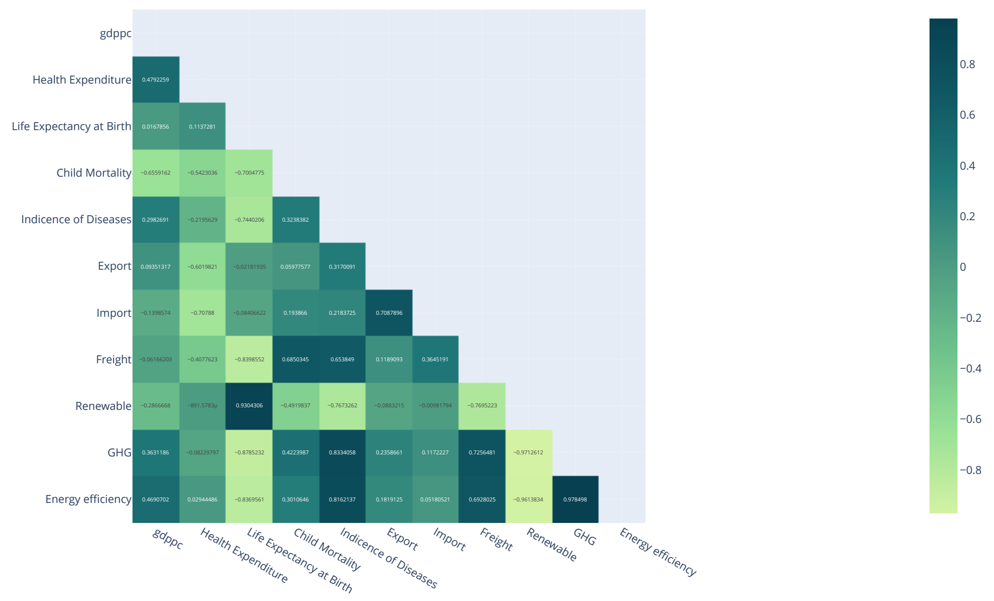


Correlation between all indicators and GDP per capita:

In the case of Greece, there is less correlation between the indicators and the GDP compared to other countries. The expectation is that the GDP of Greece will be harder to predict because there is less correlation, according to the heatmap below.

```
In [ ]: # Making of correlation heatmap for Greece with using the corr. function
        df_corr_grc = ESG[ESG['country'] == 'Greece'].drop(['date', 'country'], axis = 1)
        corr = df_corr_grc.corr()
        mask = np.triu(np.ones_like(corr, dtype=bool))
        corr.where(~mask, inplace=True)

        fig = px.imshow(corr, text_auto=True, color_continuous_scale= px.colors.sequential.Emrld, width= 1300, height=750, title = 'Correlation between indicators - Greece')

        fig.show()
```

Correlation between indicators - Greece

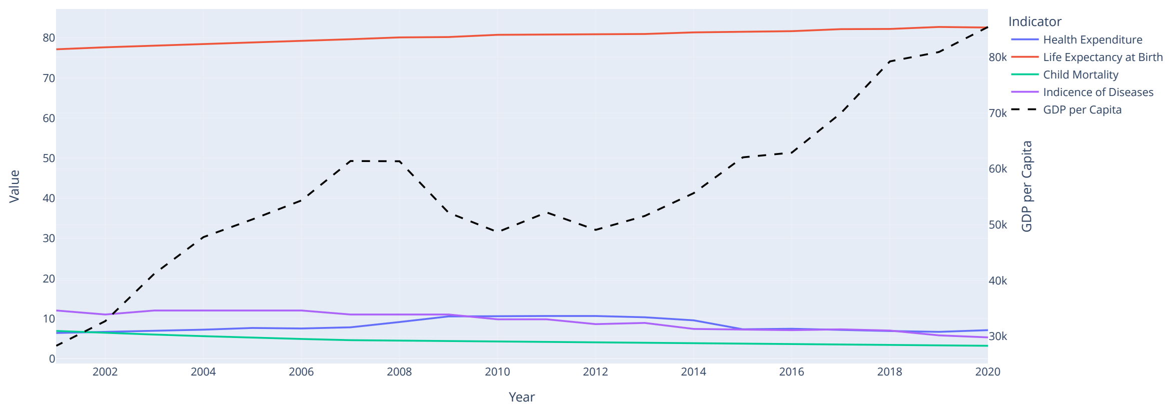


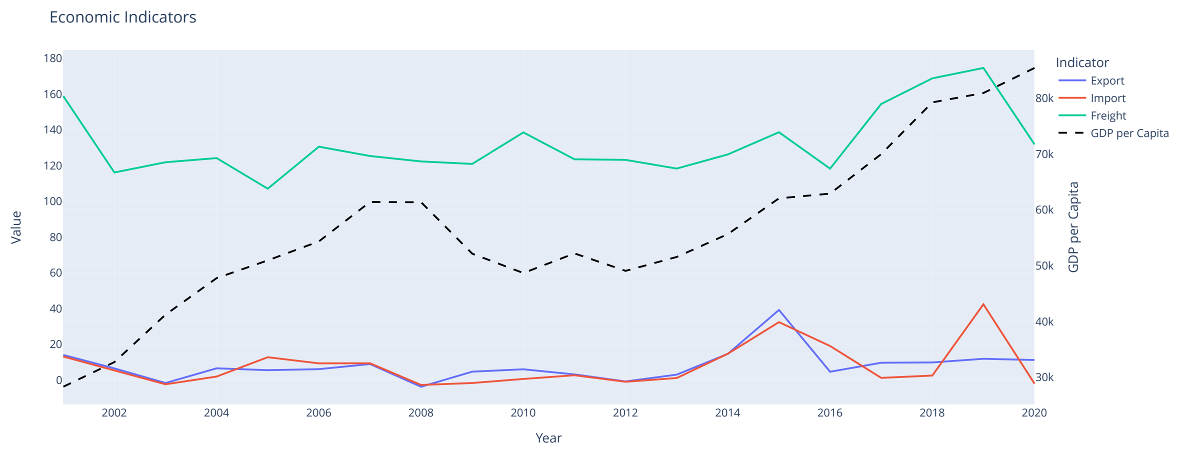
Ireland

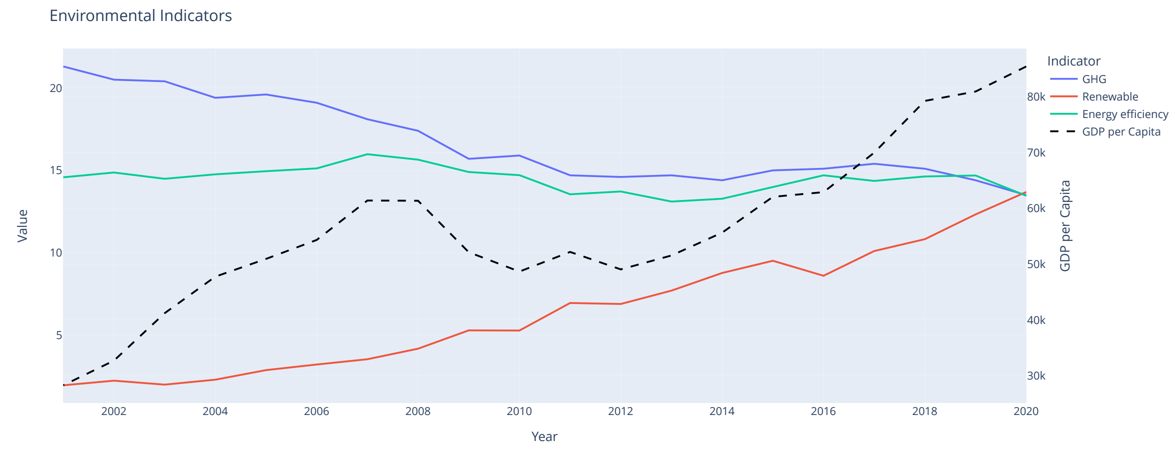
Ireland's Social, Economic and Environmental indicators, and GDP per capita plotted:

```
In [ ]: # Making of figuers for country Ireland for the social indicators
        fig1 = px.line(ESG[ESG['country']=='Ireland'], x='date', y=indicators_social, width = 1300)
        fig1.add_scatter(x=ESG[ESG['country']=='Ireland']['date'],
                        y=ESG[ESG['country']=='Ireland']['gdppc'],
                        name='GDP per Capita', yaxis="y2",
                        line = dict(color = 'black', dash = 'dash'))
        fig1.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                          title='Social Indicators', legend_title = dict(text = 'Indicator'),
                          yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig1.show()

        # Making of figuers for country Ireland for the economy indicators
        fig2 = px.line(ESG[ESG['country']=='Ireland'], x='date', y=indicators_economy, width = 1300)
        fig2.add_scatter(x=ESG[ESG['country']=='Ireland']['date'],
                        y=ESG[ESG['country']=='Ireland']['gdppc'],
                        name='GDP per Capita', yaxis="y2",
                        line = dict(color = 'black', dash = 'dash'))
        fig2.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right'),
                          title='Economic Indicators', legend_title = dict(text = 'Indicator'),
                          yaxis1=dict(title='Value'), xaxis = dict(title = 'Year'))
        fig2.show()

        # Making of figuers for country Ireland for the environmental indicators
        fig3 = px.line(ESG[ESG['country']=='Ireland'], x='date', y=indicators_env, width = 1300)
        fig3.add_scatter(x=ESG[ESG['country']=='Ireland']['date'],
                        y=ESG[ESG['country']=='Ireland']['gdppc'],
                        name='GDP per Capita', yaxis="y2",
                        line = dict(color = 'black', dash = 'dash'))
        fig3.update_layout(yaxis2=dict(title="GDP per Capita", overlaying='y', side='right', ),
                          title='Environmental Indicators', legend_title = dict(text = 'Indicator'),
                          yaxis1= dict(title = 'Value'), xaxis = dict(title = 'Year'))
        fig3.show()
```

Social Indicators



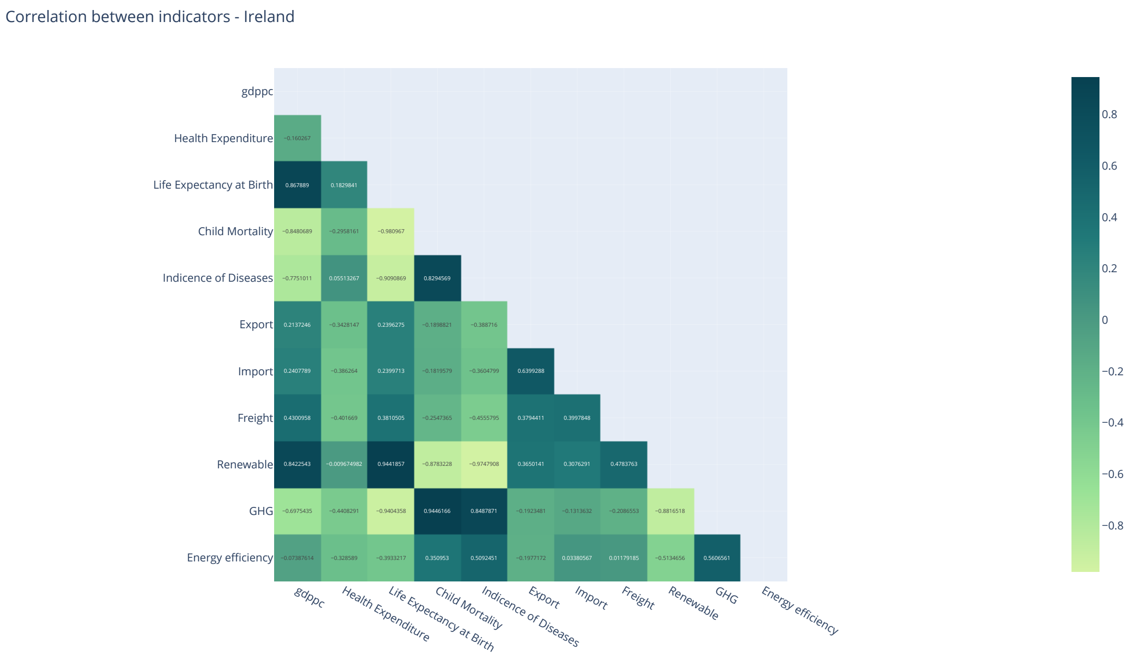




Correlation between all indicators and GDP per capita:

