Google Colaboratory

El notebook se abrió con la configuración de resultados privados; estos no se guardarán. Puedes inhabilitar esta opción en la configuración del notebook

Abrir la configuración del notebook

.

Tobacco Consumption.ipynb_ Cambiar el nombre del bloc de notas

Cambiar el nombre del bloc de notas

Destacar

Destacar/dejar de destacar el bloc de notas en Google Drive

Se guardaron todos los cambios
Comentar

Abrir panel de comentarios

Abrir configuración

Código

Insertar celda de código abajo

Ctrl+M B

Texto

Agregar celda de texto

Activar o desactivar la visibilidad del encabezado

Final Project: Tobacco Consumption

Wave 2 - Group 4 - Team 4
Team Members:

- Saúl Yael Puente Ruiz
- Ana Mónica Lizette Turcios Esquivel

Import Python Modules

Enter to Rename, Shift+Enter to Preview

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. U
  import pandas.util.testing as tm
```

Data Preparation

As a first step we import the dataset and take a look at the data.

```
TobaccoConsumptionDF = pd.read_csv("https://raw.githubusercontent.com/SaulPuente/Tobacco-Consumption/main/Tobac
TobaccoConsumptionDF.head()
```

This dataset contains information about the tobaccoo consumption in the US along 20 years.
We know this about the variables in the dataset:

- Year: Year in which the information was obtained.
- LocationAbbrev: Place in which the information was obtained.
- LocationDesc: Brief description about the location.
- Population: National population at that year.
- Topic: Type of tobacco (Combustible / Noncombustible)
- Measure: Kind of tobacco.
- Submeasure: More specific kind of tobacco.
- Data value unit: The data value unit in which the consumption was measured.
- Domestic: Amount of tobacco produced in the US.
- Imports: Amount of imported tobacco.
- Total: Total tobacco consumption (Domestic + Imports).
- Domestic Per Capita: Amount of tobacco produced in the US per person (Domestic / Population).
- Imports Per Capita: Amount of imported tobacco per person (Imports / Population).
- Total Per Capita: Total tobacco consumption per person (Total / Population).

```
len(pd.unique(TobaccoConsumptionDF['LocationAbbrev'])), len(pd.unique(TobaccoConsumptionDF['LocationDesc']))
```

As we can see, the values of 'LocationAbbrev' and 'LocationDesc' do not change, so we can ignore both columns.

```
TobaccoConsumptionDF = TobaccoConsumptionDF.drop(['LocationAbbrev', 'LocationDesc'], axis=1)
TobaccoConsumptionDF.head()
```

Now we get some information from the data

```
TobaccoConsumptionDF.describe().transpose()
```

and verify if there are missing values.

```
TobaccoConsumptionDF.isnull().sum()
```

```
Year                0
Population          0
Topic               0
Measure             0
Submeasure          0
Data Value Unit     0
Domestic            0
Imports             0
Total               0
Domestic Per Capita 0
Imports Per Capita  0
Total Per Capita    0
dtype: int64
```

Data Analysis and Visualization

% Per Capita by type of contribution (domestic or imports)

```
TobaccoConsumptionByMeasureDF = TobaccoConsumptionDF[['Measure', 'Domestic Per Capita',
                                    'Imports Per Capita', 'Total Per Capita']].groupby(['Measure'])

aux1 = TobaccoConsumptionDF[['Measure', 'Domestic Per Capita',
        'Imports Per Capita', 'Total Per Capita']].set_index('Measure').sort_index().sum(level='Measure')
aux2 = TobaccoConsumptionDF[['Measure', 'Total Per Capita']].set_index('Measure').sort_index().sum(level='Measure').value
```

```
aux2 = [i[0] for i in aux2]
aux3 = [sum(aux2) for i in range(5)]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Using the level keyword in DataFrame and S

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: Using the level keyword in DataFrame and S
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
auxDF = pd.DataFrame({"Measure": aux1.index,
                      "Total Per Capita": aux3,
                     "Domestic Per Capita": aux2,
                     "Imports Per Capita": aux2}).set_index('Measure')
auxDF.head()
```

```
TobaccoConsumptionByMeasureDF = np.round(aux1/auxDF*100,2)
TobaccoConsumptionByMeasureDF
```

