Report:

Data Science in Agriculture, Statistical Analysis and Geo Visualization

About

pre-preparing and making statistical analysis of Economic accounts for agriculture and to creating interactive maps showing the dynamics of the prices. All data is obtained from Eurostat Data Base (https://ec.europa.eu/eurostat/data/database).

Python Code:

```
pip install pycountry
conda install scikitlearn
import pandas as pd
import pycountry
import plotly.express as px

df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/data-science-in-agriculture-basic-statistical-analysis-and-geovisualisation/estat_aact_eaa01_defaultview_en.csv')
df

df.columns

col = df.columns[6:-1]
col

df = df[col]
df

df.info()

df.loc[:, 'geo'] = df['geo'].astype('category')
df.info()

df['geo'].unique()

df['geo'] = df['geo'].cat.add_categories(["GB", "GR"])

pd.options.mode.chained_assignment = None
mask = df['geo'] == 'UK'
df.loc[mask, 'geo'] = "GB"
df
```

```
mask = df['geo'] == 'EL'
df.loc[mask, 'geo'] = "GR"
df

list_alpha_2 = [i.alpha_2 for i in list(pycountry.countries)]
print("Country codes", list_alpha_2)

def country_flag(df):

if (df['geo'] in list_alpha_2):
return pycountry.countries.get(alpha_2=df['geo']).name
else:
print(df['geo'])
return 'Invalid Code'

df['country_name']=df.apply(country_flag, axis = 1)
df

mask = df['country_name'] != 'Invalid Code'
df = df[mask]
df
```

#Statistical analysis

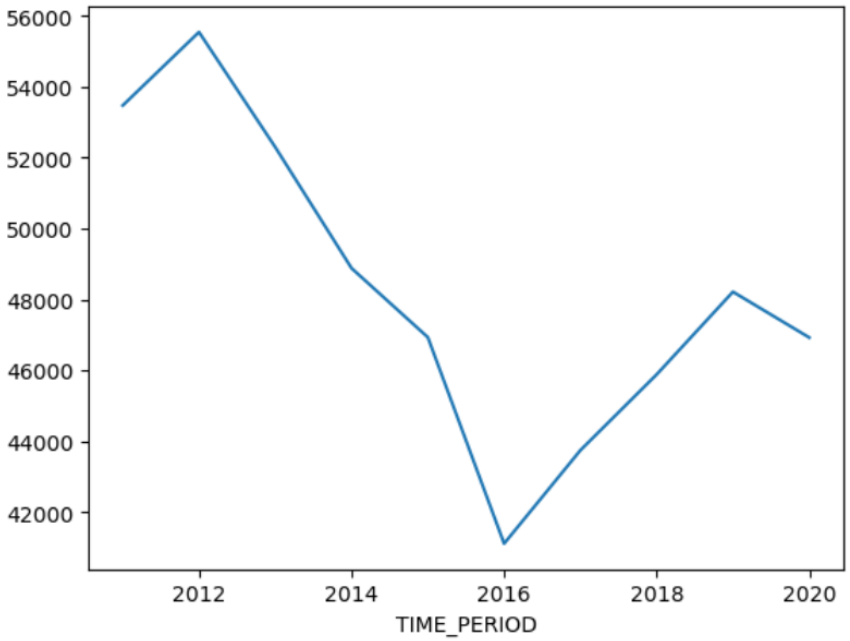
```
df.info()

df.describe()

df.describe(include=['category'])

df['country_name'].value_counts()

pt_country = pd.pivot_table(df, values= 'OBS_VALUE', index= ['TIME_PERIOD'],
columns=['country_name'], aggfunc='sum', margins=True)
pt_country

pt_country.describe()

pt = pd.pivot_table(df, values= 'OBS_VALUE', index= ['country_name'],
columns=['TIME_PERIOD'], aggfunc='sum', margins=True)
pt

pt.describe()
```

#Data visualization

#useing Pandas and Matplotlib SeaBorn libraries. Let's build a plot for the last row ('All') except the last values for column ('All'). Pandas inherits Matplotlib function for plotting.

