OSEMN-Obtain

The data process is possible thanks to both open power systems data, a free to access database on renewable energy, and the National Aeronautics and Space Administration, which has kept detailed weather data for the past several decades.

Comparing energy production and weather data, both individually and separately, we took a look at any patterns present, what they could mean, and how we can use them to an advantage, which becomes much more feasible when comparing the two datasets across from each other. We will be comparing these two datasets across three separate countries to find any distinct advantages or disadvantages those countries may hold: Germany, Italy, and France.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [2]:
```
weather = pd.read_csv("weather_data.csv")
```

In [3]:
```
weather.head()
```

Out[3]:

	utc_timestamp	AT_temperature	AT_radiation_direct_horizontal	AT_radiation_diffuse_horizontal	BE_
0	1980-01-01T00:00:00Z	-3.640	0.0	0.0	
1	1980-01-01T01:00:00Z	-3.803	0.0	0.0	
2	1980-01-01T02:00:00Z	-3.969	0.0	0.0	
3	1980-01-01T03:00:00Z	-4.076	0.0	0.0	
4	1980-01-01T04:00:00Z	-4.248	0.0	0.0	

5 rows × 85 columns

In [4]:
```
weather.tail()
```

Out[4]:

	utc_timestamp	AT_temperature	AT_radiation_direct_horizontal	AT_radiation_diffuse_horizontal
350635	2019-12-31T19:00:00Z	-1.386	0.0	0.0
350636	2019-12-31T20:00:00Z	-1.661	0.0	0.0
350637	2019-12-31T21:00:00Z	-1.986	0.0	0.0
350638	2019-12-31T22:00:00Z	-2.184	0.0	0.0
350639	2019-12-31T23:00:00Z	-2.271	0.0	0.0

5 rows × 85 columns

In [5]:
```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350640 entries, 0 to 350639
Data columns (total 85 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   utc_timestamp                   350640 non-null  object 
 1   AT_temperature                  350640 non-null  float64
 2   AT_radiation_direct_horizontal  350640 non-null  float64
 3   AT_radiation_diffuse_horizontal 350640 non-null  float64
 4   BE_temperature                  350640 non-null  float64
 5   BE_radiation_direct_horizontal  350640 non-null  float64
 6   BE_radiation_diffuse_horizontal 350640 non-null  float64
 7   BG_temperature                  350640 non-null  float64
 8   BG_radiation_direct_horizontal  350640 non-null  float64
 9   BG_radiation_diffuse_horizontal 350640 non-null  float64
 10  CH_temperature                  350640 non-null  float64
 11  CH_radiation_direct_horizontal  350640 non-null  float64
 12  CH_radiation_diffuse_horizontal 350640 non-null  float64
 13  CZ_temperature                  350640 non-null  float64
 14  CZ_radiation_direct_horizontal  350640 non-null  float64
 15  CZ_radiation_diffuse_horizontal 350640 non-null  float64
 16  DE_temperature                  350640 non-null  float64
 17  DE_radiation_direct_horizontal  350640 non-null  float64
 18  DE_radiation_diffuse_horizontal 350640 non-null  float64
 19  DK_temperature                  350640 non-null  float64
 20  DK_radiation_direct_horizontal  350640 non-null  float64
 21  DK_radiation_diffuse_horizontal 350640 non-null  float64
 22  EE_temperature                  350640 non-null  float64
 23  EE_radiation_direct_horizontal  350640 non-null  float64
 24  EE_radiation_diffuse_horizontal 350640 non-null  float64
 25  ES_temperature                  350640 non-null  float64
 26  ES_radiation_direct_horizontal  350640 non-null  float64
 27  ES_radiation_diffuse_horizontal 350640 non-null  float64
 28  FI_temperature                  350640 non-null  float64
 29  FI_radiation_direct_horizontal  350640 non-null  float64
 30  FI_radiation_diffuse_horizontal 350640 non-null  float64
 31  FR_temperature                  350640 non-null  float64
 32  FR_radiation_direct_horizontal  350640 non-null  float64
 33  FR_radiation_diffuse_horizontal 350640 non-null  float64
 34  GB_temperature                  350640 non-null  float64
 35  GB_radiation_direct_horizontal  350640 non-null  float64
 36  GB_radiation_diffuse_horizontal 350640 non-null  float64
 37  GR_temperature                  350640 non-null  float64
 38  GR_radiation_direct_horizontal  350640 non-null  float64
 39  GR_radiation_diffuse_horizontal 350640 non-null  float64
 40  HR_temperature                  350640 non-null  float64
 41  HR_radiation_direct_horizontal  350640 non-null  float64
 42  HR_radiation_diffuse_horizontal 350640 non-null  float64
 43  HU_temperature                  350640 non-null  float64
 44  HU_radiation_direct_horizontal  350640 non-null  float64
 45  HU_radiation_diffuse_horizontal 350640 non-null  float64
 46  IE_temperature                  350640 non-null  float64
 47  IE_radiation_direct_horizontal  350640 non-null  float64
 48  IE_radiation_diffuse_horizontal 350640 non-null  float64
 49  IT_temperature                  350640 non-null  float64
```

```
 50  IT_radiation_direct_horizontal   350640 non-null  float64
 51  IT_radiation_diffuse_horizontal  350640 non-null  float64
 52  LT_temperature                   350640 non-null  float64
 53  LT_radiation_direct_horizontal   350640 non-null  float64
 54  LT_radiation_diffuse_horizontal  350640 non-null  float64
 55  LU_temperature                   350640 non-null  float64
 56  LU_radiation_direct_horizontal   350640 non-null  float64
 57  LU_radiation_diffuse_horizontal  350640 non-null  float64
 58  LV_temperature                   350640 non-null  float64
 59  LV_radiation_direct_horizontal   350640 non-null  float64
 60  LV_radiation_diffuse_horizontal  350640 non-null  float64
 61  NL_temperature                   350640 non-null  float64
 62  NL_radiation_direct_horizontal   350640 non-null  float64
 63  NL_radiation_diffuse_horizontal  350640 non-null  float64
 64  NO_temperature                   350640 non-null  float64
 65  NO_radiation_direct_horizontal   350640 non-null  float64
 66  NO_radiation_diffuse_horizontal  350640 non-null  float64
 67  PL_temperature                   350640 non-null  float64
 68  PL_radiation_direct_horizontal   350640 non-null  float64
 69  PL_radiation_diffuse_horizontal  350640 non-null  float64
 70  PT_temperature                   350640 non-null  float64
 71  PT_radiation_direct_horizontal   350640 non-null  float64
 72  PT_radiation_diffuse_horizontal  350640 non-null  float64
 73  RO_temperature                   350640 non-null  float64
 74  RO_radiation_direct_horizontal   350640 non-null  float64
 75  RO_radiation_diffuse_horizontal  350640 non-null  float64
 76  SE_temperature                   350640 non-null  float64
 77  SE_radiation_direct_horizontal   350640 non-null  float64
 78  SE_radiation_diffuse_horizontal  350640 non-null  float64
 79  SI_temperature                   350640 non-null  float64
 80  SI_radiation_direct_horizontal   350640 non-null  float64
 81  SI_radiation_diffuse_horizontal  350640 non-null  float64
 82  SK_temperature                   350640 non-null  float64
 83  SK_radiation_direct_horizontal   350640 non-null  float64
 84  SK_radiation_diffuse_horizontal  350640 non-null  float64
dtypes: float64(84), object(1)
memory usage: 227.4+ MB
```

In [6]:
```
timeseries = pd.read_csv("time_series_60min_singleindex.csv")
```

In [7]:
```
timeseries.head()
```

Out[7]:

	utc_timestamp	cet_cest_timestamp	AT_load_actual_entsoe_transparency	AT_load_forecast_entsoe_
0	2014-12-31T23:00:00Z	2015-01-01T00:00:00+0100	NaN	
1	2015-01-01T00:00:00Z	2015-01-01T01:00:00+0100	5946.0	
2	2015-01-01T01:00:00Z	2015-01-01T02:00:00+0100	5726.0	
3	2015-01-01T02:00:00Z	2015-01-01T03:00:00+0100	5347.0	
4	2015-01-01T03:00:00Z	2015-01-01T04:00:00+0100	5249.0	

5 rows × 300 columns

In [8]:
```
timeseries.tail()
```

Out[8]:

	utc_timestamp	cet_cest_timestamp	AT_load_actual_entsoe_transparency	AT_load_forecast_en
50396	2020-09-30T19:00:00Z	2020-09-30T21:00:00+0200	6661.0	
50397	2020-09-30T20:00:00Z	2020-09-30T22:00:00+0200	6336.0	
50398	2020-09-30T21:00:00Z	2020-09-30T23:00:00+0200	5932.0	
50399	2020-09-30T22:00:00Z	2020-10-01T00:00:00+0200	5628.0	
50400	2020-09-30T23:00:00Z	2020-10-01T01:00:00+0200	5395.0	

5 rows × 300 columns

In [9]:
```
timeseries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50401 entries, 0 to 50400
Columns: 300 entries, utc_timestamp to UA_load_forecast_entsoe_transparen
cy
dtypes: float64(298), object(2)
memory usage: 115.4+ MB
```

OSEMN-Scrub

Let's take a look at our columns, we can see that there are quite a few countries in there! While that's certainly useful, we can probably set them aside for now.

The data cleaning process will involve dropping irrelevant columns, then splitting and recombining data by country (from our three coutnries), for later analysis.

In [10]:
```
for col in timeseries.columns:
    print(col)
```

```
utc_timestamp
cet_cest_timestamp
AT_load_actual_entsoe_transparency
AT_load_forecast_entsoe_transparency
AT_price_day_ahead
AT_solar_generation_actual
AT_wind_onshore_generation_actual
BE_load_actual_entsoe_transparency
BE_load_forecast_entsoe_transparency
BE_solar_generation_actual
BE_wind_generation_actual
BE_wind_offshore_generation_actual
BE_wind_onshore_generation_actual
BG_load_actual_entsoe_transparency
BG_load_forecast_entsoe_transparency
BG_solar_generation_actual
BG_wind_onshore_generation_actual
CH_load_actual_entsoe_transparency
CH_load_forecast_entsoe_transparency
```

We'll be looking at three key countries, to explore differences across different locations, temperatures, and wind strength (the latter two should vary seasonally). The three chosen countries will be Germany, Italy, and France.

In [11]:
```
Germany_RE = pd.read_csv("time_series_60min_singleindex.csv",
                        usecols=(lambda s: s.startswith('utc') | s.startswi
                        parse_dates=[0], index_col=0)
Italy_RE = pd.read_csv("time_series_60min_singleindex.csv",
                        usecols=(lambda s: s.startswith('utc') | s.startswi
                        parse_dates=[0], index_col=0)
France_RE = pd.read_csv("time_series_60min_singleindex.csv",
                        usecols=(lambda s: s.startswith('utc') | s.startswi
                        parse_dates=[0], index_col=0)
```

In [12]:
```
Germany_RE.tail()
```

Out[12]:

utc_timestamp	DE_load_actual_entsoe_transparency	DE_load_forecast_entsoe_transparency	DE_solar
2020-09-30 19:00:00+00:00	57559.0	56708.0	
2020-09-30 20:00:00+00:00	54108.0	53270.0	
2020-09-30 21:00:00+00:00	49845.0	49239.0	
2020-09-30 22:00:00+00:00	46886.0	46620.0	
2020-09-30 23:00:00+00:00	45461.0	44986.0	

5 rows × 41 columns

In [13]:
```
Italy_RE.tail()
```

Out[13]:

utc_timestamp	IT_load_actual_entsoe_transparency	IT_load_forecast_entsoe_transparency	IT_solar_ge
2020-09-30 19:00:00+00:00	35217.0	37048.0	
2020-09-30 20:00:00+00:00	31537.0	33255.0	
2020-09-30 21:00:00+00:00	28730.0	29573.0	
2020-09-30 22:00:00+00:00	26269.0	26251.0	
2020-09-30 23:00:00+00:00	NaN	NaN	

5 rows × 52 columns

In [14]:
```
France_RE.tail()
```

Out[14]:

	FR_load_actual_entsoe_transparency	FR_load_forecast_entsoe_transparency	FR_solar_
utc_timestamp			
2020-09-30 19:00:00+00:00	48210.0	50050.0	
2020-09-30 20:00:00+00:00	48210.0	48150.0	
2020-09-30 21:00:00+00:00	48058.0	50600.0	
2020-09-30 22:00:00+00:00	44869.0	45350.0	
2020-09-30 23:00:00+00:00	NaN	NaN	

Let's compare a specific year for our countries, they are all fairly close to each other, so we'll be sure to watch out for any variations in production patterns, rather than amouns.

