#!/usr/bin/env python

# coding: utf-8

# # \*\*Lab 2.3: Visualization\*\*

#

# <hr>

#

# ## \*\*1. Objective\*\*

# In this practice, we will use the dataset generated in the previous practice and visualize various data in order to extract knowledge.

# To do this, we will need to install new libraries in our conda environment.

#

# ##### \*\*Matplotlib\*\*

#

# A library that allows the creation of a wide variety of plots, including:

#

# \* Histograms.

# \* Scatter plots.

# \* Bar charts.

# \* Heatmaps.

# \* Etc.

#

# It offers multiple customization options.

# Once created, the plots can be exported in many formats (jpg, png, pdf...)

#

# Example: https://matplotlib.org/gallery/lines\_bars\_and\_markers/scatter\_demo2.html#sphx-glr-gallery-lines-bars-and-markers-scatter-demo2-py

#

# \*To install it:\*

# In[1]:

get\_ipython().system(' pip install matplotlib')

# ##### \*\*Seaborn\*\*

#

# Based on Matplotlib, this library acts as a high-level interface over the former.

# This greatly simplifies the most common operations.

# It also adds new types of plots.

#

# Example: https://seaborn.pydata.org/examples/scatter\_bubbles.html

#

# \*To install it:\*

# In[2]:

get\_ipython().system(' pip install seaborn')

# ##### \*\*Bokeh\*\*

#

# Unlike the previous ones, Bokeh allows the creation of interactive plots.

# Thanks to this, we can modify the plot once it's created.

#

# Example: https://demo.bokeh.org/crossfilter

#

# \*To install it:\*

# In[3]:

get\_ipython().system(' pip install bokeh')

# <hr>

#

# ## \*\*2. Data Visualization\*\*

#

# The first thing we need to do is load the data from the previous practice.

# In[4]:

import pandas as pd

data = pd.read\_pickle('f1\_23\_monaco.pkl')

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Verify that the data types are correct.

# </div>

# In[5]:

# Your code here

print(data.dtypes)

# ### \*\*2.1. Static Visualization\*\*

# We will begin with static visualization, which does not allow manipulating the graph once it is generated.

#

# First, we will load the libraries and set some variables that will be used throughout the practice.

# In[6]:

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style('whitegrid') # Sets the style of the plots

import bokeh as bk

# Graph size

fig\_size = (12,6)

# Dictionary of colors for each team, driver, and tire

team\_colors = {'Red Bull Racing': '#1E41FF', 'Alpine': '#2293D1', 'Aston Martin': '#006F62', 'Ferrari': '#DC0000', 'Williams': '#005AFF', 'Haas F1 Team': '#787878', 'AlphaTauri': '#2B4562', 'Alfa Romeo': '#900000', 'McLaren': '#FF8700', 'Mercedes': '#6CD3BF'}

driver\_colors = {'VER': '#1E41FF', 'GAS': '#2293D1', 'PER': '#1E41FF', 'ALO': '#006F62', 'LEC': '#DC0000', 'STR': '#006F62', 'SAR': '#005AFF', 'MAG': '#787878', 'DEV': '#2B4562', 'TSU': '#2B4562', 'ALB': '#005AFF', 'ZHO': '#900000', 'HUL': '#787878', 'OCO': '#2293D1', 'NOR': '#FF8700', 'HAM': '#6CD3BF', 'SAI': '#DC0000', 'RUS': '#6CD3BF', 'BOT': '#900000', 'PIA': '#FF8700'}

tire\_colors = { 'SOFT': '#FF0000', 'HARD': '#cccccc', 'MEDIUM': '#FFD700', 'INTERMEDIATE': '#00A550', 'WET': '#005AFF' }

# #### \*\*Distributions\*\*

#

# Distributions are obtained with the aim of showing the possible values that a specific variable can take. Additionally, we can observe the frequency of those values.