The GDP per capita of Ireland has a strong correlation with most of the indicators used to predict the GDP. Renewable energy consumption, child mortality, and life expectancy are highly correlated to the GDP of Ireland.

```
In [ ]:
# Making of correlation heatmap for Irleland with using the corr. function
df_corr_irl = ESG[ESG['country'] == 'Ireland'].drop(['date', 'country'], axis = 1)
corr = df_corr_irl.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
corr.where(~mask, inplace=True)

fig = px.imshow(corr, text_auto=True, color_continuous_scale= px.colors.sequential.Emrld, width= 1300, height=750, title = 'Correlation between indicators - Ireland')

fig.show()
```



3: Machine learning models: A comparative Study

This chapter focuses on the evaluation of three prominent machine-learning models:

- Random Forest: An ensemble model that creates multiple decision trees during training and outputs an averaged result.

- Linear Regression: A statistical method that models the linear relationship between features, helpful in understanding the influence of individual predictors.
- MLP Neural Network: A deep learning model capable of capturing complex, non-linear relationships using different layers of different sizes.

All three models are supervised learning techniques, meaning that they use labeled data to train and make predictions. Firstly, the data is denoised using the Savitzky-Golay (Savgol) filter, which smoothens the data. Denoising is a critical step in preparing data for machine learning models. It involves removing irrelevant or noisy information from the data, resulting in improved model performance, robustness, and accuracy. A window size of 5 is used in order to create sufficient smoothing by which the data is not getting lost. The result of the smoothing can be seen in the graph underneath the smoothing process. Additionally, a Principal Component Analysis (PCA) is done to tackle the 'Curse of dimensionality'. By looking at the cumulative variance in the plots, a dimensionality of two can be chosen because the variance exceeds the strong threshold of 95%.

Furthermore, in the visualization part, two graphs are given per country. The first graph is the alpha lambda plot. These plots visualize the errors compared to the true (raw) GDP. An alpha of 20% is chosen to see which predictions are outside the 20% margin. The second plot shows the predicted GDP based on the folds.

```
In [ ]: # Print Dataframe before denoising
        ESG.head()
```

Out[ ]:

	country	date	gdppc	Health Expenditure	Life Expectancy at Birth	Child Mortality	Indicence of Diseases	Export	Import	Freight	Renewable	GHG	Energy efficiency
1	Germany	2020	46772.825351	12.822489	81.041463	3.6	5.3	-9.274737	-8.502678	9166.371283	18.60	9.0	262.10
2	Germany	2019	46793.686762	11.696230	81.292683	3.7	6.1	1.265017	2.859892	7763.619214	17.07	9.8	285.24
3	Germany	2018	47939.278288	11.457275	80.892683	3.8	7.0	2.223462	3.992724	7969.863640	16.04	10.5	291.95
4	Germany	2017	44652.589172	11.324208	80.992683	3.9	7.1	4.898995	5.225381	7901.652344	15.22	10.9	298.12
5	Germany	2016	42136.120791	11.232638	80.990244	3.9	7.7	2.470000	4.490000	6942.706332	14.24	11.1	297.63

```
In [ ]: # Denoising of the Dataframe with the savgol filter en window size 5, but not for gdppc
        ESG_denoised = ESG.copy()
        for country in ESG.country.unique():
            for column in [col for col in ESG.columns if col not in ['country', 'date', 'gdppc']]:
                ESG_denoised.loc[ESG['country'] == country, column] = savgol_filter(ESG.loc[ESG['country'] == country, column], window_length=5, polyorder=1)
```

The first values of the datasets of the indicators after denoising with Savgol filter:

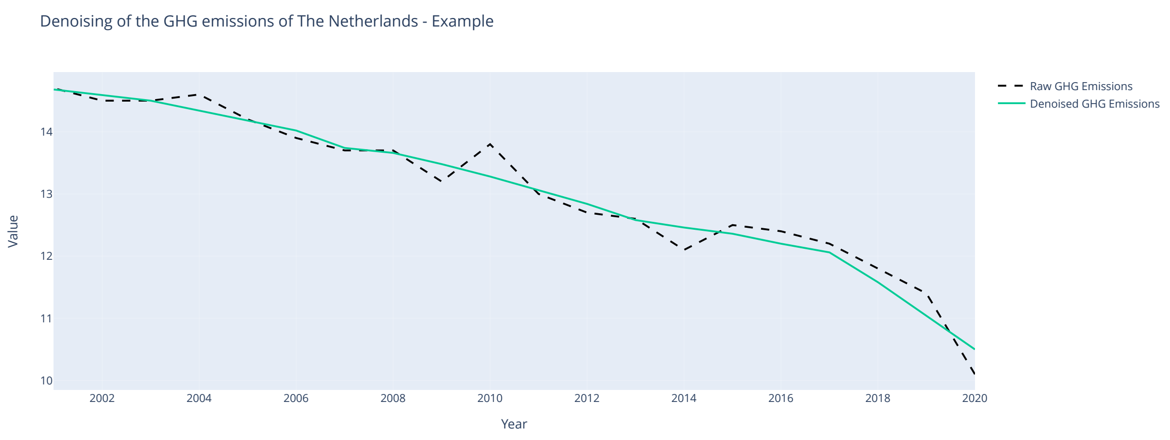
```
In [ ]: # Print Dataframe after denoising
        ESG_denoised.head()
```

Out[ ]:

	country	date	gdppc	Health Expenditure	Life Expectancy at Birth	Child Mortality	Indicence of Diseases	Export	Import	Freight	Renewable	GHG	Energy efficiency
1	Germany	2020	46772.825351	12.416913	81.122439	3.62	5.48	-5.108143	-4.057105	8810.701917	18.348	9.20	270.220
2	Germany	2019	46793.686762	12.061740	81.082195	3.70	6.06	-2.395798	-1.222021	8379.772240	17.291	9.73	278.614
3	Germany	2018	47939.278288	11.706568	81.041951	3.78	6.64	0.316547	1.613064	7948.842563	16.234	10.26	287.008
4	Germany	2017	44652.589172	11.378844	80.961951	3.84	7.10	3.259643	4.473142	7512.849395	15.424	10.68	293.774
5	Germany	2016	42136.120791	11.244037	80.921463	3.90	7.04	3.965756	4.686210	7396.954794	14.814	10.96	295.446

Below the raw data and the denoised data of the GHG emissions of the Netherlands is plotted in one graph to visualize the difference after the denoising process:

```
In [ ]: # Making of a plotly plot to show the denoised vs raw data
        fig = px.line(title='Denoising of the GHG emissions of The Netherlands - Example', width = 1300)
        fig.add_scatter(x=ESG[ESG['country']=='Netherlands']['date'],
                        y=ESG[ESG['country']=='Netherlands']['GHG'],
                        name='Raw GHG Emissions', line = dict(color = 'black', dash = 'dash'))
        fig.add_scatter(x=ESG_denoised[ESG_denoised['country']=='Netherlands']['date'],
                        y=ESG_denoised[ESG_denoised['country']=='Netherlands']['GHG'],
                        name='Denoised GHG Emissions')
        fig.update_layout(yaxis=dict(title='Value'), xaxis = dict(title = 'Year'))
        # Show the plot
        fig.show()
```