In [15]:
```
France_RE_2018 = France_RE.loc[France_RE.index.year == 2018, :]
France_RE_2019 = France_RE.loc[France_RE.index.year == 2019, :]
Italy_RE_2018 = Italy_RE.loc[Italy_RE.index.year == 2018, :]
Italy_RE_2019 = Italy_RE.loc[Italy_RE.index.year == 2019, :]
Germany_RE_2018 = Germany_RE.loc[Germany_RE.index.year == 2018, :]
Germany_RE_2019 = Germany_RE.loc[Germany_RE.index.year == 2019, :]
```

In [16]:
```
France_RE_2018.head()
```

Out[16]:

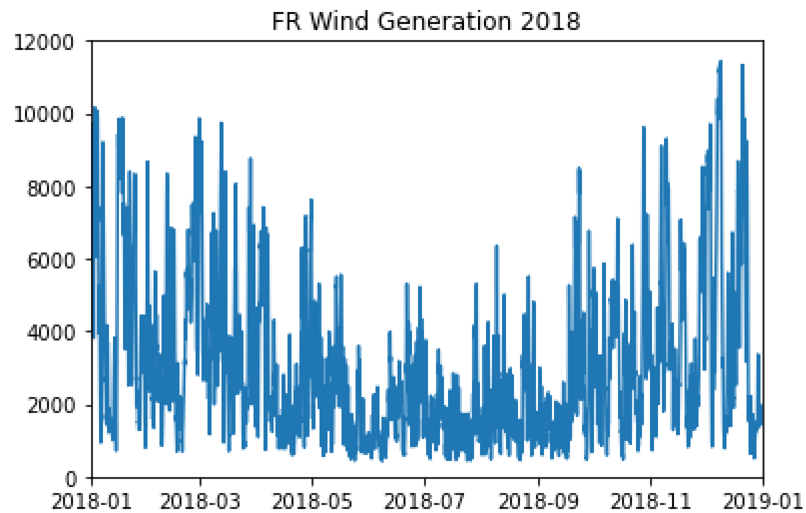
	FR_load_actual_entsoe_transparency	FR_load_forecast_entsoe_transparency	FR_solar_
utc_timestamp			
2018-01-01 00:00:00+00:00	56036.0	54300.0	
2018-01-01 01:00:00+00:00	54494.0	53600.0	
2018-01-01 02:00:00+00:00	51574.0	50000.0	
2018-01-01 03:00:00+00:00	49370.0	47100.0	
2018-01-01 04:00:00+00:00	49000.0	45850.0	

OSEMN-Explore

Exploring the data, we can begin to see patterns that we can use to make some recommendations. We can see seasonal patterns in both the wind and solar production, which supports the idea that there are reliable predictors for renewable energy production.

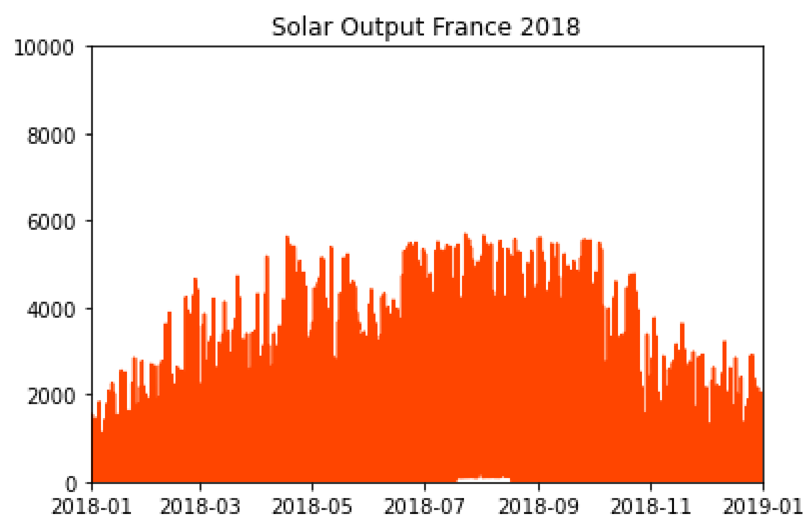
In [17]:
```
plt.plot(France_RE_2018.index, France_RE_2018['FR_wind_onshore_generation_a
plt.title('FR Wind Generation 2018')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 12000)
```

Out[17]:
```
(0.0, 12000.0)
```



In [18]:
```
plt.plot(France_RE_2018.index, France_RE_2018['FR_solar_generation_actual']
plt.title('Solar Output France 2018')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 10000)
```

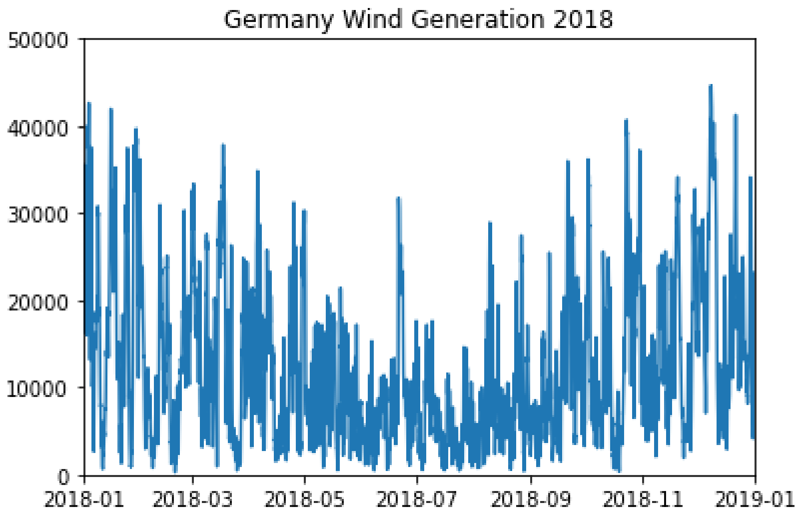
Out[18]:
```
(0.0, 10000.0)
```



Unsurprisingly, it looks like there is a dip in wind production during summer months, balanced out by more sunlight, which of course positively correlates with solar energy production. Let's see if there is a comparable drop/bump in the other countries we will be examining.

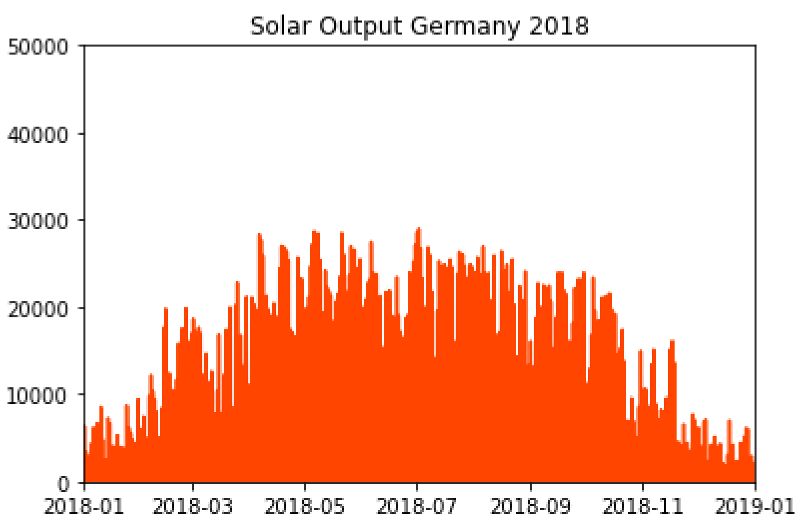
In [19]:
```
plt.plot(Germany_RE_2018.index, Germany_RE_2018['DE_wind_generation_actual'
plt.title('Germany Wind Generation 2018')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 50000)
```

Out[19]: (0.0, 50000.0)



In [20]:
```
plt.plot(Germany_RE_2018.index, Germany_RE_2018['DE_solar_generation_actual
plt.title('Solar Output Germany 2018')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 50000)
```

Out[20]: (0.0, 50000.0)

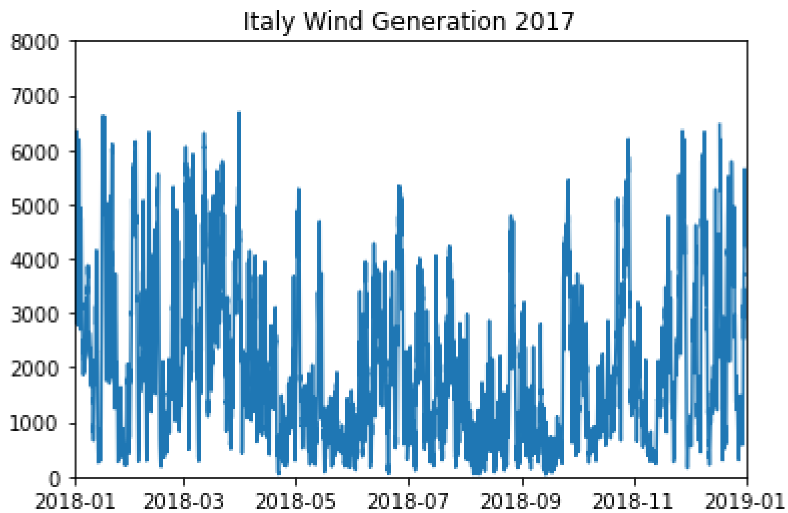


Very similar patterns, albeit with much higher production, which is also unsuprising considering Germany's extensive renewable energy laws.

Since we'll be comparing all three countries, let's check Italy out!

In [21]:
```
plt.plot(Italy_RE_2018.index, Italy_RE_2018['IT_wind_onshore_generation_act
plt.title('Italy Wind Generation 2017')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 8000)
```

Out[21]:
```
(0.0, 8000.0)
```

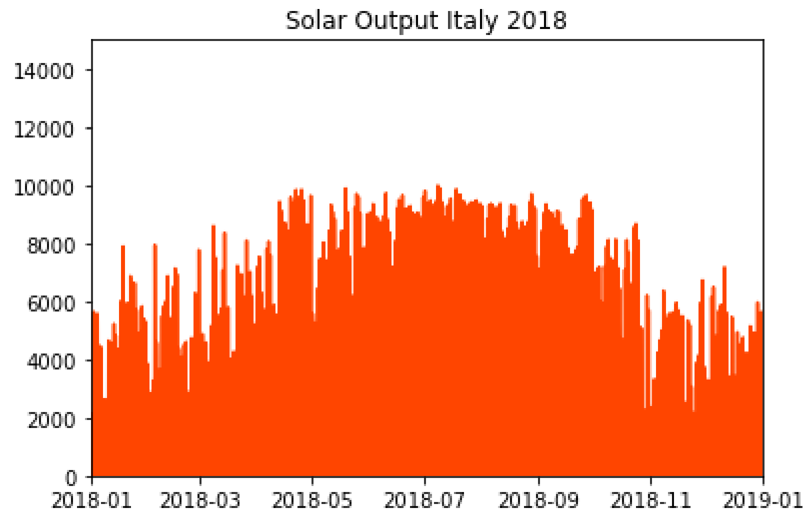


Less Seasonal variation than our previous two countries, but this can be accounted for if we consider Italy's exposure to the Mediterranean Ocean as a pensinsula. While France and Germany are by no means land locked, they have less terriory bordering the ocean, which would increase seasonal efficacy for wind turbines.

Let's perform the same exploration for solar power, and see how it stacks up.

In [22]:
```
plt.plot(Italy_RE_2018.index, Italy_RE_2018['IT_solar_generation_actual'],
plt.title('Solar Output Italy 2018')
plt.xlim(pd.Timestamp('2018-01-01'), pd.Timestamp('2019-01-01'))
plt.ylim(0, 15000)
```

Out[22]: (0.0, 15000.0)



Much less variation across the board for Italy, probably due to comparable reasons as the wind variation. Both wind and solar energy production is aided due to its' proximity to the ocean, wind due to surface water's friction with the air, creating additional wind to power turbines, and solar due to the reflective properties of the ocean's surface, creating additional sunlight. Of course Italy is the southernmost of our three countries, and as such receives more sunlight year round.