#

# \_For a single variable:\_

# \* Histogram: `sns.histplot` or `sns.countplot`

# \* KDE (Kernel Density Estimate): `sns.kdeplot`

#

# The KDE is a smoothed version of a histogram where the y-axis shows the likelihood of finding a value at each point, instead of just counting how many data points exist in each interval.

#

# \_For multiple variables:\_

# \* KDE2D: `sns.jointplot`

# \*\*Histograma de la columna "SpeedI1".\*\*

# In[7]:

plt.figure(figsize = fig\_size); # Size of the plot

hist = sns.histplot(data = data['SpeedI1'], kde = False, bins = range(90, 220, 5));

hist.set\_xticks(range(90, 220, 5)); # X-axis labels

hist.set\_xlabel('SpeedI1')

hist.set\_ylabel('Number of laps')

hist.set\_title('Histogram of the ''SpeedI1'' variable')

# \*\*Otra notación para lo anterior\*\*

# In[8]:

# Another notation

plt.figure(figsize = fig\_size);

hist = sns.histplot(data = data, x = 'SpeedI1', kde = False, bins = range(90, 220, 5));

hist.set\_xticks(range(90, 220, 5));

hist.set\_xlabel('SpeedI1')

hist.set\_ylabel('Number of laps')

hist.set\_title('Histogram of the ''SpeedI1'' variable')

# \*\*Histogram of the number of laps by team\*\*

#

# Team, as you may know, is a categorical variable, which is why we use countplot.

# In[9]:

plt.figure(figsize = fig\_size); # Size of the plot

hist = sns.countplot(x = data['Team'], hue = data["Team"], palette = team\_colors);

hist.set\_xlabel('Team')

hist.set\_ylabel('Number of laps')

hist.set\_title('Histogram of the ''Team'' variable')

# \*\*KDE of the speed at the finish line for the drivers: ALO, VER, LEC, and NOR\*\*

# In[10]:

filtered\_drivers = data.loc[data['Driver'].isin(['ALO', 'VER', 'LEC', 'NOR'])]

plt.figure(figsize = fig\_size)

hist = sns.kdeplot(data = filtered\_drivers, x = filtered\_drivers['SpeedFL'])

hist.set\_xlabel('Lap time (seconds)')

hist.set\_ylabel('Density')

hist.set\_title('KDE of the ''LapTime'' variable')

plt.show()

# \*\*KDE (separated by driver) of the speed at the finish line for the drivers: ALO, VER, LEC, and NOR\*\*

# In[11]:

plt.figure(figsize = fig\_size)

hist = sns.kdeplot(data = filtered\_drivers, x = filtered\_drivers['SpeedFL'], hue = 'Driver', palette = driver\_colors, fill = True)

hist.set\_xlabel('Lap time (seconds)')

hist.set\_ylabel('Density')

hist.set\_title('KDE of the ''LapTime'' variable')

plt.show()

# \*\*2D KDE (separated by driver) of the time in the first sector versus the third sector for ALO, VER, LEC, and NOR for the first 30 laps\*\*

# In[12]:

filtered\_drivers = data.loc[(data['LapNumber'] <= 30) & (data['Driver'].isin(['VER', 'ALO', 'NOR']))]

hist = sns.jointplot(data = filtered\_drivers, x = filtered\_drivers['Sector3Time'].dt.total\_seconds(), y = filtered\_drivers['Sector1Time'].dt.total\_seconds(), hue = 'Driver', palette = driver\_colors)

plt.show()

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Create a chart with the number of laps completed with each type of tire. Use the already created variable 'tire\_colors'.

# </div>

# In[13]:

# Your code here

plt.figure(figsize = fig\_size);

chart = sns.countplot(x = data['Compound'], hue = data["Compound"], palette = tire\_colors);

chart.set\_xlabel('Type of Tyre')

chart.set\_ylabel('Number of laps')

chart.set\_title('Histogram of the Type of Tyre variable')

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Create a chart with the positions Kevin Magnussen (MAG) has passed through. It should reflect how many laps he has been in each of the 20 positions. All possible positions should appear on the x-axis.