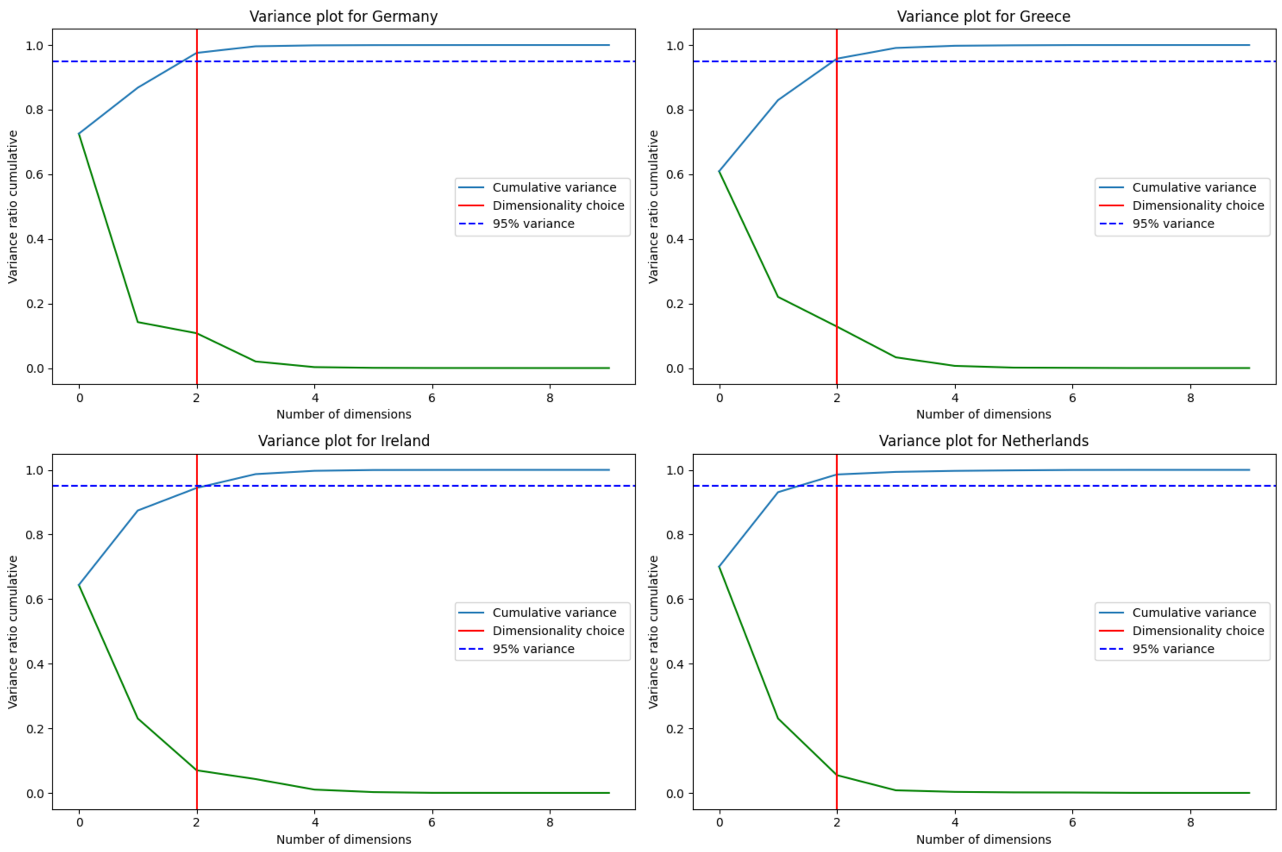


PCA dimensionality component analysis

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of a dataset while retaining the most important information. It works by transforming the original features into a new set of orthogonal features called principal components, with each component capturing the most significant patterns in the data, thus simplifying complex data and improving model performance.

```
In [ ]: # Unique countries
        countries = ESG_denoised['country'].unique()

        # Make the figure
        fig, axes = plt.subplots(2, 2, figsize=(15, 10))
        axes = axes.ravel()

        # For every country a PCA plot is made
        for index, country in enumerate(countries):
            country_data = ESG_denoised[ESG_denoised['country'] == country]
            country_data_scaled = country_data.drop(columns=['country', 'date', 'gdppc'])

            scaler_standard = StandardScaler()
            country_data_scaled = scaler_standard.fit_transform(country_data_scaled)

            pca = PCA().fit(country_data_scaled)

            axes[index].plot(pca.explained_variance_ratio_, color='g')
            axes[index].plot(pca.explained_variance_ratio_.cumsum(), label='Cumulative variance')
            axes[index].set_xlabel('Number of dimensions')
            axes[index].set_ylabel('Variance ratio cumulative')
            axes[index].set_title(f'Variance plot for {country}')
            axes[index].axvline(2, color='r', label='Dimensionality choice')
            axes[index].axhline(0.95, color='b', linestyle= '--', label='95% variance')
            axes[index].legend()
```

```python
# Figure is shown
plt.tight_layout()
plt.show()
```



In [ ]:
```python
# For each country do a separate PCA
PCA_total = []
for country in ESG_denoised['country'].unique():
    country_data = ESG_denoised[ESG_denoised['country'] == country]
    country_data_scaled = country_data.drop(columns=['country', 'date', 'gdppc'])

    scaler_standard = StandardScaler()
    country_data_scaled = scaler_standard.fit_transform(country_data_scaled)

    pca = PCA(n_components=2).fit(country_data_scaled)
    country_pca = pca.transform(country_data_scaled)

    PCA_percountry = pd.DataFrame(data=country_pca, columns=['pca1','pca2'])
    PCA_percountry['country'] = country
    PCA_percountry['date'] = country_data['date'].values
    PCA_percountry['gdppc'] = country_data['gdppc'].values

    PCA_total.append(PCA_percountry)

PCA_columns=['pca1','pca2']
# Merge the PCA's per country in a dataframe
ESG_pca_df = pd.concat(PCA_total, ignore_index=True)
ESG_pca_df.head()
```

Out[ ]:

	pca1	pca2	country	date	gdppc
0	5.996098	2.369383	Germany	2020	46772.825351
1	4.537602	1.321426	Germany	2019	46793.686762
2	3.079106	0.273468	Germany	2018	47939.278288
3	1.735894	-0.845695	Germany	2017	44652.589172
4	1.304738	-0.998890	Germany	2016	42136.120791

Functions for retrieving results

This code defines the results function, which calculates and displays key regression evaluation metrics, including RMSE, R2, and MAE. It is used to assess the performance of machine learning models for different countries, providing a quick summary of prediction accuracy.

In [ ]:
```python
# function to get the results fastly per country
def results(y_test, y_predict, method, country, K_number):
    MSE = mean_squared_error(y_test, y_predict)
    RMSE = math.sqrt(MSE)
    R2 = r2_score(y_test, y_predict)
    MAE = mean_absolute_error(y_test, y_predict)
    return print(f'{method}, {country}, K-fold #{K_number}: the RMSE is {RMSE}, and R2 is {R2}, and MAE is {MAE}')
```

In [ ]:
```python
# Function to get the metric and not only text as above
def get_metric(y_test, y_predict):
    """ MSE, RMSE, R2, and MAE are generated in this function using libraries"""
    MSE = mean_squared_error(y_test, y_predict)
    RMSE = math.sqrt(MSE)
    R2 = r2_score(y_test, y_predict)
    MAE = mean_absolute_error(y_test, y_predict)
    return MSE, RMSE, R2, MAE
```

K fold cross validation

K-fold cross-validation is a widely used technique in machine learning for assessing predictive models like ours. It is particularly useful when working with limited data, and we only have 20 years of data. The key idea is to divide the available data into 'K' equally-sized subsets, or 'folds.' In our model, these are subsets of 4 years each. The model is trained and tested five times, each time using a different fold, or different four years, as the test set and the remaining folds for training. This process helps ensure that the model's performance is not dependent on a specific train-test split, making it a valuable tool for assessing its stability and accuracy.

In [ ]:
```python
def cross_validate_train_test_sets(data, pca_columns, K_subset):
    """
    When a cross validaiton is done, this function is called which split the dataset in 5 fold
    and uses 1 fold for the test set, a complete list of lists is made for each fold
    """
    dates = sorted(data['date'].unique())
    nr_dates = len(dates)
    subset_size = int(nr_dates / K_subset)
    shuffle_data = []

    for shuffle in range(K_subset):
        start = shuffle * subset_size
        test_dates = dates[shuffle * subset_size:start + subset_size]
        train_dates = [date for date in dates if date not in test_dates]

        X_train = data[data['date'].isin(train_dates)][pca_columns]
        y_train = data[data['date'].isin(train_dates)]['gdppc']
        X_test = data[data['date'].isin(test_dates)][pca_columns]
        y_test = data[data['date'].isin(test_dates)]['gdppc']

        shuffle_data.append((X_train, y_train, X_test, y_test))

    return shuffle_data
```

Random Forest

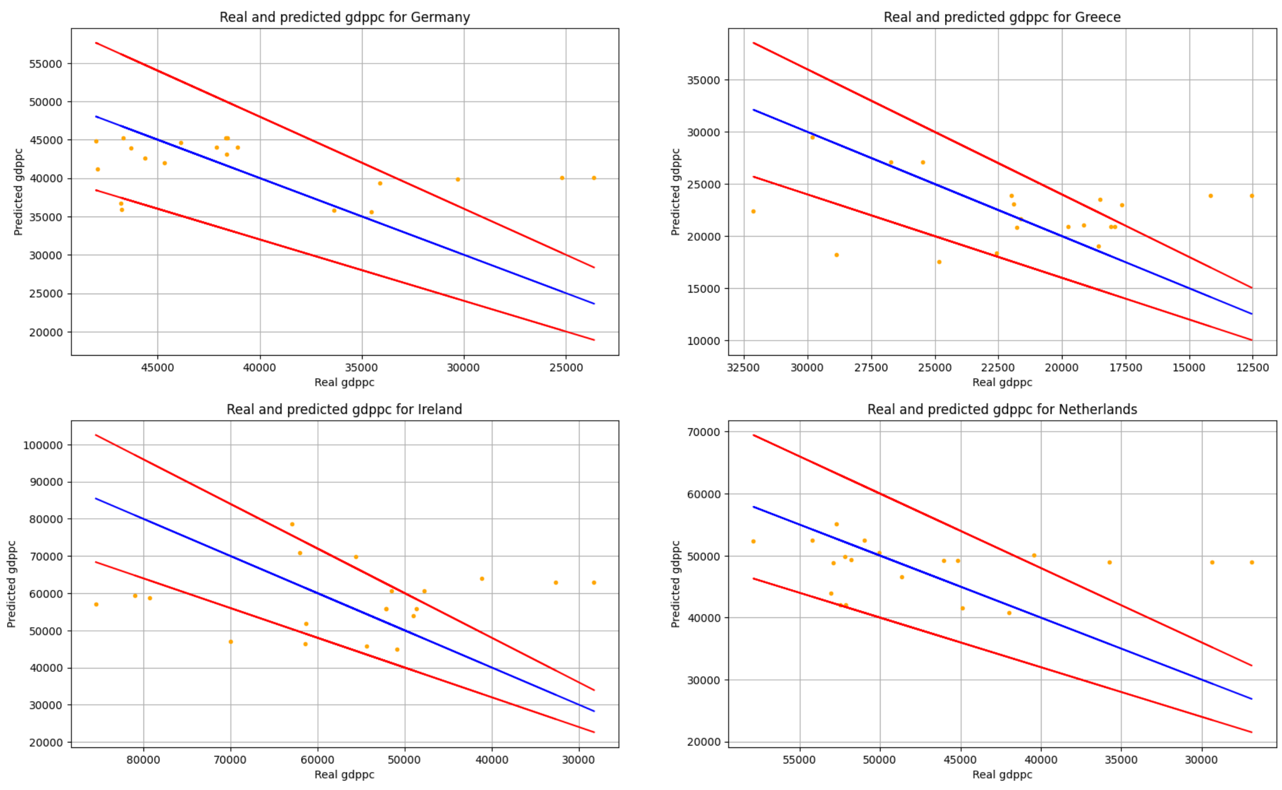
In [ ]:
```
def RandomForest_per__country(country, ESG_pca_df, PCA_columns):
    """
    The Random Forest modelling is done here, firsty determine the number of folds, split the data using cross_validate function,
    and then for every fold a machine learning is done.
    """
    predict_RF = []
    real_RF = []
    RMSE_RF = []
    MAE_RF = []
    K_subset = 5
    K_number = 0
    shuffled = cross_validate_train_test_sets(
        ESG_pca_df[ESG_pca_df['country'] == country], PCA_columns, K_subset)

    for shuffle in range(K_subset):
        X_train, y_train, X_test, y_test = shuffled[shuffle]

        random_forest_model = RandomForestRegressor(
            n_estimators=100, max_features='sqrt', random_state=42)
        random_forest_model.fit(X_train, y_train)
        y_predict_rf = random_forest_model.predict(X_test)
        predict_RF.append(y_predict_rf)
        real_RF.append(y_test)

        K_number += 1
        results(y_test, y_predict_rf, 'Random Forest', country, K_number)
        MSE, RMSE, R2, MAE = get_metric(y_test, y_predict_rf)
        RMSE_RF.append(RMSE)
        MAE_RF.append(MAE)

    RMSE_mean_RF = np.mean(RMSE_RF)
    MAE_mean_RF = np.mean(MAE_RF)

    return predict_RF, real_RF, RMSE_mean_RF, MAE_mean_RF
```

Alpha Lambda plot for the visualization of the predicted gdppc with an alpha of 20%