This is a very promising and fascinating situation. Italy seems to have to best location for consistent energy output, yet still lags behind Germany in producion of both Solar and Wind energy. Additional funding could greatly impact their economy, and the comfort of living of its' citizens.

In [23]:
```
Italy_production = Italy_RE_2018[['IT_wind_onshore_generation_actual', 'IT_
Germany_production = Germany_RE_2018[['DE_wind_generation_actual', 'DE_sola
France_production = France_RE_2018[['FR_wind_onshore_generation_actual', 'F
```

In [24]:
```
Italy_production.head()
France_production.head()
Germany_production.head()
```

Out[24]:

	DE_wind_generation_actual	DE_solar_generation_actual
utc_timestamp		
2018-01-01 00:00:00+00:00	33105.0	0.0
2018-01-01 01:00:00+00:00	33868.0	0.0
2018-01-01 02:00:00+00:00	34778.0	0.0
2018-01-01 03:00:00+00:00	34741.0	0.0
2018-01-01 04:00:00+00:00	35345.0	0.0

In [25]:
```
weather = weather.iloc[333120:]
```

In [26]:
```
weather.tail()
```

Out[26]:

	utc_timestamp	AT_temperature	AT_radiation_direct_horizontal	AT_radiation_diffuse_horizonta
350635	2019-12-31T19:00:00Z	-1.386	0.0	0.0
350636	2019-12-31T20:00:00Z	-1.661	0.0	0.0
350637	2019-12-31T21:00:00Z	-1.986	0.0	0.0
350638	2019-12-31T22:00:00Z	-2.184	0.0	0.0
350639	2019-12-31T23:00:00Z	-2.271	0.0	0.0

5 rows × 85 columns

Our weather data goes quite a bit further back than necessary, let's limit our rows by dropping any outside of the year 2018. Doing some math, we can see that the rows are seperated by the hour, and with 365 days in a year (2018 was not a leap year) we can calculate exactly where our cutoff needs to be.

In [27]:
```
weather = weather.iloc[:-8760]
```

In [28]:
```
weather.tail()
```

Out[28]:

	utc_timestamp	AT_temperature	AT_radiation_direct_horizontal	AT_radiation_diffuse_horizonta
341875	2018-12-31T19:00:00Z	-1.205	0.0	0.0
341876	2018-12-31T20:00:00Z	-1.408	0.0	0.0
341877	2018-12-31T21:00:00Z	-1.614	0.0	0.0
341878	2018-12-31T22:00:00Z	-1.688	0.0	0.0
341879	2018-12-31T23:00:00Z	-1.666	0.0	0.0

5 rows × 85 columns

In [29]:
```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 333120 to 341879
Data columns (total 85 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   utc_timestamp                      8760 non-null   object 
 1   AT_temperature                     8760 non-null   float64
 2   AT_radiation_direct_horizontal     8760 non-null   float64
 3   AT_radiation_diffuse_horizontal    8760 non-null   float64
 4   BE_temperature                     8760 non-null   float64
 5   BE_radiation_direct_horizontal     8760 non-null   float64
 6   BE_radiation_diffuse_horizontal    8760 non-null   float64
 7   BG_temperature                     8760 non-null   float64
 8   BG_radiation_direct_horizontal     8760 non-null   float64
 9   BG_radiation_diffuse_horizontal    8760 non-null   float64
 10  CH_temperature                     8760 non-null   float64
 11  CH_radiation_direct_horizontal     8760 non-null   float64
 12  CH_radiation_diffuse_horizontal    8760 non-null   float64
 13  CZ_temperature                     8760 non-null   float64
 14  CZ_radiation_direct_horizontal     8760 non-null   float64
 15  CZ_radiation_diffuse_horizontal    8760 non-null   float64
 16  DE_temperature                     8760 non-null   float64
 17  DE_radiation_direct_horizontal     8760 non-null   float64
 18  DE_radiation_diffuse_horizontal    8760 non-null   float64
 19  DK_temperature                     8760 non-null   float64
 20  DK_radiation_direct_horizontal     8760 non-null   float64
 21  DK_radiation_diffuse_horizontal    8760 non-null   float64
 22  EE_temperature                     8760 non-null   float64
 23  EE_radiation_direct_horizontal     8760 non-null   float64
 24  EE_radiation_diffuse_horizontal    8760 non-null   float64
 25  ES_temperature                     8760 non-null   float64
 26  ES_radiation_direct_horizontal     8760 non-null   float64
 27  ES_radiation_diffuse_horizontal    8760 non-null   float64
 28  FI_temperature                     8760 non-null   float64
 29  FI_radiation_direct_horizontal     8760 non-null   float64
 30  FI_radiation_diffuse_horizontal    8760 non-null   float64
 31  FR_temperature                     8760 non-null   float64
 32  FR_radiation_direct_horizontal     8760 non-null   float64
 33  FR_radiation_diffuse_horizontal    8760 non-null   float64
 34  GB_temperature                     8760 non-null   float64
 35  GB_radiation_direct_horizontal     8760 non-null   float64
 36  GB_radiation_diffuse_horizontal    8760 non-null   float64
 37  GR_temperature                     8760 non-null   float64
 38  GR_radiation_direct_horizontal     8760 non-null   float64
 39  GR_radiation_diffuse_horizontal    8760 non-null   float64
 40  HR_temperature                     8760 non-null   float64
 41  HR_radiation_direct_horizontal     8760 non-null   float64
 42  HR_radiation_diffuse_horizontal    8760 non-null   float64
 43  HU_temperature                     8760 non-null   float64
 44  HU_radiation_direct_horizontal     8760 non-null   float64
 45  HU_radiation_diffuse_horizontal    8760 non-null   float64
 46  IE_temperature                     8760 non-null   float64
 47  IE_radiation_direct_horizontal     8760 non-null   float64
 48  IE_radiation_diffuse_horizontal    8760 non-null   float64
 49  IT_temperature                     8760 non-null   float64
```

```
 50  IT_radiation_direct_horizontal   8760 non-null   float64
 51  IT_radiation_diffuse_horizontal  8760 non-null   float64
 52  LT_temperature                   8760 non-null   float64
 53  LT_radiation_direct_horizontal   8760 non-null   float64
 54  LT_radiation_diffuse_horizontal  8760 non-null   float64
 55  LU_temperature                   8760 non-null   float64
 56  LU_radiation_direct_horizontal   8760 non-null   float64
 57  LU_radiation_diffuse_horizontal  8760 non-null   float64
 58  LV_temperature                   8760 non-null   float64
 59  LV_radiation_direct_horizontal   8760 non-null   float64
 60  LV_radiation_diffuse_horizontal  8760 non-null   float64
 61  NL_temperature                   8760 non-null   float64
 62  NL_radiation_direct_horizontal   8760 non-null   float64
 63  NL_radiation_diffuse_horizontal  8760 non-null   float64
 64  NO_temperature                   8760 non-null   float64
 65  NO_radiation_direct_horizontal   8760 non-null   float64
 66  NO_radiation_diffuse_horizontal  8760 non-null   float64
 67  PL_temperature                   8760 non-null   float64
 68  PL_radiation_direct_horizontal   8760 non-null   float64
 69  PL_radiation_diffuse_horizontal  8760 non-null   float64
 70  PT_temperature                   8760 non-null   float64
 71  PT_radiation_direct_horizontal   8760 non-null   float64
 72  PT_radiation_diffuse_horizontal  8760 non-null   float64
 73  RO_temperature                   8760 non-null   float64
 74  RO_radiation_direct_horizontal   8760 non-null   float64
 75  RO_radiation_diffuse_horizontal  8760 non-null   float64
 76  SE_temperature                   8760 non-null   float64
 77  SE_radiation_direct_horizontal   8760 non-null   float64
 78  SE_radiation_diffuse_horizontal  8760 non-null   float64
 79  SI_temperature                   8760 non-null   float64
 80  SI_radiation_direct_horizontal   8760 non-null   float64
 81  SI_radiation_diffuse_horizontal  8760 non-null   float64
 82  SK_temperature                   8760 non-null   float64
 83  SK_radiation_direct_horizontal   8760 non-null   float64
 84  SK_radiation_diffuse_horizontal  8760 non-null   float64
dtypes: float64(84), object(1)
memory usage: 5.7+ MB
```

In [30]:
```
France_weather = weather[['FR_temperature', 'FR_radiation_direct_horizontal
Germany_weather = weather[['DE_temperature', 'DE_radiation_direct_horizonta
Italy_weather = weather[['IT_temperature', 'IT_radiation_direct_horizontal'
```

In [31]:
```
France_weather.head()
Italy_weather.head()
Germany_weather.head()
```

Out[31]:

	DE_temperature	DE_radiation_direct_horizontal	DE_radiation_diffuse_horizontal
333120	6.494	0.0	0.0
333121	6.171	0.0	0.0
333122	5.880	0.0	0.0
333123	5.637	0.0	0.0
333124	5.440	0.0	0.0

Neat! now our data is seperated by country, this will mke calculating linear regression quite a bit easier for us.

In [35]:
```
Germany_weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 333120 to 341879
Data columns (total 3 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   DE_temperature                   8760 non-null   float64
 1   DE_radiation_direct_horizontal   8760 non-null   float64
 2   DE_radiation_diffuse_horizontal  8760 non-null   float64
dtypes: float64(3)
memory usage: 205.4 KB
```

In [36]:
```
pd.concat([Germany_production,Germany_weather], axis=0, ignore_index=True)
```

Out[36]:

	DE_wind_generation_actual	DE_solar_generation_actual	DE_temperature	DE_radiation_direct_l
0	33105.0	0.0	NaN	
1	33868.0	0.0	NaN	
2	34778.0	0.0	NaN	
3	34741.0	0.0	NaN	
4	35345.0	0.0	NaN	
...	...	...	...	
17515	NaN	NaN	4.963	
17516	NaN	NaN	5.005	
17517	NaN	NaN	4.992	
17518	NaN	NaN	4.905	
17519	NaN	NaN	4.843	

17520 rows × 5 columns

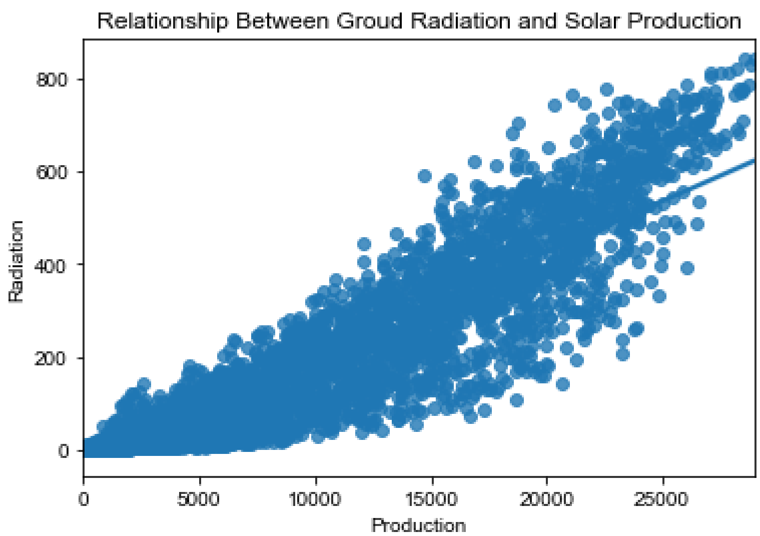
Our two data tables aren't compatible, so we'll have to find some more creative ways of modeling our linear regression.

OSEMN-Model

For our model we tested the relationships between the weather data and our production data, and graphed the most favorable results. After testing it was found that shortwave radiation was the best indicator, so let's check how applicable that is!