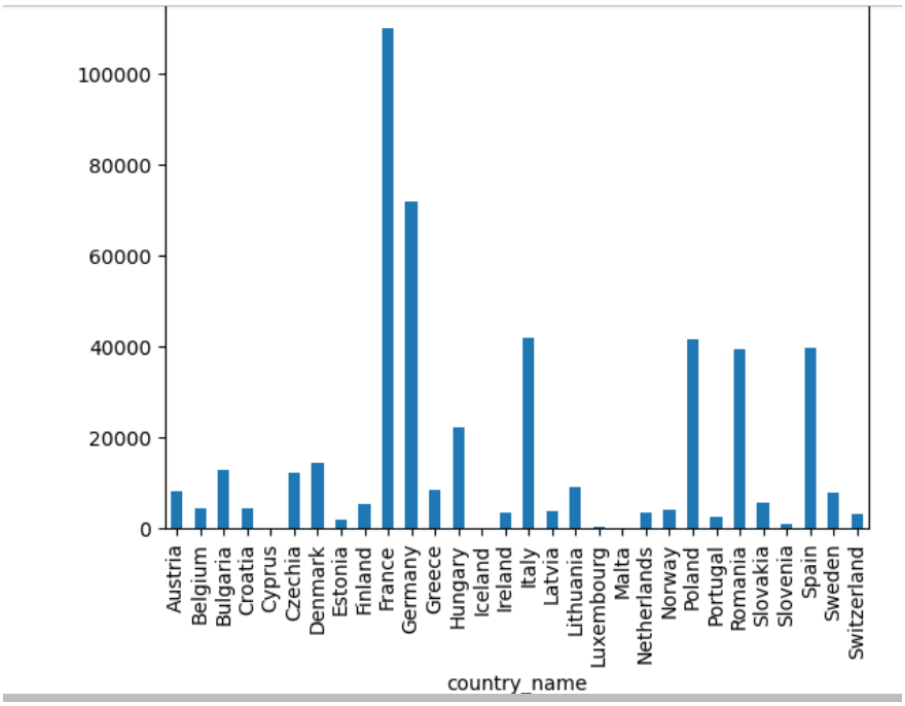
pt.iloc[-1][:-1].plot()

```
[29]: <AxesSubplot:xlabel='TIME_PERIOD'>
```



#Let's build a bar plot for summary values for each country.

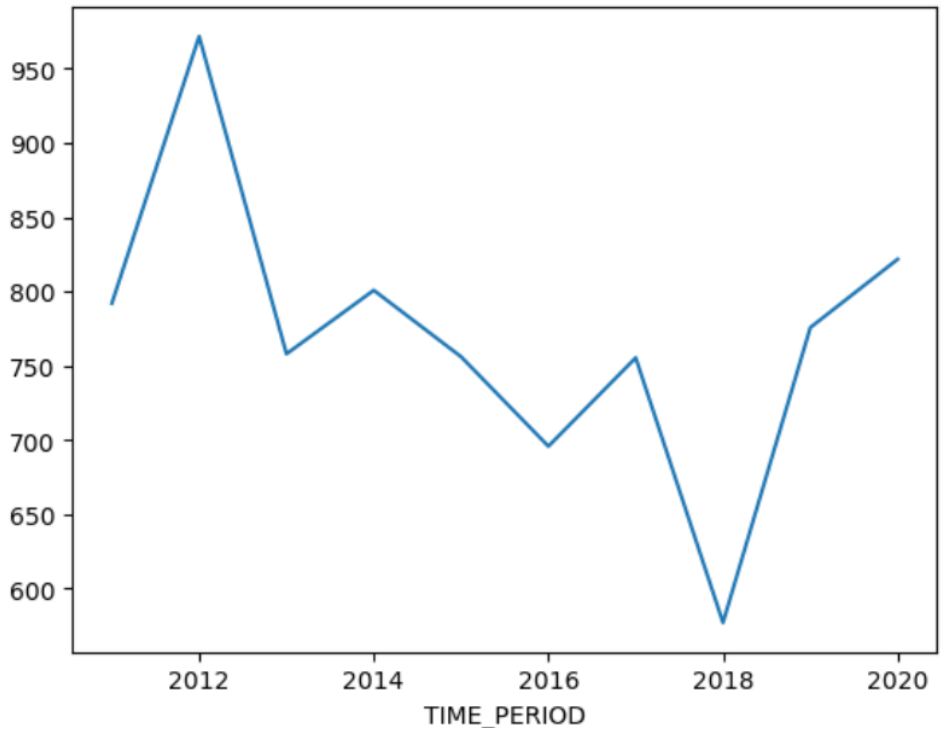
pt['All'][:-1].plot.bar(x='country_name', y='val', rot=90)



#Let's build a plot for economic accounts dynamics for Sweden.

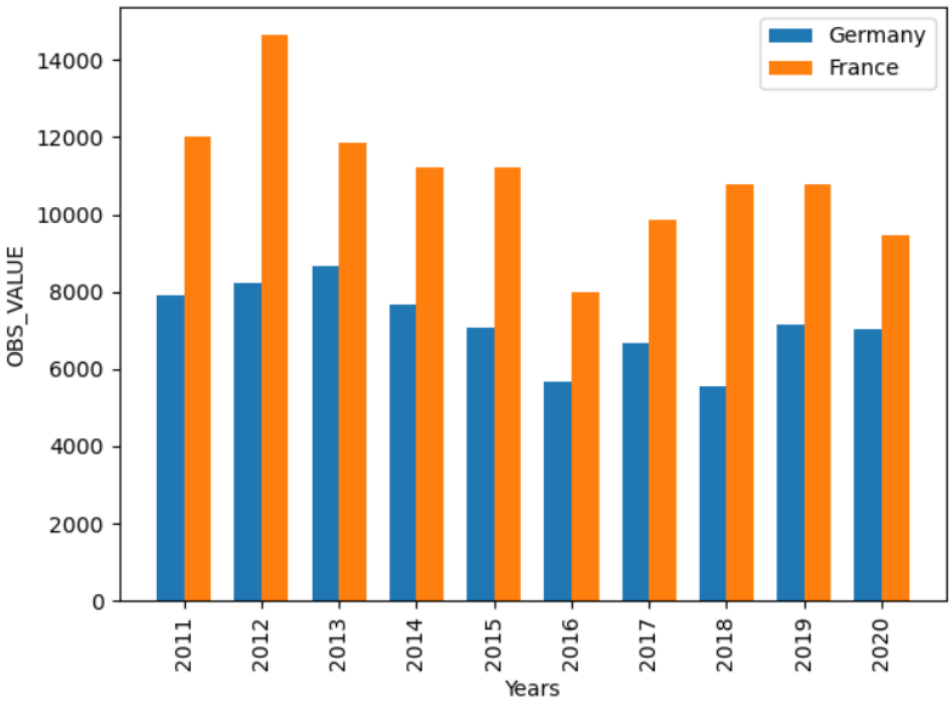
pt.loc['Sweden'][:-1].plot()

```
[31]: <AxesSubplot:xlabel='TIME_PERIOD'>
```