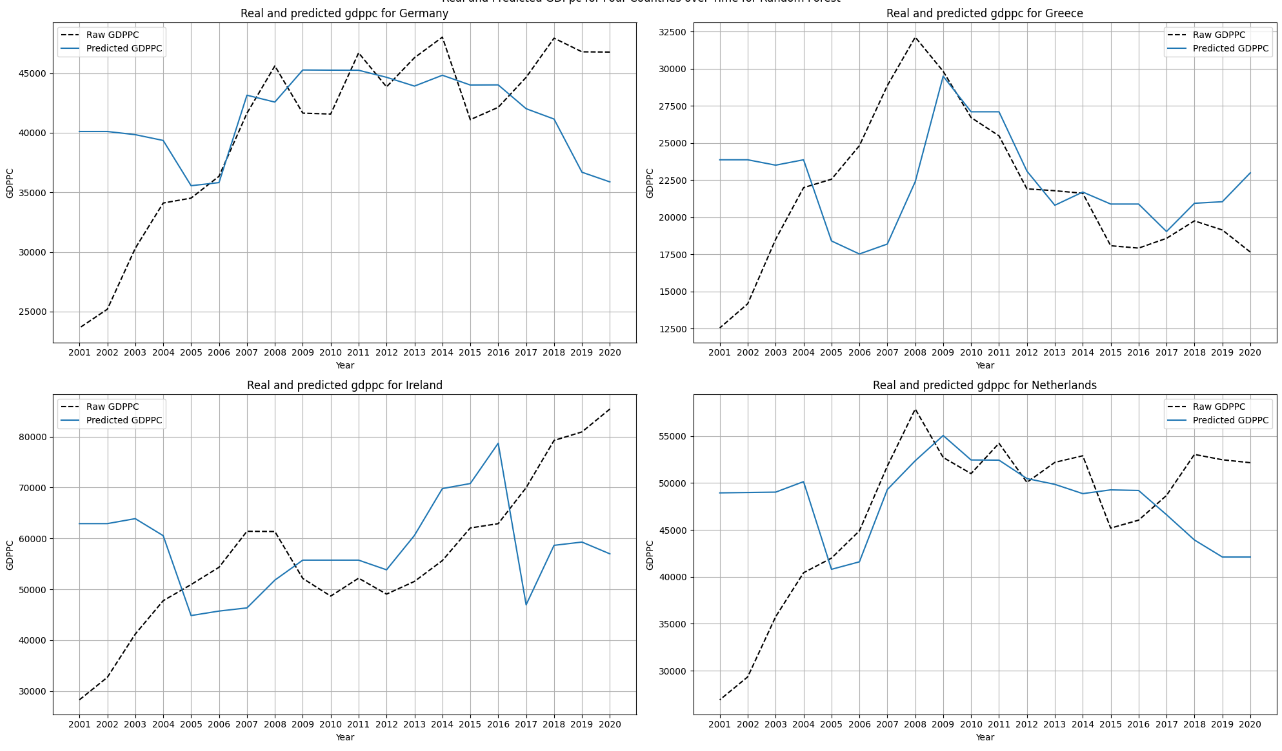
In [ ]:
```
# Making of the alpha lambda plot for a quick overview of the error and the timeseries plot
# Make figure 1
fig1, axs1 = plt.subplots(2, 2, figsize=(20, 12))
fig1.suptitle('Real and Predicted GDPpc for Four Countries for Random Forest')
subplot_index = [(0,0), (0,1), (1,0), (1,1)]
RMSE_mean_RF_all = []
MAE_mean_RF_all = []

# Make figure 2
fig2, axs2 = plt.subplots(2, 2, figsize=(20, 12))
fig2.suptitle('Real and Predicted GDPpc for Four Countries over Time for Random Forest')

for index, country in enumerate(ESG_denoised['country'].unique()):
    predict_RF, real_RF, RMSE_mean_RF, MAE_mean_RF = RandomForest_per__country(country, ESG_pca_df, PCA_columns)
    RMSE_mean_RF_all.append(RMSE_mean_RF)
    MAE_mean_RF_all.append(MAE_mean_RF)

    # For the alpha lambda plot
    alpha = 0.2
    real_RF_np = np.array(real_RF)
    upper_bound = real_RF_np * (1 + alpha)
    lower_bound = real_RF_np * (1 - alpha)

    ax1 = axs1[subplot_index[index]]
    ax1.plot(real_RF, predict_RF, '.', color='orange')
    ax1.plot(real_RF, real_RF, 'b-')
    ax1.plot(real_RF, upper_bound, 'r-')
    ax1.plot(real_RF, lower_bound, 'r-')
    ax1.invert_xaxis()
    ax1.set_title(f'Real and predicted gdppc for {country}')
    ax1.set_xlabel('Real gdppc')
    ax1.set_ylabel('Predicted gdppc')
    ax1.grid(True)

    # For the time series plot
    ax2 = axs2[subplot_index[index]]
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(real_RF[::-1]), 'k--', label = 'Raw GDPPC')
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(predict_RF[::-1]), label = 'Predicted GDPPC')

    dates = ESG[ESG['country'] == country]['date']
    ax2.set_xticks(dates)

    ax2.set_title(f'Real and predicted gdppc for {country}')
    ax2.set_xlabel('Year')
    ax2.set_ylabel('GDPPC')
    ax2.legend()
    ax2.grid(True)

plt.tight_layout()
plt.show()
```

```
Random Forest, Germany, K-fold #1: the RMSE is 12372.577582712107, and R2 is -7.847334685726432, and MAE is 11543.196570788412
Random Forest, Germany, K-fold #2: the RMSE is 1793.9773265543581, and R2 is 0.8320213539684788, and MAE is 1530.253734416976
Random Forest, Germany, K-fold #3: the RMSE is 2712.9089299035677, and R2 is -0.6794391650740021, and MAE is 2390.304434745047
Random Forest, Germany, K-fold #4: the RMSE is 2639.623616451113, and R2 is 0.14851679616705904, and MAE is 2591.278706018087
Random Forest, Germany, K-fold #5: the RMSE is 8271.693799495468, and R2 is -47.535255713460614, and MAE is 7603.088732019807
Random Forest, Greece, K-fold #1: the RMSE is 7914.619722432545, and R2 is -3.564585743216665, and MAE is 6970.240023717058
Random Forest, Greece, K-fold #2: the RMSE is 8350.16265176512, and R2 is -4.146543161193317, and MAE is 7961.232310325053
Random Forest, Greece, K-fold #3: the RMSE is 1037.998027982109, and R2 is 0.8659153713801766, and MAE is 883.0021839027922
Random Forest, Greece, K-fold #4: the RMSE is 2099.9558674720415, and R2 is -0.2870637577846076, and MAE is 1708.1898555423231
Random Forest, Greece, K-fold #5: the RMSE is 2900.965738721264, and R2 is -13.125963731790431, and MAE is 2219.9548966033435
Random Forest, Ireland, K-fold #1: the RMSE is 26404.574216199973, and R2 is -11.295462469051882, and MAE is 25072.57974112424
Random Forest, Ireland, K-fold #2: the RMSE is 10356.707242252838, and R2 is -4.218288347874428, and MAE is 9826.13314661569
Random Forest, Ireland, K-fold #3: the RMSE is 4957.881020289737, and R2 is -7.970290712795444, and MAE is 4747.97824403037
Random Forest, Ireland, K-fold #4: the RMSE is 12330.222826210047, and R2 is -5.941846126449987, and MAE is 11932.831334890754
Random Forest, Ireland, K-fold #5: the RMSE is 23625.740863193194, and R2 is -16.653674185764572, and MAE is 23431.573525027423
Random Forest, Netherlands, K-fold #1: the RMSE is 16901.882171255376, and R2 is -9.072276991975375, and MAE is 16170.065504566615
Random Forest, Netherlands, K-fold #2: the RMSE is 3494.846354353709, and R2 is 0.6795977581069077, and MAE is 3123.890227842596
Random Forest, Netherlands, K-fold #3: the RMSE is 1655.0532999011343, and R2 is -0.07202892977370312, and MAE is 1497.3101306843
Random Forest, Netherlands, K-fold #4: the RMSE is 3477.4848847191092, and R2 is 0.0062670538132483244, and MAE is 3403.7652245794707
Random Forest, Netherlands, K-fold #5: the RMSE is 8598.138826683899, and R2 is -24.22083078047709, and MAE is 7896.38295739014
```

Real and Predicted GDPpc for Four Countries for Random Forest



Real and Predicted GDPpc for Four Countries over Time for Random Forest



Linear Regression

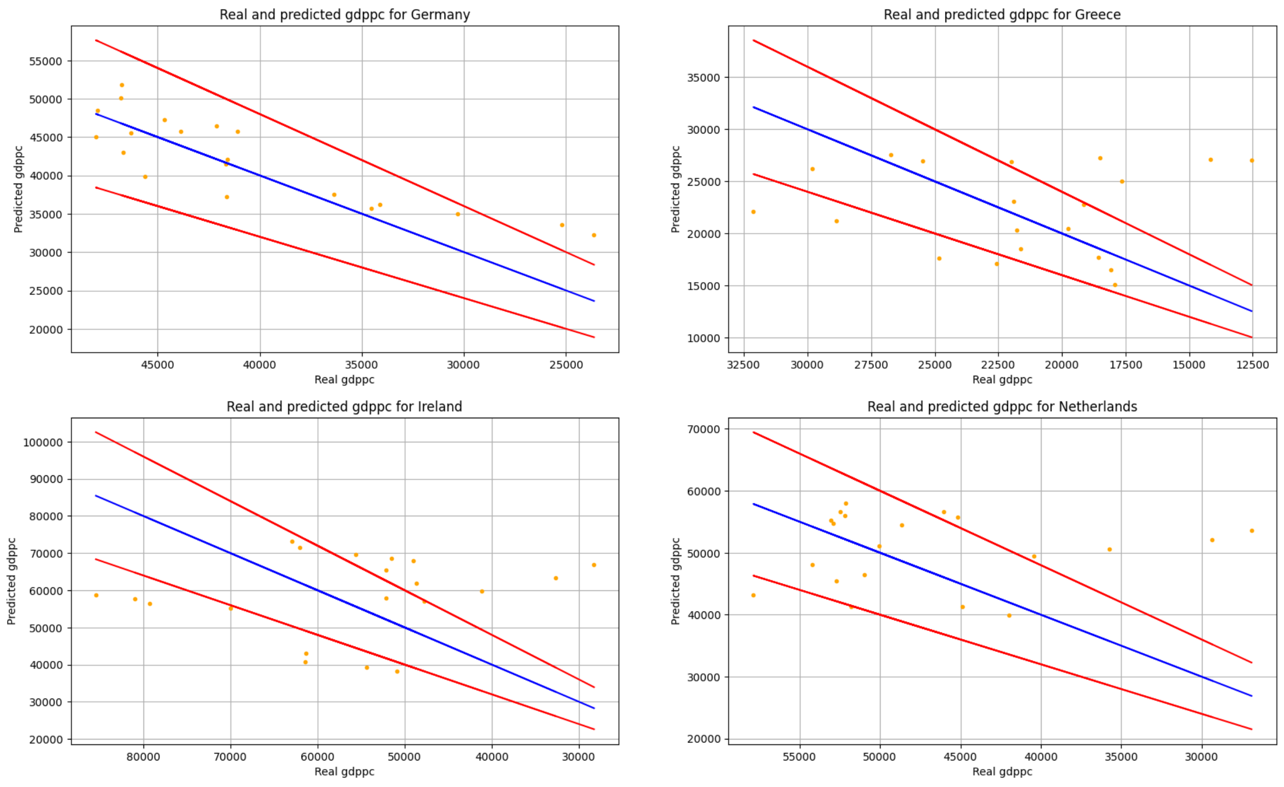
```
In [ ]: def LinearRegression_per__country(country, ESG_pca_df, PCA_columns):
            """
```

```
    The Linear regression modelling is done here, firsty determine the number of folds, split the data using cross_validate function,
    and then for every fold a machine learning is done.
    """
    predict_LR = []
    real_LR = []
    RMSE_LR = []
    MAE_LR = []
    K_subset = 5
    K_number = 0
    shuffled = cross_validate_train_test_sets(
        ESG_pca_df[ESG_pca_df['country'] == country], PCA_columns, K_subset)

    for shuffle in range(K_subset):
        X_train, y_train, X_test, y_test = shuffled[shuffle]

        linear_regression_model = LinearRegression()
        linear_regression_model.fit(X_train, y_train)
        y_predict_LR = linear_regression_model.predict(X_test)
        predict_LR.append(y_predict_LR)
        real_LR.append(y_test)

        K_number += 1
        results(y_test, y_predict_LR, 'Linear Regression', country, K_number)
        MSE, RMSE, R2, MAE = get_metric(y_test, y_predict_LR)

        RMSE_LR.append(RMSE)
        MAE_LR.append(MAE)

    RMSE_mean_LR = np.mean(RMSE_LR)
    MAE_mean_LR = np.mean(MAE_LR)

    return predict_LR, real_LR, RMSE_mean_LR, MAE_mean_LR
```