```
TobaccoConsumptionByMeasureDF.plot.pie(y="Domestic Per Capita", autopct='%1.2f%%', figsize=(8, 8), startangle=25)
plt.legend(loc="lower right", title = "Measures:")
plt.title("Domestic Per Capita by type of Contribution", fontsize=20)
plt.ylabel("Domestic Per Capita", fontsize=16)
```

```
TobaccoConsumptionByMeasureDF.plot.pie(y="Imports Per Capita", autopct='%1.2f%%', figsize=(8, 8), startangle=25)
plt.legend(loc="lower right", title = "Measures:")
plt.title("Imports Per Capita by type of Contribution", fontsize=20)
plt.ylabel("Imports Per Capita", fontsize=16)
```

```
TobaccoConsumptionByMeasureDF.plot.pie(y="Total Per Capita", autopct='%1.2f%%', figsize=(8, 8))
plt.legend(loc="lower right", title = "Measures:")
plt.title("Total Per Capita by type of Contribution", fontsize=20)
plt.ylabel("Total Per Capita", fontsize=16)
```

```
TobaccoConsumptionByMeasureDF.loc['Total General']= TobaccoConsumptionDF[['Domestic Per Capita', 'Imports Per Capita
TobaccoConsumptionByMeasureDF
```

DISTRIBUTION OF TOTAL AVERAGE PerCapita BY SUBMEASURES,YEARS 2000-2020

```
pd.set_option("display.max_rows", None, "display.max_columns", None)
TobaccoConsumptionDFBySubmeasure = TobaccoConsumptionDF[['Year','Topic','Measure','Submeasure','Total Per Capita']].s
TobaccoConsumptionDFBySubmeasure
```

```
TobaccoConsumptionDF[['Year','Measure','Total Per Capita']].pivot_table(index='Year',columns='Measure', values='Total Per
```

```
TobaccoConsumptionDF[['Year','Submeasure','Total Per Capita']].pivot_table(index='Year',columns='Submeasure', values='To
```

```
TobaccoConsumptionDF[['Year','Measure','Total Per Capita']].loc[TobaccoConsumptionDF.loc[:,'Measure'] != 'All Combustible
```

Distribution PerCapita (domestic and imports) by Measure

```
TobaccoConsumptionDF[['Year','Measure','Domestic Per Capita','Imports Per Capita','Total Per Capita']].pivot_table(index='Ye
```

```
TobaccoConsumptionDF[['Year','Measure','Domestic Per Capita']].pivot_table(index='Year',columns='Measure', values='Dome
```

```
TobaccoConsumptionDF[['Measure','Domestic Per Capita']].loc[TobaccoConsumptionDF.loc[:,'Measure'] != 'All Combustibles'
plt.ylabel("Measure")
```

```
TobaccoConsumptionDF.pivot_table(index='Year', columns='Submeasure', values='Total Per Capita')
```

Tobacco Consumption by Year

```
TotalPerCapitaTS = TobaccoConsumptionDF[['Year','Total Per Capita']].groupby('Year').mean()
TotalPerCapitaTS
```

```
fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
```

```
TotalPerCapitaTS['Total Per Capita'].plot(figsize=(11, 6))
plt.xlabel("Year")
plt.ylabel("Consumption Per Capita")
plt.title("Consumption Average Per Capita by Years, 2000 - 2020")
```

As can be seen, the average total tobacco consumption per capita shows a downward trend.

```
plt.figure(figsize=(16,8))
CombustibleTobaccoConsumptionbySubmeasureDF = TobaccoConsumptionDF[['Year','Topic','Submeasure','Total Per Capita']
CombustibleTobaccoConsumptionbySubmeasureDF.drop('Topic', axis='columns', inplace=True)
CombustibleTobaccoConsumptionbySubmeasureDF.groupby('Submeasure')["Total Per Capita"].plot(legend=True)
```

How can we analyze the behavior of the different sub-measures of the type of tobacco: combustible tobacco shows the same downward trend, with the Cigarette Removals type being the one with the greatest contribution.