# </div>

# In[14]:

# Your code here

# Another notatio

plt.figure(figsize = fig\_size)

mag=data[data['Driver']== 'MAG']

hist = sns.histplot(data = mag, x = 'Position', kde = False, bins=range(1, 21, 1))

hist.set\_xticks(range(1, 21, 1))

hist.set\_xlabel('Position')

hist.set\_ylabel('Laps')

hist.set\_title('Histogram of the Magnussen position')

# #### \*\*Relationships\*\*

#

# There are multiple types of charts to represent the relationship between variables (2 typically).

#

# \_For 2 variables:\_

# \* Bars: `sns.barplot`

# \* Line: `sns.lineplot`

# \* Scatter: `sns.scatterplot`

# \* Regression line: `sns.regplot`

#

# \_For more variables:\_

# \* Color and size to represent more information.

# \* Pairwise: `sns.lineplot`

# \* 3D visualization: `mpl\_toolkits.mplot3d`

# \* Reduce to 2 variables:

# \* PCA (Principal Component Analysis): `sklearn.decomposition.PCA`

# \* TSNE (T-distributed Stochastic Neighbor Embedding): `sklearn.manifold.TSNE`

# \*\*Bar chart: Relationship between lap time and position.\*\*

# In[15]:

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.barplot(data = data, x = 'Position', y = data['LapTime'].dt.total\_seconds())

the\_plot.set\_xlabel('Position')

the\_plot.set\_ylabel('Lap time (s)')

the\_plot.set\_title('Relationship between Lap time and Position')

# \*\*Bar chart: Average lap time for each tire.\*\*

# In[16]:

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.barplot(data = data, x = 'Compound', y = data['LapTime'].dt.total\_seconds(), hue = 'Compound', palette = tire\_colors)

the\_plot.set\_xlabel('Tire')

the\_plot.set\_ylabel('Lap time (s)')

the\_plot.set\_title('Relationship between Lap time and Tire')

# \*\*Line chart: Relationship between lap time and lap number.\*\*

# In[17]:

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.lineplot(data = data, x = 'LapNumber', y = data['LapTime'].dt.total\_seconds())

the\_plot.set\_xlabel('Lap')

the\_plot.set\_ylabel('Lap time (s)')

the\_plot.set\_title('Relationship between Lap time and Lap')

# \*\*Scatter plot: Relationship between second sector times and speed at the second detection point by tire.\*\*

# In[18]:

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.scatterplot(data = data, x = 'Sector2Time', y = 'SpeedI2', hue = 'Compound', palette = tire\_colors);

the\_plot.set\_ylabel('Speed at second point')

the\_plot.set\_xlabel('Time in second sector (s)')

the\_plot.set\_title('Relationship between second sector time and speed at second point')

# \*\*Adding the previous trend line\*\*

#

# The appropriate plot is the 'regplot'. This does not support the 'hue' parameter.

# In[19]:

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.regplot(data = data, x = data['Sector2Time'].dt.total\_seconds(), y = 'SpeedI2', line\_kws = {'color': 'red'});

the\_plot.set\_ylabel('Speed at second point')

the\_plot.set\_xlabel('Time in second sector (s)')

the\_plot.set\_title('Relationship between second sector time and speed at second point')

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Obtain the previous plot for the laps after 55.

# </div>

# In[20]:

# Your code here

laps\_after\_55 = data[data['LapNumber'] >= 55]

plt.figure(figsize=fig\_size)

the\_plot = sns.regplot(data = laps\_after\_55, x = laps\_after\_55['Sector2Time'].dt.total\_seconds(), y = 'SpeedI2', line\_kws = {'color': 'red'});

the\_plot.set\_ylabel('Speed at second point')

the\_plot.set\_xlabel('Time in second sector (s)')

the\_plot.set\_title('Relationship between second sector time and speed at second point for laps after 55')

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Create a chart with the average (maximum and minimum) lap time for each driver.