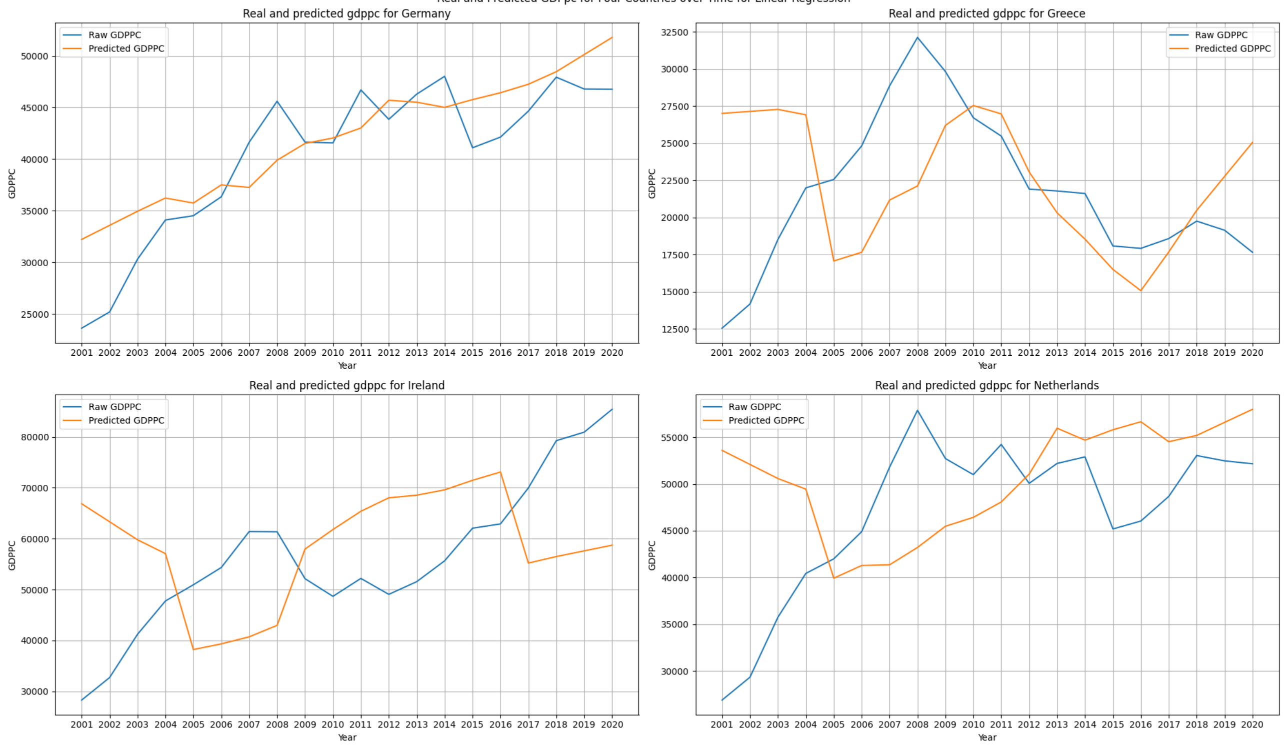
```
In [ ]: # Making of the alpha lambda plot for a quick overview of the error and the timeseries plot
# Make figure 1
fig1, axs1 = plt.subplots(2, 2, figsize=(20, 12))
fig1.suptitle('Real and Predicted GDPpc for Four Countries for Linear Regression')
subplot_index = [(0,0), (0,1), (1,0), (1,1)]
RMSE_mean_LR_all = []
MAE_mean_LR_all = []

# Make figure 2
fig2, axs2 = plt.subplots(2, 2, figsize=(20, 12))
fig2.suptitle('Real and Predicted GDPpc for Four Countries over Time for Linear Regression')

for index, country in enumerate(ESG_denoised['country'].unique()):
    predict_LR, real_LR, RMSE_mean_LR, MAE_mean_LR = LinearRegression_per__country(country, ESG_pca_df, PCA_columns)
    RMSE_mean_LR_all.append(RMSE_mean_LR)
    MAE_mean_LR_all.append(MAE_mean_LR)

    # For the alpha lambda plot
    alpha = 0.2
    real_LR_np = np.array(real_LR)
    upper_bound = real_LR_np * (1 + alpha)
    lower_bound = real_LR_np * (1 - alpha)

    ax1 = axs1[subplot_index[index]]
    ax1.plot(real_LR, predict_LR, '.', color='orange')
    ax1.plot(real_LR, real_LR, 'b-')
    ax1.plot(real_LR, upper_bound, 'r-')
    ax1.plot(real_LR, lower_bound, 'r-')
    ax1.invert_xaxis()
    ax1.set_title(f'Real and predicted gdppc for {country}')
    ax1.set_xlabel('Real gdppc')
    ax1.set_ylabel('Predicted gdppc')
    ax1.grid(True)

    # For the time series plot
    real_LR
    ax2 = axs2[subplot_index[index]]
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(real_LR[::-1]), label = 'Raw GDPPC')
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(predict_LR[::-1]), label = 'Predicted GDPPC')


    dates = ESG[ESG['country'] == country]['date']
    ax2.set_xticks(dates)

    ax2.set_title(f'Real and predicted gdppc for {country}')
    ax2.set_xlabel('Year')
    ax2.set_ylabel('GDPPC')
    ax2.legend()
    ax2.grid(True)

plt.tight_layout()
plt.show()
```

```
Linear Regression, Germany, K-fold #1: the RMSE is 6528.329376295132, and R2 is -1.46318029055855, and MAE is 5940.887436315661
Linear Regression, Germany, K-fold #2: the RMSE is 3695.6324468559933, and R2 is 0.28715071291772154, and MAE is 3117.1850435598917
Linear Regression, Germany, K-fold #3: the RMSE is 2078.030053124255, and R2 is 0.014634176736075366, and MAE is 1535.6834910029738
Linear Regression, Germany, K-fold #4: the RMSE is 3530.399105552621, and R2 is -0.5231395100282399, and MAE is 3187.1088681577585
Linear Regression, Germany, K-fold #5: the RMSE is 3294.91898142065, and R2 is -6.70119116198989, and MAE is 2874.525021866022
Linear Regression, Greece, K-fold #1: the RMSE is 10934.666875388226, and R2 is -7.712692310885535, and MAE is 10277.769141042118
Linear Regression, Greece, K-fold #2: the RMSE is 7754.461507267657, and R2 is -3.4384265578598834, and MAE is 7584.395662016962
Linear Regression, Greece, K-fold #3: the RMSE is 2084.06963844129, and R2 is 0.459481715258686, and MAE is 1773.5021837163495
Linear Regression, Greece, K-fold #4: the RMSE is 2357.2617408036676, and R2 is -0.6217926983686228, and MAE is 2243.279863596099
Linear Regression, Greece, K-fold #5: the RMSE is 4160.553769182848, and R2 is -28.05595575591774, and MAE is 3162.966290008161
Linear Regression, Ireland, K-fold #1: the RMSE is 26723.338687166033, and R2 is -11.594123937366488, and MAE is 24265.788794383676
Linear Regression, Ireland, K-fold #2: the RMSE is 16979.493935516723, and R2 is -13.026010434671775, and MAE is 16701.743864250588
Linear Regression, Ireland, K-fold #3: the RMSE is 13604.47335772855, and R2 is -66.54260185811867, and MAE is 12780.02040704188
Linear Regression, Ireland, K-fold #4: the RMSE is 12997.922022265135, and R2 is -6.714024025504247, and MAE is 12636.791704086414
Linear Regression, Ireland, K-fold #5: the RMSE is 22323.295062888614, and R2 is -14.760893489080768, and MAE is 21889.704086055284
Linear Regression, Netherlands, K-fold #1: the RMSE is 19569.694987441784, and R2 is -12.502855951970705, and MAE is 18325.80994566989
Linear Regression, Netherlands, K-fold #2: the RMSE is 9238.607479990105, and R2 is -1.2389912830376777, and MAE is 7696.5417977191955
Linear Regression, Netherlands, K-fold #3: the RMSE is 5294.19417930327, and R2 is -9.969384730461563, and MAE is 4737.932819915999
Linear Regression, Netherlands, K-fold #4: the RMSE is 7789.484283933274, and R2 is -3.9860496358167374, and MAE is 6695.925861384847
Linear Regression, Netherlands, K-fold #5: the RMSE is 4736.238961191989, and R2 is -6.6527529660058455, and MAE is 4485.697787081832
```

Real and Predicted GDPpc for Four Countries for Linear Regression



Real and Predicted GDPpc for Four Countries over Time for Linear Regression



MLP Neural Network

For the MLP, the sizes of each layer have to be chosen. This is done by doing a hyperparameter search and looking at the bias and variance. The hidden layer configuration of 32, 32, 64 had the highest training score and lowest cross-validation score.