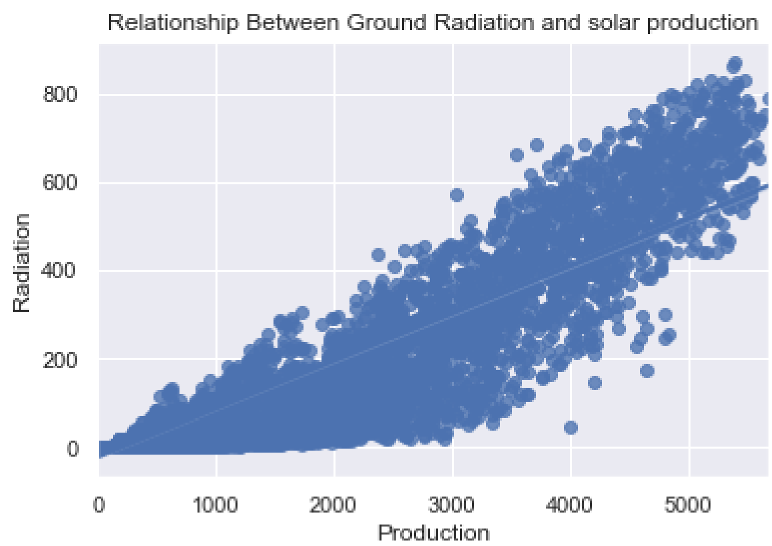
In [37]:
```
Germany_temperature_correlation = sns.regplot(x=Germany_production['DE_sola
sns.set_theme()
plt.title('Relationship Between Groud Radiation and Solar Production')
plt.xlabel('Production')
plt.ylabel('Radiation')
```

Out[37]:
```
Text(0, 0.5, 'Radiation')
```



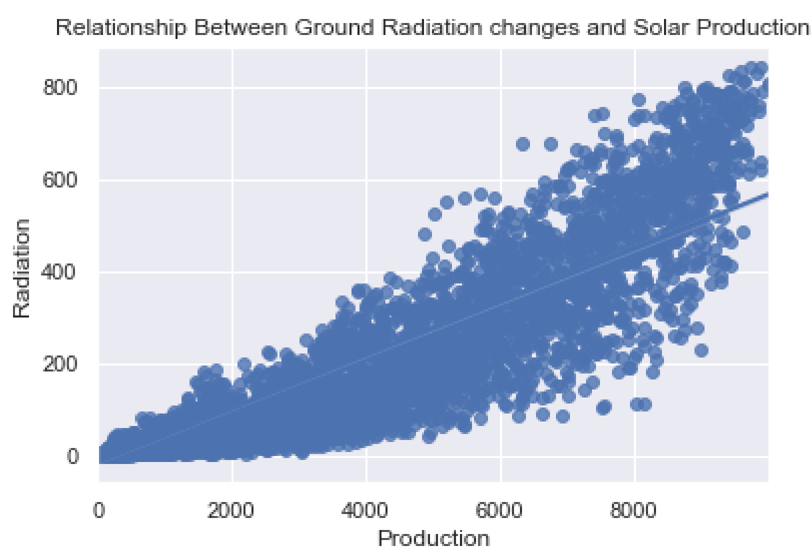
In [38]:
```
France_temperature_correlation = sns.regplot(x=France_production['FR_solar_
sns.set_theme()
plt.title('Relationship Between Ground Radiation and solar production')
plt.xlabel('Production')
plt.ylabel('Radiation')
```

Out[38]: Text(0, 0.5, 'Radiation')



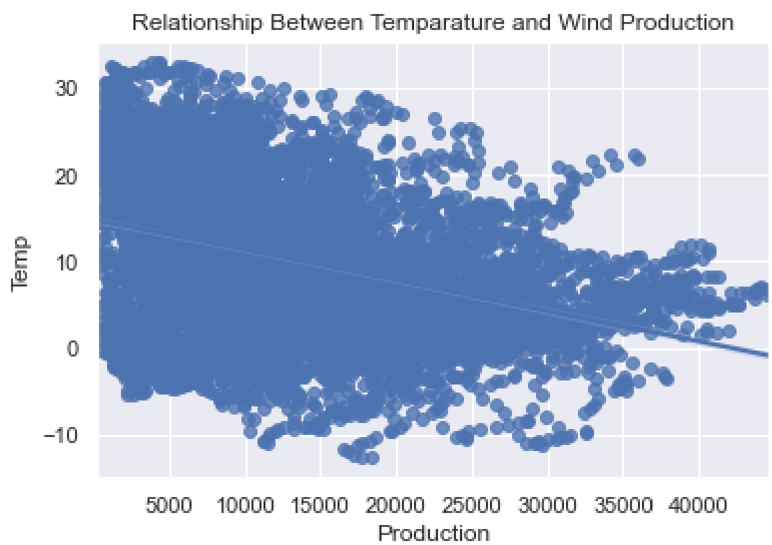
In [39]:
```
Italy_temperature_correlation = sns.regplot(x=Italy_production['IT_solar_ge
sns.set_theme()
plt.title('Relationship Between Ground Radiation changes and Solar Producti
plt.xlabel('Production')
plt.ylabel('Radiation')
```

Out[39]: Text(0, 0.5, 'Radiation')



In [40]:
```
Germany_temperature_correlation = sns.regplot(x=Germany_production['DE_wind
sns.set_theme()
plt.title('Relationship Between Temparature and Wind Production')
plt.xlabel('Production')
plt.ylabel('Temp')
```

Out[40]:
```
Text(0, 0.5, 'Temp')
```



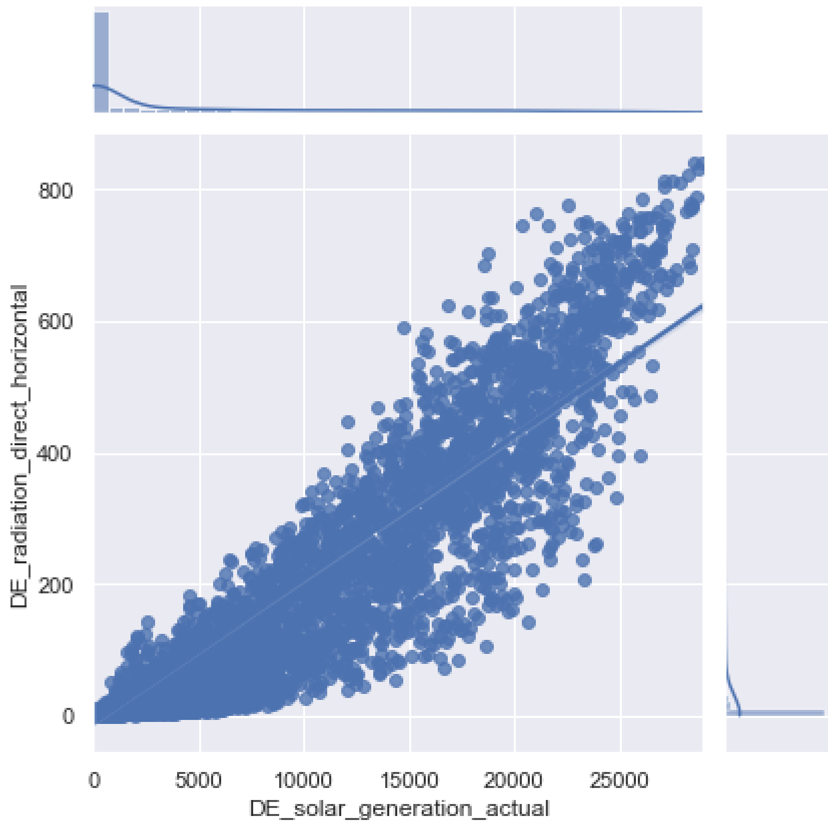
Well it looks like temperature certainly isn't a very reliable for wind, which is surprising due to wind production's peak during colder months. While there is certainly a NEGATIVE correlation (as graphed above), it's not very directional, and probably wouldn't be accurate enough to yield actionable results.

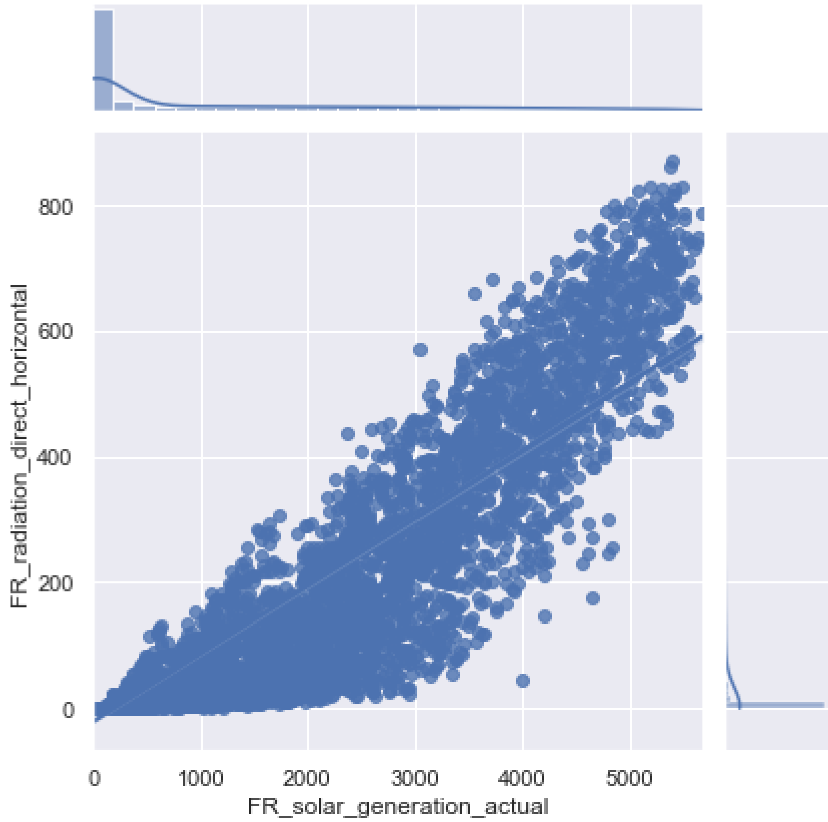
Let's try solar energy, and see where that takes us!

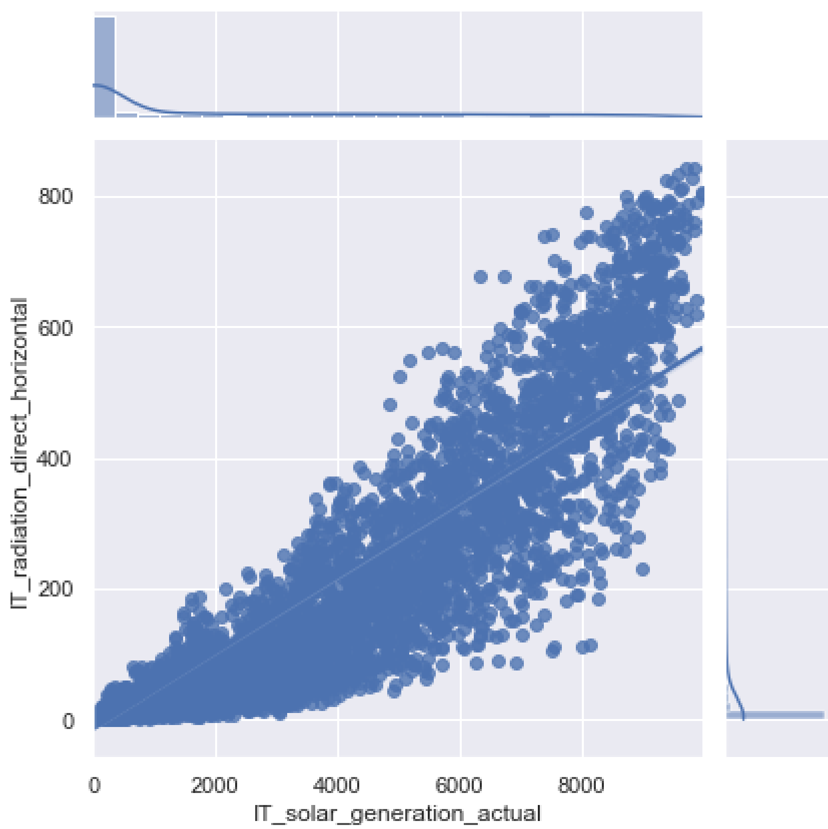
In [41]:
```
Germany_weather.reset_index(drop=True, inplace=True)
Germany_production.reset_index(drop=True, inplace=True)
France_weather.reset_index(drop=True, inplace=True)
France_production.reset_index(drop=True, inplace=True)
Italy_weather.reset_index(drop=True, inplace=True)
Italy_production.reset_index(drop=True, inplace=True)
```

In [42]:
```
sns.jointplot(x=Germany_production['DE_solar_generation_actual'],
y=Germany_weather['DE_radiation_direct_horizontal'], kind='reg')

sns.jointplot(x=France_production['FR_solar_generation_actual'],
y=France_weather['FR_radiation_direct_horizontal'], kind='reg')

sns.jointplot(x=Italy_production['IT_solar_generation_actual'],
y=Italy_weather['IT_radiation_direct_horizontal'], kind='reg')
```

Out[42]: <seaborn.axisgrid.JointGrid at 0x7faf5cedd1f0>




FR_radiation_direct_horizontal
FR_solar_generation_actual
0
200
400
600
800
0
1000
2000
3000
4000
5000


IT_radiation_direct_horizontal
IT_solar_generation_actual
0
200
400
600
800
0
2000
4000
6000
8000

Looks quite a bit more accurate! There is definitely SOME correlation here, let's use linear regression to see if we're right, if our scores for each country are high enough, we can use them to make our predictions!