# </div>

# In[21]:

# Your code here

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.barplot(data = data, x = 'Driver', y = data['LapTime'].dt.total\_seconds(), hue='Driver', palette=driver\_colors)

the\_plot.set\_xlabel('Driver')

the\_plot.set\_ylabel('Lap Time')

the\_plot.set\_title('Relationship between Average Lap time and Driver')

plt.show()

# #### \*\*Relationships: \*Multiple variables\*\*\*

# \*\*Color and shape to represent more information.\*\*

# In[22]:

plt.figure(figsize = fig\_size); # Size of the plot

plt\_data = data.loc[data.Driver.isin(['ALO', 'BOT', 'PIA'])]

the\_plot = sns.scatterplot(data = plt\_data, x = 'Sector2Time', y = 'Sector3Time', hue = 'Driver', style = 'Compound', palette = driver\_colors)

the\_plot.set\_xlabel('Sector 2 Time')

the\_plot.set\_ylabel('Sector 3 Time')

the\_plot.set\_title('Relationship between Sector 3 and Sector 2 Times')

# \*\*Pairwise: Relationship between 'SpeedI1', 'SpeedI2', 'SpeedFL'.\*\*

# In[23]:

plt.figure(figsize = fig\_size); # Size of the plot

plt\_data = data.loc[data.Driver.isin(['ALO', 'BOT', 'PIA'])]

the\_plot = sns.pairplot(data = plt\_data, vars = ['SpeedI1', 'SpeedI2', 'SpeedFL'], hue = 'Driver')

# \*\*3D scatter. It is necessary to use the matplotlib library.\*\*

# In[24]:

from mpl\_toolkits import mplot3d # Import the 3D part of the matplotlib library

plt\_data = data.loc[data.Driver.isin(['ALO', 'BOT', 'PIA'])]

plt.figure(figsize = (10,12)); # Size of the plot

ax = plt.axes(projection = '3d') # Specify the plot type

ax.scatter3D(plt\_data['SpeedI1'], plt\_data['SpeedI2'], plt\_data['SpeedFL'], c = plt\_data['Driver'].apply(lambda x: driver\_colors[x]))

plt.xlabel('SpeedI1')

plt.ylabel('SpeedI2')

ax.set\_zlabel('SpeedFL')

plt.title('3D Scatter Plot')

# #### \*\*Relationships: \*Multiple variables with dimensionality reduction\*\*\*

#

# When working with data that has many dimensions (more than 2), it is difficult to visualize them directly. To understand their structure and find patterns, we need to reduce the number of dimensions without losing too much information. PCA and t-SNE are two methods that help us visualize high-dimensional data by reducing it to 2D or 3D.

#

# To use these methods, we need to first install the scikit-learn library:

# In[25]:

get\_ipython().system(' pip install scikit-learn')

# \*\*PCA: Visualize the laps of 3 drivers based on 6 variables: the time in each of the 3 sectors and the maximum speeds.\*\*

# In[26]:

from sklearn.decomposition import PCA # Import PCA

filtered\_data = data.loc[data.Driver.isin(['ALO', 'BOT', 'PIA'])].copy() # It's better to make a copy if modifications are to follow

# Convert sector times from timedelta to float

filtered\_data[['Sector1Time', 'Sector2Time', 'Sector3Time']] = filtered\_data[['Sector1Time', 'Sector2Time', 'Sector3Time']].apply(lambda x: x.dt.total\_seconds())

# Select the columns to represent

data\_6D = filtered\_data[['Sector1Time', 'Sector2Time', 'Sector3Time', 'SpeedI2', 'SpeedFL', 'SpeedST']]

pca = PCA(n\_components = 2, random\_state = 10) # Configure to reduce to 2 dimensions and set the seed.

new\_2D\_data = pca.fit\_transform(data\_6D.values) # Transform the data to 2 dimensions