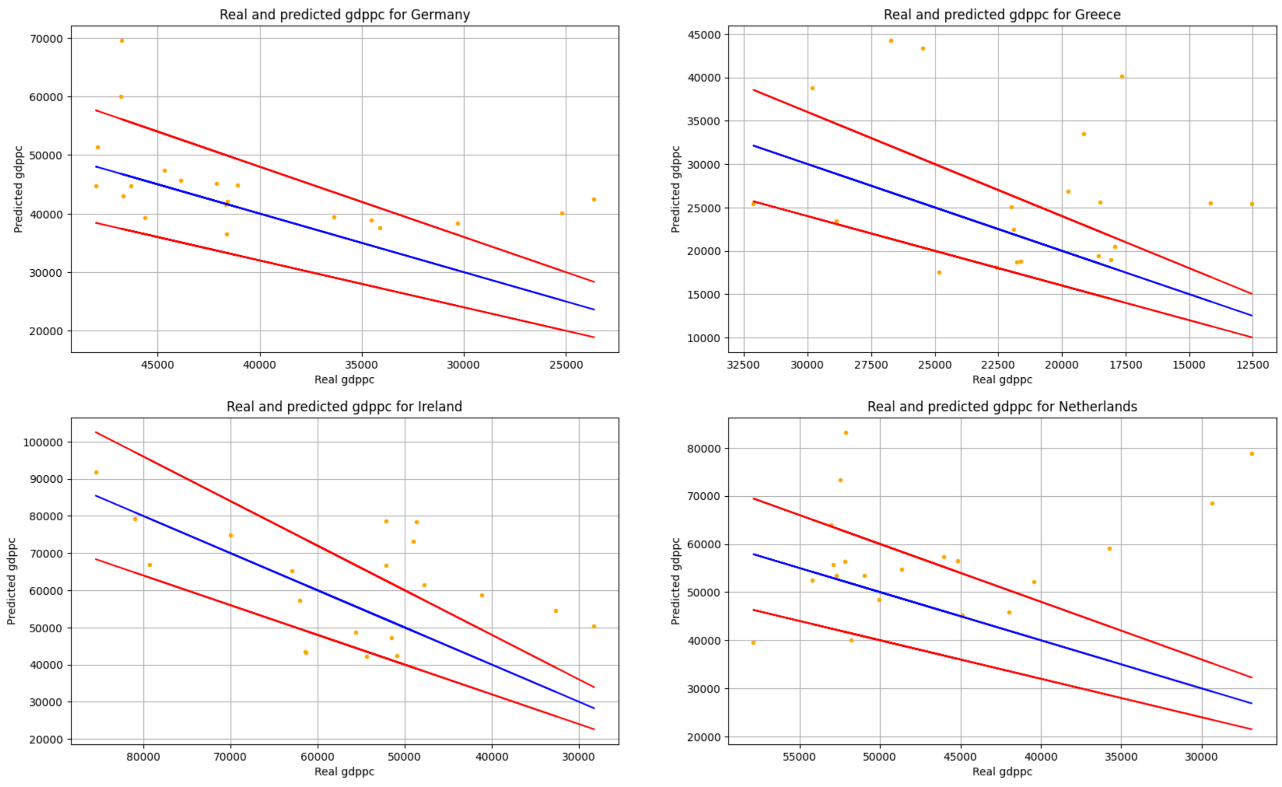
```
In [ ]: def MLP_per__country(country, ESG_pca_df, PCA_columns):
            """
            The MLP modelling is done here, firsty determine the number of folds, split the data using cross_validate function,
            and then for every fold a machine learning is done. The y_predict is found via the x_test and then comapared to the original output (real_mlp)
            """
            predict_MLP = []
            real_MLP = []
            RMSE_MLP = []
            MAE_MLP = []
            K_subset = 5
            K_number = 0
            shuffled = cross_validate_train_test_sets(
                ESG_pca_df[ESG_pca_df['country'] == country], PCA_columns, K_subset)

            for shuffle in range(K_subset):
                X_train, y_train, X_test, y_test = shuffled[shuffle]

                MLP_model = MLPRegressor(hidden_layer_sizes=(32, 32, 64),
                                         activation='relu', solver='adam', max_iter= 5000)
```

```
        MLP_model.fit(X_train, y_train)
        y_predict_MLP = MLP_model.predict(X_test)
        predict_MLP.append(y_predict_MLP)
        real_MLP.append(y_test)

        K_number += 1
        results(y_test, y_predict_MLP, 'MLP neural network', country, K_number)
        MSE, RMSE, R2, MAE = get_metric(y_test, y_predict_MLP)
        RMSE_MLP.append(RMSE)
        MAE_MLP.append(MAE)

    RMSE_mean_MLP = np.mean(RMSE_MLP)
    MAE_mean_MLP = np.mean(MAE_MLP)

    return predict_MLP, real_MLP, RMSE_mean_MLP, MAE_mean_MLP
```

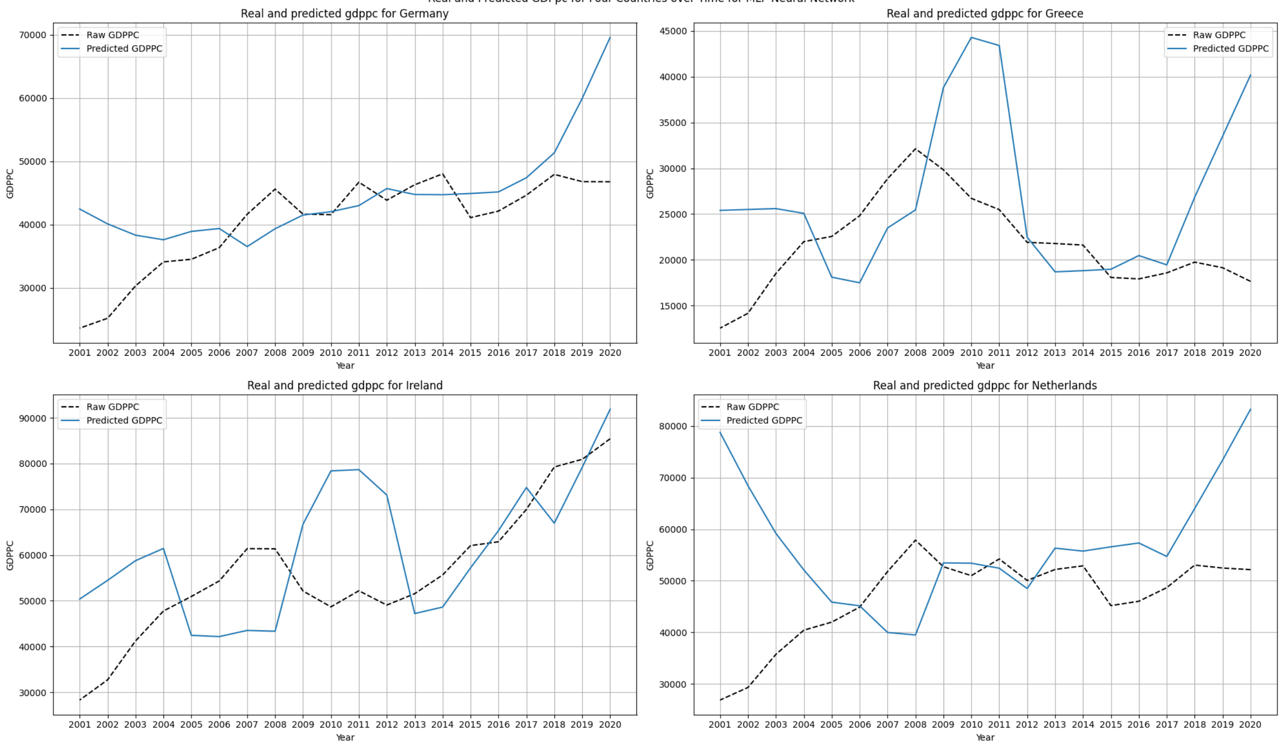
In [ ]:
```
# Making of the alpha lambda plot for a quick overview of the error and the timeseries plot
# Make figure 1
fig1, axs1 = plt.subplots(2, 2, figsize=(20, 12))
fig1.suptitle('Real and Predicted GDPpc for Four Countries for MLP Neural Network')
subplot_index = [(0,0), (0,1), (1,0), (1,1)]
RMSE_mean_MLP_all = []
MAE_mean_MLP_all = []

# Mak figure 2
fig2, axs2 = plt.subplots(2, 2, figsize=(20, 12))
fig2.suptitle('Real and Predicted GDPpc for Four Countries over Time for MLP Neural Network')

for index, country in enumerate(ESG_denoised['country'].unique()):
    predict_MLP, real_MLP, RMSE_mean_MLP, MAE_mean_MLP = MLP_per__country(country, ESG_pca_df, PCA_columns)
    RMSE_mean_MLP_all.append(RMSE_mean_MLP)
    MAE_mean_MLP_all.append(MAE_mean_MLP)

    # For the alpha lambda plot
    alpha = 0.2
    real_MLP_np = np.array(real_MLP)
    upper_bound = real_MLP_np * (1 + alpha)
    lower_bound = real_MLP_np * (1 - alpha)

    ax1 = axs1[subplot_index[index]]
    ax1.plot(real_MLP, predict_MLP, '.', color='orange')
    ax1.plot(real_MLP, real_MLP, 'b-')
    ax1.plot(real_MLP, upper_bound, 'r-')
    ax1.plot(real_MLP, lower_bound, 'r-')
    ax1.invert_xaxis()
    ax1.set_title(f'Real and predicted gdppc for {country}')
    ax1.set_xlabel('Real gdppc')
    ax1.set_ylabel('Predicted gdppc')
    ax1.grid(True)

    # For the time series plot
    ax2 = axs2[subplot_index[index]]
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(real_MLP[::-1]), 'k--', label = 'Raw GDPPC')
    ax2.plot(ESG[ESG['country'] == country]['date'],
             np.concatenate(predict_MLP[::-1]), label = 'Predicted GDPPC')

    dates = ESG[ESG['country'] == country]['date']
    ax2.set_xticks(dates)

    ax2.set_title(f'Real and predicted gdppc for {country}')
    ax2.set_xlabel('Year')
    ax2.set_ylabel('GDPPC')
    ax2.legend()
    ax2.grid(True)

plt.tight_layout()
plt.show()
```

```
MLP neural network, Germany, K-fold #1: the RMSE is 12787.11917713996, and R2 is -8.4501240776973, and MAE is 11319.609689416666
MLP neural network, Germany, K-fold #2: the RMSE is 4846.980030477148, and R2 is -0.22620374317616987, and MAE is 4702.624227249133
MLP neural network, Germany, K-fold #3: the RMSE is 2080.1474235956607, and R2 is 0.01262511297290958, and MAE is 1536.29613591817
MLP neural network, Germany, K-fold #4: the RMSE is 3039.0837264534366, and R2 is -0.12869708914524924, and MAE is 2918.888917430264
MLP neural network, Germany, K-fold #5: the RMSE is 13351.871480719694, and R2 is -125.45986388965939, and MAE is 10534.289625216403
MLP neural network, Greece, K-fold #1: the RMSE is 9395.58859130581, and R2 is -5.4326400056167925, and MAE is 8584.978941008743
MLP neural network, Greece, K-fold #2: the RMSE is 6056.688866902391, and R2 is -1.7076733966321012, and MAE is 5953.046915472462
MLP neural network, Greece, K-fold #3: the RMSE is 13323.013294965538, and R2 is -21.0897529681392, and MAE is 11252.61484710646
MLP neural network, Greece, K-fold #4: the RMSE is 2482.4532961327295, and R2 is -0.7986302738223656, and MAE is 2331.8627086356446
MLP neural network, Greece, K-fold #5: the RMSE is 13813.182770306199, and R2 is -319.2730561695818, and MAE is 11202.77095378827
MLP neural network, Ireland, K-fold #1: the RMSE is 19082.30464340634, and R2 is -5.421672911498801, and MAE is 18770.327746409956
MLP neural network, Ireland, K-fold #2: the RMSE is 14703.980676511104, and R2 is -9.518517162151284, and MAE is 14142.51245331711
MLP neural network, Ireland, K-fold #3: the RMSE is 24384.469741333938, and R2 is -215.99061433251876, and MAE is 23724.896983686016
MLP neural network, Ireland, K-fold #4: the RMSE is 4946.974897577305, and R2 is -0.1174099803314923, and MAE is 4668.118048298991
MLP neural network, Ireland, K-fold #5: the RMSE is 7389.856652306036, and R2 is -0.7271746808266992, and MAE is 6318.202776324673
MLP neural network, Netherlands, K-fold #1: the RMSE is 35012.152851909814, and R2 is -42.221034006819814, and MAE is 31515.633957325455
MLP neural network, Netherlands, K-fold #2: the RMSE is 11091.014265745845, and R2 is -2.226873193328126, and MAE is 8576.989030237466
MLP neural network, Netherlands, K-fold #3: the RMSE is 1734.5193328026066, and R2 is -0.17744553191662282, and MAE is 1627.988950134868
MLP neural network, Netherlands, K-fold #4: the RMSE is 8400.524493675175, and R2 is -4.798985083236875, and MAE is 7414.290902626748
MLP neural network, Netherlands, K-fold #5: the RMSE is 19742.174171762832, and R2 is -131.96594986141895, and MAE is 17245.03510396216
```