In [43]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lr = LinearRegression()
```

In [44]:
```
X_solar = Germany_weather[['DE_temperature', 'DE_radiation_direct_horizonta
y_solar = Germany_production['DE_solar_generation_actual']
```

In [45]:
```
scores_solar = cross_val_score(lr, X_solar, y_solar, cv=5)
print(scores_solar, "\naverage =", np.mean(scores_solar))
```

```
[0.93530772 0.9708968  0.94691825 0.94086074 0.8758698 ]
average = 0.9339706614086379
```

In [46]:
```
FR_X_solar = France_weather[['FR_temperature', 'FR_radiation_direct_horizon
FR_y_solar = France_production['FR_solar_generation_actual']
```

In [47]:
```
FR_scores_solar = cross_val_score(lr, FR_X_solar, FR_y_solar, cv=5)
print(FR_scores_solar, "\naverage =", np.mean(FR_scores_solar))
```

```
[0.9233064  0.94767226 0.93783295 0.93191536 0.85084134]
average = 0.9183136612015425
```

In [48]:
```
Italy_production['IT_wind_onshore_generation_actual'].fillna(value=Italy_pr
Italy_production['IT_solar_generation_actual'].fillna(value=Italy_productio
```

```
/Users/lucacaruccio/opt/anaconda3/envs/learn-env/lib/python3.8/site-packa
ges/pandas/core/series.py:4517: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  return super().fillna(
```

In [49]:
```
IT_X_solar = Italy_weather[['IT_temperature', 'IT_radiation_direct_horizont
IT_y_solar = Italy_production['IT_solar_generation_actual']
```

In [50]:
```
IT_scores_solar = cross_val_score(lr, IT_X_solar, IT_y_solar, cv=5)
print(IT_scores_solar, "\naverage =", np.mean(IT_scores_solar))
```

```
[0.93383258 0.94929228 0.96177289 0.95823815 0.91169799]
average = 0.9429667770143929
```

Next we will take a look at energy source production and carbon emissions in the United States, and work on some more advanced models to see if we can use what we know about historical production to help predict carbon emissions.

In [51]:
```
Energy_production = pd.read_csv('MER_T01_02.csv')
```

In [52]:
```
Energy_production.head()
```

Out[52]:

	MSN	YYYYMM	Value	Column_Order	Description	Unit
0	CLPRBUS	194913	11.973882	1	Coal Production	Quadrillion Btu
1	CLPRBUS	195013	14.060135	1	Coal Production	Quadrillion Btu
2	CLPRBUS	195113	14.419325	1	Coal Production	Quadrillion Btu
3	CLPRBUS	195213	12.734313	1	Coal Production	Quadrillion Btu
4	CLPRBUS	195313	12.277746	1	Coal Production	Quadrillion Btu

In [53]:
```
CO2 = pd.read_csv('co2-data.csv')
```

In [54]:
```
CO2.head()
```

Out[54]:

	iso_code	country	year	co2	co2_growth_prct	co2_growth_abs	consumption_co2	trade_cc
0	AFG	Afghanistan	1949	0.015	NaN	NaN	NaN	Na
1	AFG	Afghanistan	1950	0.084	475.000	0.070	NaN	Na
2	AFG	Afghanistan	1951	0.092	8.696	0.007	NaN	Na
3	AFG	Afghanistan	1952	0.092	NaN	NaN	NaN	Na
4	AFG	Afghanistan	1953	0.106	16.000	0.015	NaN	Na

5 rows × 55 columns

In [55]:
```
CO2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23708 entries, 0 to 23707
Data columns (total 55 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   iso_code                              20930 non-null  object 
 1   country                               23708 non-null  object 
 2   year                                  23708 non-null  int64  
 3   co2                                   23170 non-null  float64
 4   co2_growth_prct                       21910 non-null  float64
 5   co2_growth_abs                        22017 non-null  float64
 6   consumption_co2                       3350 non-null   float64
 7   trade_co2                             3318 non-null   float64
 8   trade_co2_share                       3318 non-null   float64
 9   co2_per_capita                        22383 non-null  float64
 10  consumption_co2_per_capita            3350 non-null   float64
 11  share_global_co2                      23103 non-null  float64
 12  cumulative_co2                        23578 non-null  float64
 13  share_global_cumulative_co2           23578 non-null  float64
 14  co2_per_gdp                           14918 non-null  float64
 15  consumption_co2_per_gdp               3088 non-null   float64
 16  co2_per_unit_energy                   6743 non-null   float64
 17  cement_co2                            12182 non-null  float64
 18  coal_co2                              16991 non-null  float64
 19  flaring_co2                           4302 non-null   float64
 20  gas_co2                               8693 non-null   float64
 21  oil_co2                               19711 non-null  float64
 22  other_industry_co2                    1563 non-null   float64
 23  cement_co2_per_capita                 12153 non-null  float64
 24  coal_co2_per_capita                   16471 non-null  float64
 25  flaring_co2_per_capita                4301 non-null   float64
 26  gas_co2_per_capita                    8665 non-null   float64
 27  oil_co2_per_capita                    19393 non-null  float64
 28  other_co2_per_capita                  1563 non-null   float64
 29  share_global_coal_co2                 16991 non-null  float64
 30  share_global_oil_co2                  19711 non-null  float64
 31  share_global_gas_co2                  8693 non-null   float64
 32  share_global_flaring_co2              4302 non-null   float64
 33  share_global_cement_co2               12182 non-null  float64
 34  cumulative_coal_co2                   18552 non-null  float64
 35  cumulative_oil_co2                    19963 non-null  float64
 36  cumulative_gas_co2                    9187 non-null   float64
 37  cumulative_flaring_co2                4933 non-null   float64
 38  cumulative_cement_co2                 12563 non-null  float64
 39  share_global_cumulative_coal_co2      18552 non-null  float64
 40  share_global_cumulative_oil_co2       19963 non-null  float64
 41  share_global_cumulative_gas_co2       9187 non-null   float64
 42  share_global_cumulative_flaring_co2   4933 non-null   float64
 43  share_global_cumulative_cement_co2    12563 non-null  float64
 44  total_ghg                             5208 non-null   float64
 45  ghg_per_capita                        5155 non-null   float64
 46  methane                               5211 non-null   float64
 47  methane_per_capita                    5157 non-null   float64
 48  nitrous_oxide                         5211 non-null   float64
 49  nitrous_oxide_per_capita              5157 non-null   float64
```

```
 50  primary_energy_consumption          6044 non-null   float64
 51  energy_per_capita                   6044 non-null   float64
 52  energy_per_gdp                      6044 non-null   float64
 53  population                          21071 non-null  float64
 54  gdp                                 13002 non-null  float64
dtypes: float64(52), int64(1), object(2)
memory usage: 9.9+ MB
```

In [56]:
```
CO2['country'].unique()
```

Out[56]:
```
array(['Afghanistan', 'Africa', 'Albania', 'Algeria', 'Andorra', 'Angol
a',
       'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Arub
a',
       'Asia', 'Asia (excl. China & India)', 'Australia', 'Austria',
       'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados',
       'Belarus', 'Belgium', 'Belize', 'Benin', 'Bermuda', 'Bhutan',
       'Bolivia', 'Bonaire Sint Eustatius and Saba',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei', 'Bulgaria', 'Burkina Faso',
       'Burundi', 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde',
       'Central African Republic', 'Chad', 'Chile', 'China',
       'Christmas Island', 'Colombia', 'Comoros', 'Congo', 'Cook Island
s',
       'Costa Rica', "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus',
       'Czechia', 'Democratic Republic of Congo', 'Denmark', 'Djibouti',
       'Dominica', 'Dominican Republic', 'EU-27', 'EU-28', 'Ecuador',
       'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Eswatini', 'Ethiopia', 'Europe', 'Europe (excl. EU-27)',
       'Europe (excl. EU-28)', 'Faeroe Islands', 'Fiji', 'Finland',
       'France', 'French Equatorial Africa', 'French Polynesia',
       'French West Africa', 'Gabon', 'Gambia', 'Georgia', 'Germany',
       'Ghana', 'Greece', 'Greenland', 'Grenada', 'Guatemala', 'Guinea',
       'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong',
       'Hungary', 'Iceland', 'India', 'Indonesia',
       'International transport', 'Iran', 'Iraq', 'Ireland', 'Israel',
       'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya',
       'Kiribati', 'Kosovo', 'Kuwait', 'Kuwaiti Oil Fires', 'Kyrgyzstan',
       'Laos', 'Latvia', 'Lebanon', 'Leeward Islands', 'Lesotho',
       'Liberia', 'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg',
       'Macao', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali',
       'Malta', 'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico',
       'Micronesia', 'Moldova', 'Mongolia', 'Montenegro', 'Montserrat',
       'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nauru', 'Nepal',
       'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua',
       'Niger', 'Nigeria', 'Niue', 'North America',
       'North America (excl. USA)', 'North Korea', 'North Macedonia',
       'Norway', 'Oceania', 'Oman', 'Pakistan', 'Palau', 'Palestine',
       'Panama', 'Panama Canal Zone', 'Papua New Guinea', 'Paraguay',
       'Peru', 'Philippines', 'Poland', 'Portugal', 'Puerto Rico',
       'Qatar', 'Romania', 'Russia', 'Rwanda', 'Ryukyu Islands',
       'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines',
       'Samoa', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
       'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore',
       'Sint Maarten (Dutch part)', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South America',
       'South Korea', 'South Sudan', 'Spain', 'Sri Lanka',
       'St. Kitts-Nevis-Anguilla', 'Sudan', 'Suriname', 'Sweden',
       'Switzerland', 'Syria', 'Taiwan', 'Tajikistan', 'Tanzania',
       'Thailand', 'Timor', 'Togo', 'Tonga', 'Trinidad and Tobago',
       'Tunisia', 'Turkey', 'Turkmenistan', 'Turks and Caicos Islands',
       'Tuvalu', 'Uganda', 'Ukraine', 'United Arab Emirates',
       'United Kingdom', 'United States', 'Uruguay', 'Uzbekistan',
```

```
       'Vanuatu', 'Venezuela', 'Vietnam', 'Wallis and Futuna Islands',
       'World', 'Yemen', 'Zambia', 'Zimbabwe'], dtype=object)
```

Let's drop the countries we don't need, for the sake of simplicity.

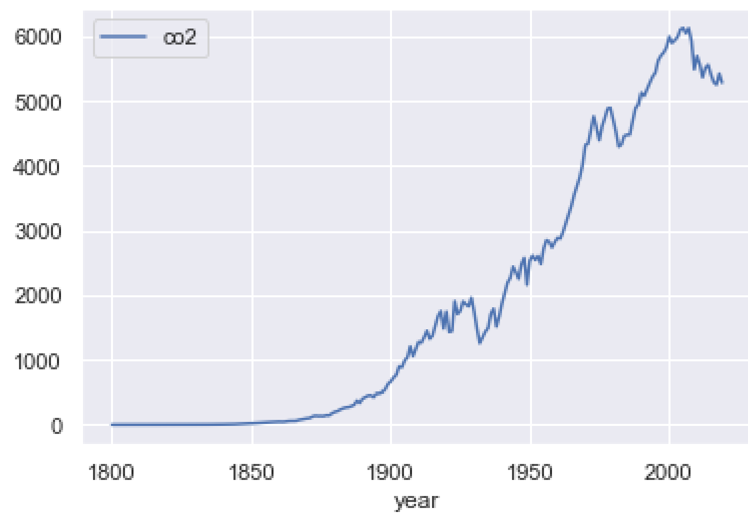
In [57]:
```
US_Energy_df = CO2[(CO2['country'] == 'United States')]