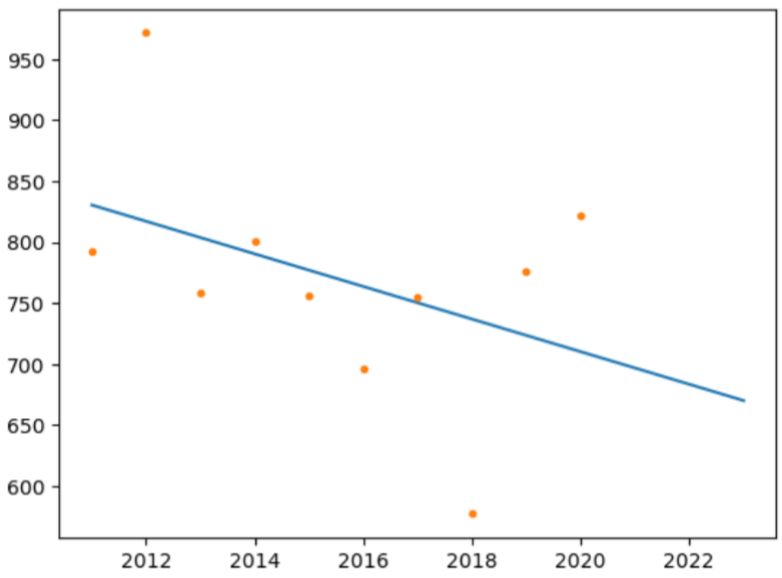
#Let's compare economic accounts for Germany and France on a bar plot. To do this we should make a lot of preparation:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(len(pt.columns)-1) # the label locations
width = 0.35 # the width of the bars

fig, ax = plt.subplots() # Create subplots
rects1 = ax.bar(x - width/2, pt.loc['Germany'][:-1], width, label='Germany') #
parameters of bars
rects2 = ax.bar(x + width/2, pt.loc['France'][:-1], width, label='France')
```




```
# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('OBS_VALUE')
ax.set_xlabel('Years')
ax.set_xticks(x)
plt.xticks(rotation = 90)
ax.set_xticklabels(pt.columns[:-1])
ax.legend()

fig.tight_layout()

plt.show()


#Also we can build some specific plots using SeaBorn library.

import seaborn as sns
d = pd.DataFrame(pt.loc['Sweden'][:-1])
print(d)
sns.regplot(x=d.index.astype(int), y="Sweden", data=d,)


X_pred= np.append(X, [2021, 2022, 2023])
X_pred = np.reshape(X_pred, (-1, 1))
# calculate trend
trend = model.predict(X_pred)

plt.plot(X_pred, trend, "-", X, y, ".")
```

```
[35]: [<matplotlib.lines.Line2D at 0x7f6d217a2450>,
       <matplotlib.lines.Line2D at 0x7f6d217a2250>]
```



#Interactive maps

```
import json
!wget european-union-countries.geojson "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/data-science-in-agriculture-basic-statistical-analysis-and-geo-visualisation/european-union-countries.geojson"
with open("european-union-countries.geojson", encoding="utf8") as json_file:
EU_map = json.load(json_file)

fig = px.choropleth(
df,
geojson=EU_map,
locations='country_name',
featureidkey='properties.name',
color= 'OBS_VALUE',
scope='europe',
hover_name= 'country_name',
hover_data= ['country_name', 'OBS_VALUE'],
animation_frame= 'TIME_PERIOD',
color_continuous_scale=px.colors.diverging.RdYlGn[::-1]
)

fig.update_geos(showcountries=False, showcoastlines=False, showland=True, fitbounds=False)

fig.update_layout(
title_text ="Agriculture Economic accounts",
title_x = 0.5,
geo= dict(
showframe= False,
showcoastlines= False,
```

```
projection_type = 'equirectangular'
),
margin={"r":0,"t":0,"l":0,"b":0}
)

from IPython.display import HTML
HTML(fig.to_html())
```