Real and Predicted GDPpc for Four Countries for MLP Neural Network



Real and Predicted GDPpc for Four Countries over Time for MLP Neural Network



4: Interim Results

This chapter will elaborate on the results of the three machine learning models to predict the GDP based on the ESG indicators together. The models have predicted the GDP by implementing the k-fold cross-validation. In every iteration, the test set data shifted four years. For example, in the first fold, the years 2001, 2002, 2003, and 2004 were used as test data. In the next fold, the years 2005, 2006, 2007, and 2008 etc. The mean of the metrics are generated over the five folds to get the results underneath.

It can be seen through the data of the above table that the Random Forest performs overall the best. It depends strongly per country. However, to evaluate the machine learning models in order to look at the ESG pillar separately, the Random Forest has been chosen. The model is also decently interpretable and has a fast modeling time. In the next chapter, the ESG pillars will be evaluated separately per country to predict the last 4 years of the dataset.

In [ ]:

```
# DataFrame made to visualize the results
countries = ['Germany', 'Greece', 'Ireland', 'Netherlands']

df_ML_results = pd.DataFrame({
    'Random Forest - MAE': MAE_mean_RF_all,
    'Random Forest - RMSE': RMSE_mean_RF_all,
    'Linear Regression - MAE': MAE_mean_LR_all,
    'Linear Regression - RMSE': RMSE_mean_LR_all,
    'Neural Network - MAE': MAE_mean_MLP_all,
    'Neural Network - RMSE': RMSE_mean_MLP_all
}, index=countries)

averages = df_ML_results.mean(axis=0)
```

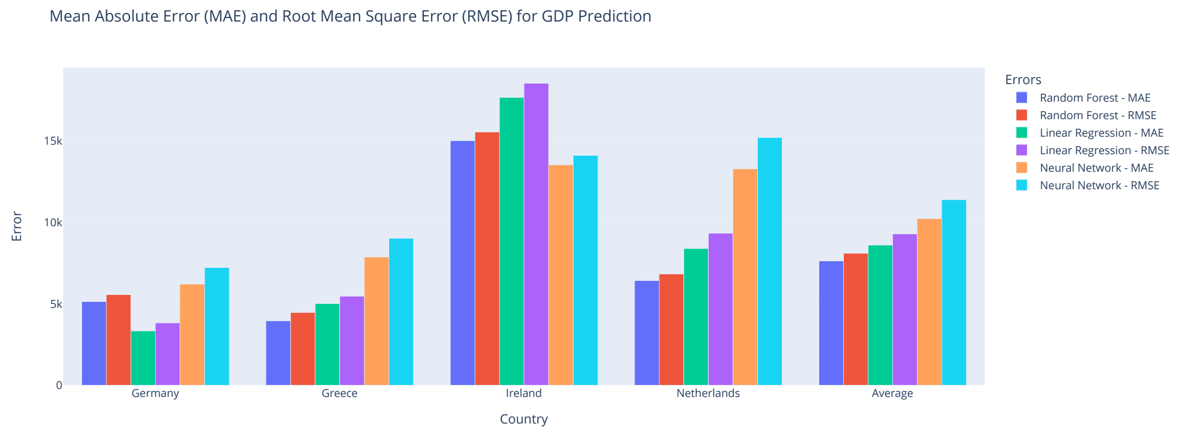
```
df_ML_results.loc['Average'] = averages

df_ML_results
```

Out[ ]:

	Random Forest - MAE	Random Forest - RMSE	Linear Regression - MAE	Linear Regression - RMSE	Neural Network - MAE	Neural Network - RMSE
Germany	5131.624436	5558.156251	3331.077972	3825.461993	6202.341719	7221.040368
Greece	3948.523854	4460.740402	5008.382628	5458.202706	7865.054873	9014.185364
Ireland	15002.219198	15535.025234	17654.809771	18525.704613	13524.811602	14101.517322
Netherlands	6418.282809	6825.481107	8388.381642	9325.643978	13275.987589	15196.077023
Average	7625.162574	8094.850748	8595.663003	9283.753323	10217.048946	11383.205019

In [ ]:

```
# Make a bar plot
fig = px.bar(df_ML_results, x=df_ML_results.index, y=df_ML_results.columns,
             title='Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for GDP Prediction', width = 1300, barmode='group')
fig.update_layout(xaxis_title='Country', yaxis_title='Error', legend_title = 'Errors')

fig.show()
```



5: Analysis per ESG pillar

The Random Forest will be used to further deep-dive into the interrelation between the ESG pillars (or categories) and GDP. Each pillar will be evaluated by implementing a Random Forest separately per pillar and per country.

In [ ]:

```
# For a easier search method a dict is made
indicator_groups = {
    'social': indicators_social,
    'environment': indicators_env,
    'economy': indicators_economy
}

PCA_total = []
# Per country separately, and per inidicator group, a PCA is done
for country in ESG_denoised['country'].unique():
    country_data = ESG_denoised[ESG_denoised['country'] == country]
    merged_data = country_data[['country', 'date', 'gdppc']].copy()
    merged_data = merged_data.reset_index(drop=True)

    for group, indicators in indicator_groups.items():
        country_data_group = country_data[indicators]

        scaler_standard = StandardScaler()
        country_data_scaled = scaler_standard.fit_transform(country_data_group)

        pca = PCA(n_components=2).fit(country_data_scaled)
        country_pca = pca.transform(country_data_scaled)

        PCA_percountry = pd.DataFrame(data=country_pca, columns=[f'pca1_{group}', f'pca2_{group}'])

        merged_data = pd.concat([merged_data, PCA_percountry], axis=1)
    PCA_total.append(merged_data)

# Merge the PCA's per country in a dataframe
ESG_pca_df_perESG = pd.concat(PCA_total, ignore_index=True)
ESG_pca_df_perESG.head()
```

Out[ ]:

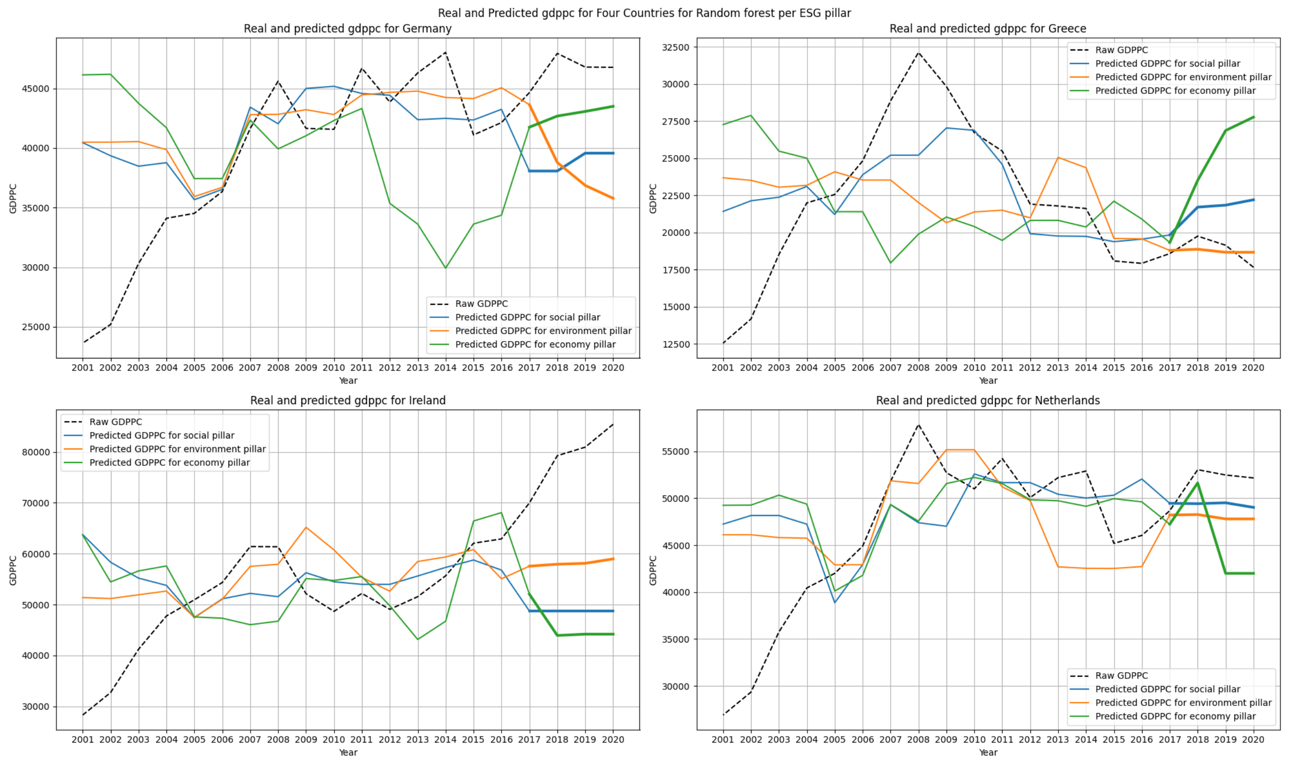
	country	date	gdppc	pca1_social	pca2_social	pca1_environment	pca2_environment	pca1_economy	pca2_economy
0	Germany	2020	46772.825351	-3.030834	-0.621470	3.629551	0.511276	4.303858	-0.109303
1	Germany	2019	46793.686762	-2.447899	-0.695303	2.867539	0.289289	2.739721	0.041283
2	Germany	2018	47939.278288	-1.864963	-0.769136	2.105528	0.067302	1.175584	0.191869
3	Germany	2017	44652.589172	-1.346002	-0.784865	1.501242	-0.117893	-0.446947	0.335658
4	Germany	2016	42136.120791	-1.175840	-0.620578	1.193572	-0.128253	-0.732920	0.461872