```
plt.figure(figsize=(16,8))
CombustibleTobaccoConsumptionbySubmeasureDF = TobaccoConsumptionDF[['Year','Topic','Submeasure','Total Per Capita']
CombustibleTobaccoConsumptionbySubmeasureDF = CombustibleTobaccoConsumptionbySubmeasureDF.loc[CombustibleTo
CombustibleTobaccoConsumptionbySubmeasureDF = CombustibleTobaccoConsumptionbySubmeasureDF.loc[CombustibleTo
CombustibleTobaccoConsumptionbySubmeasureDF.drop('Topic', axis='columns', inplace=True)
CombustibleTobaccoConsumptionbySubmeasureDF.groupby('Submeasure')["Total Per Capita"].plot(legend=True)
```

Machine Learning Model

To predict the cigarettes consumption for 2021 we are using the SARIMAX model as we saw in class. First we need to prepare the time series. From the original dataset we are selecting the 'Year' and 'Total' columns, taking only the rows where 'Measure' is 'Cigarettes.

```
CigarettesConsumptionDF = TobaccoConsumptionDF[["Year", "Measure", "Total"]].loc[TobaccoConsumptionDF["Measure"] =
CigarettesConsumptionDF.drop('Measure', axis='columns', inplace=True)
CigarettesConsumptionDF
```

The next step is to select 'Year' as the index of our time series, and convert its values to datetime.

```
from datetime import datetime
CigarettesConsumptionDF['Year']=pd.to_datetime(CigarettesConsumptionDF['Year'], format='%Y')
CigarettesConsumptionDF.set_index('Year', inplace=True)

#check datatype of index
CigarettesConsumptionDF.index
```

```
DatetimeIndex(['2000-01-01', '2001-01-01', '2002-01-01', '2003-01-01',
               '2004-01-01', '2005-01-01', '2006-01-01', '2007-01-01',
               '2008-01-01', '2009-01-01', '2010-01-01', '2011-01-01',
               '2012-01-01', '2013-01-01', '2014-01-01', '2015-01-01',
               '2016-01-01', '2017-01-01', '2018-01-01', '2019-01-01',
               '2020-01-01'],
              dtype='datetime64[ns]', name='Year', freq=None)
```

Now we can train our model with the data from the time series.

```
# fit model
model = SARIMAX(CigarettesConsumptionDF['Total'], order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
model_fit = model.fit(disp=False)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information wa
  % freq, ValueWarning)
```

With the model ready we can predict the cigarettes consumption for 2021.

```
# make prediction
yhat = model_fit.predict(len(CigarettesConsumptionDF), len(CigarettesConsumptionDF))
print("Prediction of cigarette consumption for 2021: " + str(round(yhat.values[0])) + " Cigarettes")
```

```
Prediction of cigarette consumption for 2021: 226674854356 Cigarettes
```

We can also compare the predictions with real values of the time series by plotting them.

```
predictions = []
for i in range(len(CigarettesConsumptionDF)+1):
  predictions.append(model_fit.predict(i,i))
predictions = np.array(predictions)

years, cigarettes = np.array([i for i in range(2000,2021)]), CigarettesConsumptionDF['Total'].to_numpy()

plt.figure(figsize=(12,8))
plt.plot(years, cigarettes, label="Real Values")
plt.plot(np.append(years,2021)[1:],predictions[1:],'or', label="Predicted Values")
plt.xlabel('Years', fontsize=14)
plt.ylabel('Cigarretes Consumption', fontsize=14)
plt.title('Cigarette Consumption Per Year', fontsize=16)
plt.legend()
plt.show()
```

```
#convert to time series:
ts = CigarettesConsumptionDF['Total']
ts.head(10)
```

```
Year
2000-01-01    4.355700e+11
2001-01-01    4.267200e+11
2002-01-01    4.157240e+11
2003-01-01    4.003270e+11
2004-01-01    3.976550e+11
2005-01-01    3.810980e+11
2006-01-01    3.805940e+11
```

```
2007-01-01    3.615900e+11
2008-01-01    3.464200e+11
2009-01-01    3.177360e+11
Name: Total, dtype: float64

def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = timeseries.rolling(4).mean()
    rolstd = timeseries.rolling(4).std()

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show()


    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
       dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)

plt.figure(figsize=(10,6))
test_stationarity(ts)
```