US_Energy_df.head()
```

Out[57]:

	iso_code	country	year	co2	co2_growth_prct	co2_growth_abs	consumption_co2	trade_c
22369	USA	United States	1800	0.253	NaN	NaN	NaN	N
22370	USA	United States	1801	0.267	5.797	0.015	NaN	N
22371	USA	United States	1802	0.289	8.219	0.022	NaN	N
22372	USA	United States	1803	0.297	2.532	0.007	NaN	N
22373	USA	United States	1804	0.333	12.346	0.037	NaN	N

5 rows × 55 columns

In [58]:
```
US_Energy_df.plot(x="year", y=["co2"])
plt.show()
```



Unsurprsingly, we can see a general uptrend in the carbon emissions for the united states, which supports our final recommendation of moving towards renewable energy sources.

In [59]:
```
Energy_production.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8424 entries, 0 to 8423
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   MSN           8424 non-null   object
 1   YYYYMM        8424 non-null   int64 
 2   Value         8424 non-null   object
 3   Column_Order  8424 non-null   int64 
 4   Description   8424 non-null   object
 5   Unit          8424 non-null   object
dtypes: int64(2), object(4)
memory usage: 395.0+ KB
```

In [60]:
```
CO2_dated = Energy_production[(Energy_production['YYYYMM'] > 199999)]
```

In [61]:
```
CO2_dated.head()
```

Out[61]:

	MSN	YYYYMM	Value	Column_Order	Description	Unit
375	CLPRBUS	200001	1.855655	1	Coal Production	Quadrillion Btu
376	CLPRBUS	200002	1.846984	1	Coal Production	Quadrillion Btu
377	CLPRBUS	200003	2.106475	1	Coal Production	Quadrillion Btu
378	CLPRBUS	200004	1.732611	1	Coal Production	Quadrillion Btu
379	CLPRBUS	200005	1.879543	1	Coal Production	Quadrillion Btu

In [62]:
```
del Energy_production['MSN']
```

In [63]:
```
del Energy_production['Column_Order']
```

In [64]:
```
Energy_production['Description'].unique()
```

Out[64]:
```
array(['Coal Production', 'Natural Gas (Dry) Production',
       'Crude Oil Production', 'Natural Gas Plant Liquids Production',
       'Total Fossil Fuels Production',
       'Nuclear Electric Power Production',
       'Hydroelectric Power Production', 'Geothermal Energy Production',
       'Solar Energy Production', 'Wind Energy Production',
       'Biomass Energy Production', 'Total Renewable Energy Production',
       'Total Primary Energy Production'], dtype=object)
```

In [65]:
```
Coal = Energy_production[(Energy_production['Description'] == 'Coal Product
L_Natural_Gas = Energy_production[(Energy_production['Description'] == 'Nat
D_Natural_Gas = Energy_production[(Energy_production['Description'] == 'Nat
Crude_Oil = Energy_production[(Energy_production['Description'] == 'Crude O
Nuclear = Energy_production[(Energy_production['Description'] == 'Nuclear E
Solar = Energy_production[(Energy_production['Description'] == 'Solar Energ
Hydroelectric = Energy_production[(Energy_production['Description'] == 'Hyd
Wind = Energy_production[(Energy_production['Description'] == 'Wind Energy
Geothermal = Energy_production[(Energy_production['Description'] == 'Geothe
```

In [66]:
```
Geothermal.tail()
Geothermal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 648 entries, 4536 to 5183
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   YYYYMM       648 non-null    int64
 1   Value        648 non-null    object
 2   Description  648 non-null    object
 3   Unit         648 non-null    object
dtypes: int64(1), object(3)
memory usage: 25.3+ KB
```

In [67]:
```
Coal = Coal[(Coal['YYYYMM'] > 199999)]
L_Natural_Gas = L_Natural_Gas[(L_Natural_Gas['YYYYMM'] > 199999)]
D_Natural_Gas = D_Natural_Gas[(D_Natural_Gas['YYYYMM'] > 199999)]
Crude_Oil = Crude_Oil[(Crude_Oil['YYYYMM'] > 199999)]
Nuclear = Nuclear[(Nuclear['YYYYMM'] > 199999)]
Solar = Solar[(Solar['YYYYMM'] > 199999)]
Hydroelectric = Hydroelectric[(Hydroelectric['YYYYMM'] > 199999)]
Wind = Wind[(Wind['YYYYMM'] > 199999)]
Geothermal = Geothermal[(Geothermal['YYYYMM'] > 199999)]
```

In [68]:
```
Coal = Coal.astype('str')
L_Natural_Gas = Coal.astype('str')
D_Natural_Gas = Coal.astype('str')
Crude_Oil = Crude_Oil.astype('str')
Nuclear = Nuclear.astype('str')
Solar = Solar.astype('str')
Hydroelectric = Hydroelectric.astype('str')
Wind = Wind.astype('str')
Geothermal = Geothermal.astype('str')
```

In [69]:
```
Coal = Coal[Coal['YYYYMM'].str.endswith('13')]
L_Natural_Gas = L_Natural_Gas[L_Natural_Gas['YYYYMM'].str.endswith('13')]
D_Natural_Gas = D_Natural_Gas[D_Natural_Gas['YYYYMM'].str.endswith('13')]
Crude_Oil = Crude_Oil[Crude_Oil['YYYYMM'].str.endswith('13')]
Nuclear = Nuclear[Nuclear['YYYYMM'].str.endswith('13')]
Solar = Solar[Solar['YYYYMM'].str.endswith('13')]
Hydroelectric = Hydroelectric[Hydroelectric['YYYYMM'].str.endswith('13')]
Wind = Wind[Wind['YYYYMM'].str.endswith('13')]
Geothermal = Geothermal[Geothermal['YYYYMM'].str.endswith('13')]
```

In [70]:
```
Coal.head(25)
```

Out[70]:

	YYYYMM	Value	Description	Unit
387	200013	22.735478	Coal Production	Quadrillion Btu
400	200113	23.54708	Coal Production	Quadrillion Btu
413	200213	22.732237	Coal Production	Quadrillion Btu
426	200313	22.093652	Coal Production	Quadrillion Btu
439	200413	22.852099	Coal Production	Quadrillion Btu
452	200513	23.185189	Coal Production	Quadrillion Btu
465	200613	23.78951	Coal Production	Quadrillion Btu
478	200713	23.492742	Coal Production	Quadrillion Btu
491	200813	23.851368	Coal Production	Quadrillion Btu
504	200913	21.623721	Coal Production	Quadrillion Btu
517	201013	22.038226	Coal Production	Quadrillion Btu
530	201113	22.221407	Coal Production	Quadrillion Btu
543	201213	20.676893	Coal Production	Quadrillion Btu
556	201313	20.001304	Coal Production	Quadrillion Btu
569	201413	20.285705	Coal Production	Quadrillion Btu
582	201513	17.946095	Coal Production	Quadrillion Btu
595	201613	14.667089	Coal Production	Quadrillion Btu
608	201713	15.625377	Coal Production	Quadrillion Btu
621	201813	15.363442	Coal Production	Quadrillion Btu
634	201913	14.255763	Coal Production	Quadrillion Btu
647	202013	10.802494	Coal Production	Quadrillion Btu

In [71]:
```
Coal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 387 to 647
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   YYYYMM       21 non-null     object
 1   Value        21 non-null     object
 2   Description  21 non-null     object
 3   Unit         21 non-null     object
dtypes: object(4)
memory usage: 840.0+ bytes
```

In [72]:
```
Coal.head(30)
```

Out[72]:

	YYYYMM	Value	Description	Unit
387	200013	22.735478	Coal Production	Quadrillion Btu
400	200113	23.54708	Coal Production	Quadrillion Btu
413	200213	22.732237	Coal Production	Quadrillion Btu
426	200313	22.093652	Coal Production	Quadrillion Btu
439	200413	22.852099	Coal Production	Quadrillion Btu
452	200513	23.185189	Coal Production	Quadrillion Btu
465	200613	23.78951	Coal Production	Quadrillion Btu
478	200713	23.492742	Coal Production	Quadrillion Btu
491	200813	23.851368	Coal Production	Quadrillion Btu
504	200913	21.623721	Coal Production	Quadrillion Btu
517	201013	22.038226	Coal Production	Quadrillion Btu
530	201113	22.221407	Coal Production	Quadrillion Btu
543	201213	20.676893	Coal Production	Quadrillion Btu
556	201313	20.001304	Coal Production	Quadrillion Btu
569	201413	20.285705	Coal Production	Quadrillion Btu
582	201513	17.946095	Coal Production	Quadrillion Btu
595	201613	14.667089	Coal Production	Quadrillion Btu
608	201713	15.625377	Coal Production	Quadrillion Btu
621	201813	15.363442	Coal Production	Quadrillion Btu
634	201913	14.255763	Coal Production	Quadrillion Btu
647	202013	10.802494	Coal Production	Quadrillion Btu

In [73]:
```
X_train = (Coal.merge(Solar, on="YYYYMM").merge(Wind, on='YYYYMM').merge(L_
```

In [74]:
```
X_train
```

Out[74]:

YMM	Value_x	Description_x	Unit_x	Value_y	Description_y	Unit_y	Value_x	Description_
)0013	22.735478	Coal Production	Quadrillion Btu	0.063469	Solar Energy Production	Quadrillion Btu	0.057057	Wind Energ Productic
)0113	23.54708	Coal Production	Quadrillion Btu	0.061674	Solar Energy Production	Quadrillion Btu	0.069617	Wind Energ Productic
)0213	22.732237	Coal Production	Quadrillion Btu	0.05998	Solar Energy Production	Quadrillion Btu	0.105334	Wind Energ Productic
)0313	22.093652	Coal Production	Quadrillion Btu	0.058432	Solar Energy Production	Quadrillion Btu	0.113273	Wind Energ Productic
)0413	22.852099	Coal Production	Quadrillion Btu	0.058376	Solar Energy Production	Quadrillion Btu	0.141664	Wind Energ Productic
)0513	23.185189	Coal Production	Quadrillion Btu	0.057893	Solar Energy Production	Quadrillion Btu	0.178088	Wind Energ Productic
)0613	23.78951	Coal Production	Quadrillion Btu	0.060755	Solar Energy Production	Quadrillion Btu	0.263738	Wind Energ Productic
)0713	23.492742	Coal Production	Quadrillion Btu	0.065621	Solar Energy Production	Quadrillion Btu	0.340503	Wind Energ Productic
)0813	23.851368	Coal Production	Quadrillion Btu	0.074207	Solar Energy Production	Quadrillion Btu	0.545548	Wind Energ Productic
)0913	21.623721	Coal Production	Quadrillion Btu	0.078178	Solar Energy Production	Quadrillion Btu	0.721129	Wind Energ Productic
)1013	22.038226	Coal Production	Quadrillion Btu	0.091282	Solar Energy Production	Quadrillion Btu	0.923427	Wind Energ Productic
)1113	22.221407	Coal Production	Quadrillion Btu	0.112429	Solar Energy Production	Quadrillion Btu	1.167636	Wind Energ Productic
)1213	20.676893	Coal Production	Quadrillion Btu	0.158961	Solar Energy Production	Quadrillion Btu	1.340059	Wind Energ Productic
)1313	20.001304	Coal Production	Quadrillion Btu	0.224524	Solar Energy Production	Quadrillion Btu	1.601359	Wind Energ Productic
)1413	20.285705	Coal Production	Quadrillion Btu	0.337421	Solar Energy Production	Quadrillion Btu	1.727542	Wind Energ Productic
)1513	17.946095	Coal Production	Quadrillion Btu	0.426734	Solar Energy Production	Quadrillion Btu	1.777306	Wind Energ Productic

YMM	Value_x	Description_x	Unit_x	Value_y	Description_y	Unit_y	Value_x	Description_
)1613	14.667089	Coal Production	Quadrillion Btu	0.570218	Solar Energy Production	Quadrillion Btu	2.095595	Wind Energ Productic
)1713	15.625377	Coal Production	Quadrillion Btu	0.776925	Solar Energy Production	Quadrillion Btu	2.342891	Wind Energ Productic
)1813	15.363442	Coal Production	Quadrillion Btu	0.915105	Solar Energy Production	Quadrillion Btu	2.482364	Wind Energ Productic
)1913	14.255763	Coal Production	Quadrillion Btu	1.017004	Solar Energy Production	Quadrillion Btu	2.634538	Wind Energ Productic
)2013	10.802494	Coal Production	Quadrillion Btu	1.24558	Solar Energy Production	Quadrillion Btu	3.005187	Wind Energ Productic

× 28 columns

In [87]:
```
Coal.rename(columns = {'Value':'Coal Value'}, inplace = True)
L_Natural_Gas.rename(columns = {'Value':'L_Gas Value'}, inplace = True)
D_Natural_Gas.rename(columns = {'Value':'D_Gas Value'}, inplace = True)
Crude_Oil.rename(columns = {'Value':'Oil Value'}, inplace = True)
Nuclear.rename(columns = {'Value':'Nuclear Value'}, inplace = True)
Hydroelectric.rename(columns = {'Value':'Hydro Value'}, inplace = True)
Geothermal.rename(columns = {'Value':'Geothermal Value'}, inplace = True)
Wind.rename(columns = {'Value':'Wind Value'}, inplace = True)
Solar.rename(columns = {'Value':'Solar Value'}, inplace = True)
```

In [88]:
```
US_Energy_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 220 entries, 22369 to 22588
Data columns (total 55 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   iso_code                             220 non-null    object 
 1   country                              220 non-null    object 
 2   year                                 220 non-null    int64  
 3   co2                                  220 non-null    float64
 4   co2_growth_prct                      219 non-null    float64
 5   co2_growth_abs                       219 non-null    float64
 6   consumption_co2                      29 non-null     float64
 7   trade_co2                            29 non-null     float64
 8   trade_co2_share                      29 non-null     float64
 9   co2_per_capita                       220 non-null    float64
 10  consumption_co2_per_capita           29 non-null     float64
 11  share_global_co2                     220 non-null    float64
 12  cumulative_co2                       220 non-null    float64
 13  share_global_cumulative_co2          220 non-null    float64
 14  co2_per_gdp                          197 non-null    float64
 15  consumption_co2_per_gdp              27 non-null     float64
 16  co2_per_unit_energy                  57 non-null     float64
 17  cement_co2                           120 non-null    float64
 18  coal_co2                             220 non-null    float64
 19  flaring_co2                          70 non-null     float64
 20  gas_co2                              138 non-null    float64
 21  oil_co2                              160 non-null    float64
 22  other_industry_co2                   30 non-null     float64
 23  cement_co2_per_capita                120 non-null    float64
 24  coal_co2_per_capita                  220 non-null    float64
 25  flaring_co2_per_capita               70 non-null     float64
 26  gas_co2_per_capita                   138 non-null    float64
 27  oil_co2_per_capita                   160 non-null    float64
 28  other_co2_per_capita                 30 non-null     float64
 29  share_global_coal_co2                220 non-null    float64
 30  share_global_oil_co2                 160 non-null    float64
 31  share_global_gas_co2                 138 non-null    float64
 32  share_global_flaring_co2             70 non-null     float64
 33  share_global_cement_co2              120 non-null    float64
 34  cumulative_coal_co2                  220 non-null    float64
 35  cumulative_oil_co2                   160 non-null    float64
 36  cumulative_gas_co2                   138 non-null    float64
 37  cumulative_flaring_co2               70 non-null     float64
 38  cumulative_cement_co2                120 non-null    float64
 39  share_global_cumulative_coal_co2     220 non-null    float64
 40  share_global_cumulative_oil_co2      160 non-null    float64
 41  share_global_cumulative_gas_co2      138 non-null    float64
 42  share_global_cumulative_flaring_co2  70 non-null     float64
 43  share_global_cumulative_cement_co2   120 non-null    float64
 44  total_ghg                            27 non-null     float64
 45  ghg_per_capita                       27 non-null     float64
 46  methane                              27 non-null     float64
 47  methane_per_capita                   27 non-null     float64
 48  nitrous_oxide                        27 non-null     float64
 49  nitrous_oxide_per_capita             27 non-null     float64
```