Random Forest for ESG pillars separately

In [ ]:

```
def RandomForest_per__country_ESG(country, ESG_pca_df, PCA_columns):
    predict_RF = []
    real_RF = []
    RMSE_RF = []
    MAE_RF = []
    K_subset = 5
    shuffled = cross_validate_train_test_sets(
          ESG_pca_df[ESG_pca_df['country'] == country], PCA_columns, K_subset)

    for shuffle in range(K_subset):
            X_train, y_train, X_test, y_test = shuffled[shuffle]

            random_forest_model = RandomForestRegressor(
                  n_estimators=100, max_features='sqrt', random_state=42)
            random_forest_model.fit(X_train, y_train)
            y_predict_rf = random_forest_model.predict(X_test)
            predict_RF.append(y_predict_rf)
            real_RF.append(y_test)

            MSE, RMSE, R2, MAE = get_metric(y_test, y_predict_rf)
            RMSE_RF.append(RMSE)
            MAE_RF.append(MAE)
    RMSE_mean_RF = np.mean(RMSE_RF)
    MAE_mean_RF = np.mean(MAE_RF)

    return predict_RF, real_RF, RMSE_mean_RF, MAE_mean_RF
```

```
In [ ]: # List per each indicator and column in PCA DataFrame
pca_columns_social = ['pca1_social', 'pca2_social']
pca_columns_environment = ['pca1_environment', 'pca2_environment']
pca_columns_economy = ['pca1_economy', 'pca2_economy']

RMSE_mean_social = []
MAE_mean_social = []
RMSE_mean_environment = []
MAE_mean_environment = []
RMSE_mean_economy = []
MAE_mean_economy = []

# Making of the figure
fig, axs = plt.subplots(2, 2, figsize=(20, 12))
fig.suptitle('Real and Predicted gdppc for Four Countries for Random forest per ESG pillar')
subplot_index = [(0,0), (0,1), (1,0), (1,1)]
for index, country in enumerate(ESG_denoised['country'].unique()):
    # For each indicator, do a separata Random Forest
    predict_RF_ESG_social, real_ESG, RMSE_mean_ESG_social, MAE_mean_ESG_social = \
        RandomForest_per__country_ESG(country, ESG_pca_df_perESG, pca_columns_social)
    predict_RF_ESG_environment, real_ESG, RMSE_mean_ESG_environment, MAE_mean_ESG_environment = \
        RandomForest_per__country_ESG(country, ESG_pca_df_perESG, pca_columns_environment)
    predict_RF_ESG_economy, real_ESG, RMSE_mean_ESG_economy, MAE_mean_ESG_economy = \
        RandomForest_per__country_ESG(country, ESG_pca_df_perESG, pca_columns_economy)

    RMSE_mean_social.append(RMSE_mean_ESG_social)
    MAE_mean_social.append(MAE_mean_ESG_social)
    RMSE_mean_environment.append(RMSE_mean_ESG_environment)
    MAE_mean_environment.append(MAE_mean_ESG_environment)
    RMSE_mean_economy.append(RMSE_mean_ESG_economy)
    MAE_mean_economy.append(MAE_mean_ESG_economy)

    # Plot the data found
    ax = axs[subplot_index[index]]
    ax.plot(ESG[ESG['country'] == country]['date'],
            np.concatenate(real_ESG[::-1]), 'k--', label = 'Raw GDPPC')
    ax.plot(ESG[ESG['country'] == country]['date'],
            np.concatenate(predict_RF_ESG_social[::-1]), label='Predicted GDPPC for social pillar')
    ax.plot(ESG[ESG['country'] == country]['date'],
            np.concatenate(predict_RF_ESG_environment[::-1]), label='Predicted GDPPC for environment pillar')
    ax.plot(ESG[ESG['country'] == country]['date'],
            np.concatenate(predict_RF_ESG_economy[::-1]), label='Predicted GDPPC for economy pillar')

    # Highlight the last years
    last_four_years = ESG[ESG['country'] == country]['date'][:4]

    ax.plot(last_four_years, np.concatenate(predict_RF_ESG_social[::-1])[:4], color = '#1F77B4', linewidth=3)
    ax.plot(last_four_years, np.concatenate(predict_RF_ESG_environment[::-1])[:4], color = '#FF7F0E', linewidth=3)
    ax.plot(last_four_years, np.concatenate(predict_RF_ESG_economy[::-1])[:4], color = '#2CA02C', linewidth=3)

    dates = ESG[ESG['country'] == country]['date']
    ax.set_xticks(dates)

    ax.set_title(f'Real and predicted gdppc for {country}')
    ax.set_xlabel('Year')
    ax.set_ylabel('GDPPC')
    ax.legend()
    ax.grid(True)

plt.tight_layout()
plt.show()
```



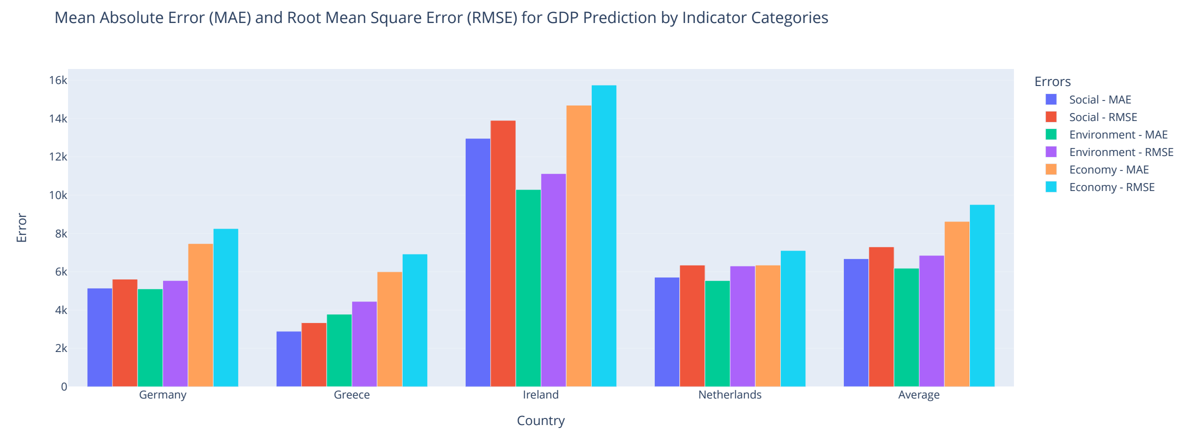
```
In [ ]: # Making of a new DataFrame with the results of the Random Forest per indicator
indexes = ['Germany', 'Greece', 'Ireland', 'Netherlands']

ESG_results = pd.DataFrame({
    'Social - MAE': MAE_mean_social,
    'Social - RMSE': RMSE_mean_social,
    'Environment - MAE': MAE_mean_environment,
    'Environment - RMSE': RMSE_mean_environment,
    'Economy - MAE': MAE_mean_economy,
    'Economy - RMSE': RMSE_mean_economy
}, index=countries)

averages = ESG_results.mean(axis=0)
ESG_results.loc['Average'] = averages
```

```
ESG_results
```

Out[ ]:

	Social - MAE	Social - RMSE	Environment - MAE	Environment - RMSE	Economy - MAE	Economy - RMSE
Germany	5143.222739	5610.301650	5103.640618	5538.916579	7468.858063	8253.009107
Greece	2888.600258	3334.961329	3780.913374	4450.137049	5994.404703	6925.155980
Ireland	12961.607116	13901.486073	10291.054680	11122.013780	14692.891092	15746.925691
Netherlands	5710.361198	6344.686258	5535.311199	6300.268752	6346.200377	7106.339376
Average	6675.947828	7297.858827	6177.729968	6852.834040	8625.588559	9507.857539

In [ ]:
```
# Bar plot made for a better and easier visualization
fig = px.bar(ESG_results, x=ESG_results.index, y=ESG_results.columns,
             title='Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for GDP Prediction by Indicator Categories', width = 1300, barmode='group')
fig.update_layout(xaxis_title='Country', yaxis_title='Error', legend_title = 'Errors')

fig.show()
```



6: Conclusion and Discussion

In our research project, we research the relationship between a country's Gross Domestic Product (GDP) and a diverse set of indicators, including Social, Economic (Governance), and Environmental indicators, also known as ESG indicators. Our goal was to not only understand the factors influencing GDP but also to develop predictive machine learning models that could accurately predict a country's economic performance expressed in GDP based on these indicators. We analyzed the economies of the Netherlands, Germany, Greece, and Ireland, each representing unique economic characteristics within Europe. In this analysis, we have drawn several conclusions:

The Influence of ESG Indicators on GDP

Our research revealed that Environmental, Social, and Governance (ESG) indicators have a significant impact on a country's GDP per capita. Specifically, we found that ESG indicators correlate with GDP, expressing the importance of sustainability, social well-being, and efficient governance in economic trends. **Social Indicators** such as health expenditure, immunization rates, life expectancy, and child mortality were shown to play an important role in influencing a country's GDP. These factors underscore the importance of investing in healthcare and maintaining a healthy population for economic development. The strong correlation of these social indicators with GDP further supports this conclusion. For example, the social indicators have a very strong correlation with the GDP of the Netherlands. Also, the models where we only used the social indicators to predict GDP, scored better than only using the economic indicators. **Economic Indicators** like export and import data and freight efficiency were linked to a country's economic performance, emphasizing the importance of trade and logistics in economic growth. However, the correlation between these economic indicators and GDP is weaker compared to social and environmental indicators. Also, when predicting the GDP of countries, we would not recommend only using economic indicators based on our results. **Environmental Indicators**, including renewable energy consumption, energy efficiency, and greenhouse gas emissions, give the importance of sustainability in economic trends. Countries committed to eco-friendly practices tended to have higher GDP because environmental indicators correlate strongly with GDP. For instance, the strong correlation between renewable energy consumption and GDP in Germany and Ireland underscores the importance of environmentally sustainable energy sources. When using the environmental indicators in the machine learning model, the lowest Mean Absolute Error and Root Mean Square Error occur.

Machine Learning Models for GDP Prediction

We applied different machine learning models, including Random Forest, Linear Regression, and MLP Neural Network, to predict GDP based on all indicators combined. Among the models, the Random Forest delivered the most accurate average GDP predictions across the four countries, as indicated by lower Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) compared to the other models.

Variations Among Countries

Our analysis showed variations in prediction errors among the machine learning models when applied to each of the four countries. These differences could be held accountable to the different economic profiles and characteristics of each country. Notably, Greece and Germany had the lowest prediction errors, indicating that the indicators chosen effectively predicted the GDP for these countries. Also, the best Machine Learning model was different among the chosen countries. In the case of Germany, the Linear Regression model was much more accurate than the Random Forest model, so this is also country-specific.

The Impact of Indicators and ML Models

In conclusion, our study provides insights for governments and policymakers, emphasizing the significance of ESG indicators in predicting a country's economic trend. Moreover, the prediction of machine learning models, particularly the Random Forest model, offers a promising tool for forecasting GDP, which can be used to inform economic decisions and policies. On the other hand, the errors found after predicting the GDP with these methods are still quite high, and a GDP needs to have an accurate prediction. We recommend also exploring other methods of predicting GDP and increasing the number of indicators used in the model to improve accuracy.

6: References

- Europa.eu (2023) https://ec.europa.eu/eurostat/databrowser/explore/all/all_themes?lang=en&display=list&sort=category
- Sklearn.ensemble.RandomForestRegressor. (2023). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html%E2%80%8C
- Sustainable Development Goals - Data Catalog. (2018). Worldbank.org. https://datacatalog.worldbank.org/search/dataset/0037918