```
 50  primary_energy_consumption        57 non-null     float64
 51  energy_per_capita                 57 non-null     float64
 52  energy_per_gdp                    57 non-null     float64
 53  population                        220 non-null    float64
 54  gdp                               197 non-null    float64
dtypes: float64(52), int64(1), object(2)
memory usage: 96.2+ KB
```

In [89]:
```
CO2_dated = US_Energy_df[(US_Energy_df['year'] > 1999)]
```

In [90]:
```
CO2_dated = CO2_dated[['co2']]
```

In [91]:
```
CO2_dated.head(20)
```

Out[91]:

	co2
22569	5998.070
22570	5900.437
22571	5942.652
22572	5991.960
22573	6107.618
22574	6131.893
22575	6051.051
22576	6128.430
22577	5930.540
22578	5491.036
22579	5698.056
22580	5565.294
22581	5367.569
22582	5514.029
22583	5561.719
22584	5412.432
22585	5292.268
22586	5253.606
22587	5424.882
22588	5284.697

In [93]:
```
X_train = (Coal.merge(Solar, on="YYYYMM").merge(Wind, on='YYYYMM').merge(L_
```

In [94]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 0 to 20
Data columns (total 28 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   YYYYMM            21 non-null     object
 1   Coal Value        21 non-null     object
 2   Description_x     21 non-null     object
 3   Unit_x            21 non-null     object
 4   Solar Value       21 non-null     object
 5   Description_y     21 non-null     object
 6   Unit_y            21 non-null     object
 7   Wind Value        21 non-null     object
 8   Description_x     21 non-null     object
 9   Unit_x            21 non-null     object
 10  L_Gas Value       21 non-null     object
 11  Description_y     21 non-null     object
 12  Unit_y            21 non-null     object
 13  D_Gas Value       21 non-null     object
 14  Description_x     21 non-null     object
 15  Unit_x            21 non-null     object
 16  Oil Value         21 non-null     object
 17  Description_y     21 non-null     object
 18  Unit_y            21 non-null     object
 19  Nuclear Value     21 non-null     object
 20  Description_x     21 non-null     object
 21  Unit_x            21 non-null     object
 22  Hydro Value       21 non-null     object
 23  Description_y     21 non-null     object
 24  Unit_y            21 non-null     object
 25  Geothermal Value  21 non-null     object
 26  Description       21 non-null     object
 27  Unit              21 non-null     object
dtypes: object(28)
memory usage: 4.8+ KB
```

In [95]:
```
del X_train['Description_x']
```

In [96]:
```
del X_train['Unit_x']
```

In [97]:
```
del X_train['Description_y']
```

In [98]:
```
del X_train['Unit_y']
```

In [99]:
```
del X_train['Description']
```

In [100]:
```
del X_train['Unit']
```

In [101]:
```
CO2_dated
```

Out[101]:

	co2
22569	5998.070
22570	5900.437
22571	5942.652
22572	5991.960
22573	6107.618
22574	6131.893
22575	6051.051
22576	6128.430
22577	5930.540
22578	5491.036
22579	5698.056
22580	5565.294
22581	5367.569
22582	5514.029
22583	5561.719
22584	5412.432
22585	5292.268
22586	5253.606
22587	5424.882
22588	5284.697

In [102]:
```
y = CO2_dated['co2']
```

In [103]:
```
y
```

Out[103]:
```
22569    5998.070
22570    5900.437
22571    5942.652
22572    5991.960
22573    6107.618
22574    6131.893
22575    6051.051
22576    6128.430
22577    5930.540
22578    5491.036
22579    5698.056
22580    5565.294
22581    5367.569
22582    5514.029
22583    5561.719
22584    5412.432
22585    5292.268
22586    5253.606
22587    5424.882
22588    5284.697
Name: co2, dtype: float64
```

In [104]:
```
X_train = X_train.sort_values('YYYYMM')
```

In [105]:
```
X_train = X_train[:-1]
```

In [106]:
```
y
```

Out[106]:
```
22569    5998.070
22570    5900.437
22571    5942.652
22572    5991.960
22573    6107.618
22574    6131.893
22575    6051.051
22576    6128.430
22577    5930.540
22578    5491.036
22579    5698.056
22580    5565.294
22581    5367.569
22582    5514.029
22583    5561.719
22584    5412.432
22585    5292.268
22586    5253.606
22587    5424.882
22588    5284.697
Name: co2, dtype: float64
```

In [107]:
```
X_train
```

Out[107]:

	YYYYMM	Coal Value	Solar Value	Wind Value	L_Gas Value	D_Gas Value	Oil Value	Nuclear Value	Hydro Value
0	200013	22.735478	0.063469	0.057057	22.735478	22.735478	12.358101	7.862349	2.811116
1	200113	23.54708	0.061674	0.069617	23.54708	23.54708	12.281566	8.028853	2.241858
2	200213	22.732237	0.05998	0.105334	22.732237	22.732237	12.160213	8.145429	2.689017
3	200313	22.093652	0.058432	0.113273	22.093652	22.093652	11.959568	7.959622	2.792539
4	200413	22.852099	0.058376	0.141664	22.852099	22.852099	11.550086	8.222774	2.688468
5	200513	23.185189	0.057893	0.178088	23.185189	23.185189	10.974152	8.16081	2.702942
6	200613	23.78951	0.060755	0.263738	23.78951	23.78951	10.766775	8.214626	2.869035
7	200713	23.492742	0.065621	0.340503	23.492742	23.492742	10.741447	8.458589	2.446389
8	200813	23.851368	0.074207	0.545548	23.851368	23.851368	10.613302	8.426491	2.511108
9	200913	21.623721	0.078178	0.721129	21.623721	21.623721	11.340126	8.35522	2.668824
10	201013	22.038226	0.091282	0.923427	22.038226	22.038226	11.610472	8.434433	2.538541
11	201113	22.221407	0.112429	1.167636	22.221407	22.221407	11.99753	8.268698	3.102852
12	201213	20.676893	0.158961	1.340059	20.676893	20.676893	13.8419	8.061822	2.628702
13	201313	20.001304	0.224524	1.601359	20.001304	20.001304	15.865054	8.244433	2.562382
14	201413	20.285705	0.337421	1.727542	20.285705	20.285705	18.607136	8.337559	2.466577
15	201513	17.946095	0.426734	1.777306	17.946095	17.946095	19.712044	8.336886	2.321177
16	201613	14.667089	0.570218	2.095595	14.667089	14.667089	18.537316	8.426753	2.472442
17	201713	15.625377	0.776925	2.342891	15.625377	15.625377	19.575779	8.418968	2.766967
18	201813	15.363442	0.915105	2.482364	15.363442	15.363442	22.834796	8.438068	2.663138
19	201913	14.255763	1.017004	2.634538	14.255763	14.255763	25.473071	8.451852	2.563228

In [108]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_train, y, test_size=0
```

In [115]:
```
X_train.head(25)
```

Out[115]:

	YYYYMM	Coal Value	Solar Value	Wind Value	L_Gas Value	D_Gas Value	Oil Value	Nuclear Value	Hydro Value
5	200513	23.185189	0.057893	0.178088	23.185189	23.185189	10.974152	8.16081	2.702942
11	201113	22.221407	0.112429	1.167636	22.221407	22.221407	11.99753	8.268698	3.102852
3	200313	22.093652	0.058432	0.113273	22.093652	22.093652	11.959568	7.959622	2.792539
18	201813	15.363442	0.915105	2.482364	15.363442	15.363442	22.834796	8.438068	2.663138
16	201613	14.667089	0.570218	2.095595	14.667089	14.667089	18.537316	8.426753	2.472442
13	201313	20.001304	0.224524	1.601359	20.001304	20.001304	15.865054	8.244433	2.562382
2	200213	22.732237	0.05998	0.105334	22.732237	22.732237	12.160213	8.145429	2.689017
9	200913	21.623721	0.078178	0.721129	21.623721	21.623721	11.340126	8.35522	2.668824
19	201913	14.255763	1.017004	2.634538	14.255763	14.255763	25.473071	8.451852	2.563228
4	200413	22.852099	0.058376	0.141664	22.852099	22.852099	11.550086	8.222774	2.688468
12	201213	20.676893	0.158961	1.340059	20.676893	20.676893	13.8419	8.061822	2.628702
7	200713	23.492742	0.065621	0.340503	23.492742	23.492742	10.741447	8.458589	2.446389
10	201013	22.038226	0.091282	0.923427	22.038226	22.038226	11.610472	8.434433	2.538541
14	201413	20.285705	0.337421	1.727542	20.285705	20.285705	18.607136	8.337559	2.466577
6	200613	23.78951	0.060755	0.263738	23.78951	23.78951	10.766775	8.214626	2.869035

In [116]:
```
X_train = X_train.sort_values('YYYYMM')
```

In [117]:
```
import numpy as np
from sklearn                          import metrics, svm
from sklearn.linear_model             import LogisticRegression
from sklearn import preprocessing
from sklearn import utils
```

In [160]:
```
X_train
```

Out[160]:

	Coal Value	Solar Value	Wind Value	L_Gas Value	D_Gas Value	Oil Value	Nuclear Value	Hydro Value	Geotherma Valu
0	22.732237	0.059980	0.105334	22.732237	22.732237	12.160213	8.145429	2.689017	0.17116
1	22.093652	0.058432	0.113273	22.093652	22.093652	11.959568	7.959622	2.792539	0.17344
2	22.852099	0.058376	0.141664	22.852099	22.852099	11.550086	8.222774	2.688468	0.17814
3	23.185189	0.057893	0.178088	23.185189	23.185189	10.974152	8.160810	2.702942	0.18070
4	23.789510	0.060755	0.263738	23.789510	23.789510	10.766775	8.214626	2.869035	0.18120
5	23.492742	0.065621	0.340503	23.492742	23.492742	10.741447	8.458589	2.446389	0.18577
6	21.623721	0.078178	0.721129	21.623721	21.623721	11.340126	8.355220	2.668824	0.20018
7	22.038226	0.091282	0.923427	22.038226	22.038226	11.610472	8.434433	2.538541	0.20797
8	22.221407	0.112429	1.167636	22.221407	22.221407	11.997530	8.268698	3.102852	0.21231
9	20.676893	0.158961	1.340059	20.676893	20.676893	13.841900	8.061822	2.628702	0.21159
10	20.001304	0.224524	1.601359	20.001304	20.001304	15.865054	8.244433	2.562382	0.21400
11	20.285705	0.337421	1.727542	20.285705	20.285705	18.607136	8.337559	2.466577	0.21449
12	14.667089	0.570218	2.095595	14.667089	14.667089	18.537316	8.426753	2.472442	0.20960
13	15.363442	0.915105	2.482364	15.363442	15.363442	22.834796	8.438068	2.663138	0.20886
14	14.255763	1.017004	2.634538	14.255763	14.255763	25.473071	8.451852	2.563228	0.20126

All right looks like we have everything where it needs to go! let's fit our datasets so they can be measured usig random forest regressor, and then test for feature importance with XGBoost

In [122]:
```
y = y.drop(index=[22569,22570,22577,22584,22586])
```

In [123]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor ()
rfr.fit (X_train, y)
```

Out[123]: RandomForestRegressor()

In [131]:
```
from sklearn.metrics import accuracy_score
from scipy.cluster import hierarchy as hc
```

In [128]:
```
X_train = X_train.reset_index(drop=True)
```

In [129]:
```
y = y.reset_index(drop=True)
```

In [174]:
```
import sklearn.metrics as metrics
```

In [134]:
```
AS = RandomForestRegressor(n_estimators=250,min_samples_leaf=3 ,n_jobs=-1,m
%time AS.fit(X_train, y_train)
y_pred= AS.predict(X_test)
rfr.score(X_test, y_test)
```

```
CPU times: user 375 ms, sys: 70.1 ms, total: 445 ms
Wall time: 419 ms
```

Out[134]: 0.9061844189317136

In [144]:
```
X_train['YYYYMM'] = (X_train['YYYYMM'].astype(float))
X_train['Coal Value'] = (X_train['Coal Value'].astype(float))
X_train['Solar Value'] = (X_train['Solar Value'].astype(float))
X_train['Wind Value'] = (X_train['Wind Value'].astype(float))
X_train['L_Gas Value'] = (X_train['L_Gas Value'].astype(float))
X_train['D_Gas Value'] = (X_train['D_Gas Value'].astype(float))
X_train['Oil Value'] = (X_train['Oil Value'].astype(float))
X_train['Nuclear Value'] = (X_train['Nuclear Value'].astype(float))
X_train['Hydro Value'] = (X_train['Hydro Value'].astype(float))
X_train['Geothermal Value'] = (X_train['Geothermal Value'].astype(float))
```

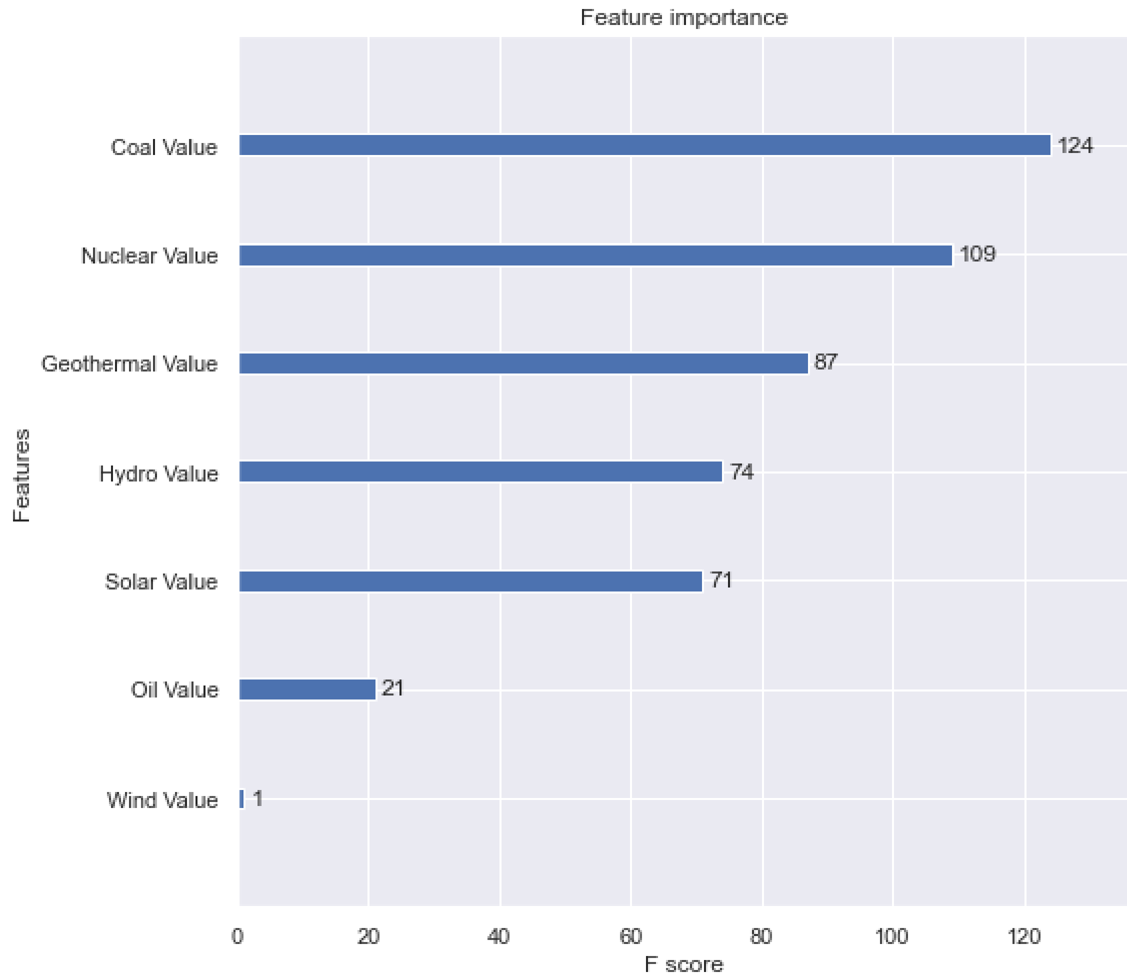
In [150]:
```
X_train.isnull()
```

Out[150]:

	YYYYMM	Coal Value	Solar Value	Wind Value	L_Gas Value	D_Gas Value	Oil Value	Nuclear Value	Hydro Value	Geothermal Value
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False	False

In [164]:
```
from xgboost import XGBRegressor
from xgboost import plot_importance

xgb = XGBRegressor()
xgb.fit(X_train, y)

im=pd.DataFrame({'importance':xgb.feature_importances_,'var':X_train.column
im=im.sort_values(by='importance',ascending=False)
```

In [165]:
```
fig,ax = plt.subplots(figsize=(8,8))
plot_importance(xgb,max_num_features=10,ax=ax)
plt.show()
```



In [171]:
```
from surprise.prediction_algorithms import SVD
from surprise import Reader, Dataset
reader = Reader()
from surprise.model_selection import cross_validate
```

In [172]:
```
data = Dataset.load_from_df(X_train[['Coal Value', 'Nuclear Value', 'Geothe
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=10, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 10 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Fold 6  Fold 7
Fold 8  Fold 9  Fold 10 Mean    Std
RMSE (testset)    0.8026  0.8095  0.7993  0.8066  0.8011  0.7884  0.8193
0.7855  0.7904  0.8288  0.8031  0.0130
MAE (testset)     0.8024  0.8093  0.7993  0.8065  0.8010  0.7884  0.8193
0.7855  0.7904  0.8288  0.8031  0.0130
Fit time          0.01    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00    0.00    0.00    0.00
```

Out[172]:
```
{'test_rmse': array([0.80256464, 0.80947218, 0.79927318, 0.80664024, 0.80
106739,
        0.788408  , 0.819297  , 0.78551   , 0.790396  , 0.828836  ]),
 'test_mae': array([0.802397 , 0.809288 , 0.799273 , 0.806494 , 0.800957
5, 0.788408 ,
        0.819297 , 0.78551  , 0.790396 , 0.828836 ]),
 'fit_time': (0.007638931274414062,
  0.0016126632690429688,
  0.0009551048278808594,
  0.0010819435119628906,
  0.0008447170257568359,
  0.0009279251098632812,
  0.0009770393371582031,
  0.0010519027709960938,
  0.0009191036224365234,
  0.0009129047393798828),
 'test_time': (9.608268737792969e-05,
  4.6253204345703125e-05,
  3.695487976074219e-05,
  3.1948089599609375e-05,
  2.4080276489257812e-05,
  1.7881393432617188e-05,
  1.6689300537109375e-05,
  2.4080276489257812e-05,
  2.47955322265625e-05,
  2.2172927856445312e-05)}
```

OSEMN-Interpret

Our accuracy scores are looking pretty great! All above 90%, we can see that shortwave radiation is a very good indicator for solar energy production.

Using radiation to be able to predict the amount of solar energy produced on a given day has fantastic real world applications, and not just financially! This can help plan around clean and useful energy for citizens, which is a rare win/win/win. Saving money, helping people, and protecting the environment are all possible with data like this, which can allow governments and energy companies to tailor their energy plants to create consistently profitable outputs of energy.

We can also see relative trends in CO_2 production, and the relationship that different energy sources have on carbon emissions. Looking at our models, we can see relatively low RMSE and MAE scores for our RandomForest and XGBoost, which raises our confidence in their accuracy.

Using this new information, power plants can accurately predict the output of renewable energy based on weather patterns, and carbon emissions of whichever energy they decide to utilize when constructing a powerplant. This should take some of the guesswork out of predicting the impacts of different energy sources.

In [ ]:

```

```