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.scatterplot(x = new\_2D\_data[:,0], y = new\_2D\_data[:,1], hue = filtered\_data['Driver'], palette = driver\_colors); # Scatter plot

the\_plot.set\_xlabel('New Dimension X')

the\_plot.set\_ylabel('New Dimension Y')

the\_plot.set\_title('Dimensionality Reduction with PCA')

# \*\*t-SNE: Visualize the laps of 3 drivers based on 6 variables: the time in each of the 3 sectors and the maximum speeds.\*\*

# In[27]:

from sklearn.manifold import TSNE # Import tSNE

tsne = TSNE(n\_components = 2, random\_state = 10) # Configure to reduce to 2 dimensions and set the seed.

new\_2D\_data = tsne.fit\_transform(data\_6D.values) # Transform the data to 2 dimensions

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.scatterplot(x = new\_2D\_data[:,0], y = new\_2D\_data[:,1], hue = filtered\_data['Driver'], palette = driver\_colors); # Scatter plot

the\_plot.set\_xlabel('New Dimension X')

the\_plot.set\_ylabel('New Dimension Y')

the\_plot.set\_title('Dimensionality Reduction with t-SNE')

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Modify the previous plot so that each point is colored based on the tire instead of the driver.

# </div>

# In[36]:

# Your code here

from sklearn.manifold import TSNE # Import tSNE

tsne = TSNE(n\_components = 2, random\_state = 10) # Configure to reduce to 2 dimensions and set the seed.

new\_2D\_data = tsne.fit\_transform(data\_6D.values) # Transform the data to 2 dimensions

plt.figure(figsize = fig\_size); # Size of the plot

the\_plot = sns.scatterplot(x = new\_2D\_data[:,0], y = new\_2D\_data[:,1], hue = filtered\_data['Compound'], palette = tire\_colors); # Scatter plot

the\_plot.set\_xlabel('New Dimension X')

the\_plot.set\_ylabel('New Dimension Y')

the\_plot.set\_title('Dimensionality Reduction with t-SNE')

# ### \*\*2.2. Dynamic Visualization\*\*

# In this last section, we will use the Bokeh library.

#

# This library generates HTML code instead of producing a static image.

# In[29]:

from bokeh.plotting import figure, show, ColumnDataSource

from bokeh.io import output\_notebook

output\_notebook() # Configure Bokeh

# In[30]:

# Create a ColumnDataSource object that contains all the relevant data for the plot

source = ColumnDataSource(data = dict(

x = new\_2D\_data[:,0],

y = new\_2D\_data[:,1],

driver = filtered\_data['Driver'],

color = filtered\_data['Driver'].apply(lambda x: driver\_colors[x]),

lap\_time = filtered\_data['LapTime'].dt.total\_seconds(),

lap\_number = filtered\_data['LapNumber'],

compound = filtered\_data['Compound'],

tire\_life = filtered\_data['TyreLife'],

)

)

# Customize the information to display when hovering the mouse

TOOLTIPS = """

<div>

<div style="padding-right:5px;padding-left:5px">

<span style="font-size: 20px; color: @color;"><strong>@driver</strong></span>

<ul style="font-size: 15px; margin-left:-20px; margin-top:0px">

<li><strong>Lap: </strong>@lap\_number</li>

<li><strong>Tire: </strong>@compound</li>

<li><strong>Lap Time: </strong>@lap\_time</li>

<li><strong>Tyre life: </strong>@tire\_life</li>

</ul>

</div>

</div>

"""

# Create the plot

p = figure(width = 1000, height = 600, tooltips = TOOLTIPS)

p.scatter(source = source, x = 'x', y = 'y', fill\_color = 'color', line\_color = None, size = 7)

show(p)

# <div class="alert alert-block alert-info">

# <b>Exercise:</b> Modify the previous plot so that each point is colored based on the tire instead of the driver.

# </div>

# In[31]:

# Create a ColumnDataSource object that contains all the relevant data for the plot

source = ColumnDataSource(data = dict(

x = new\_2D\_data[:,0],

y = new\_2D\_data[:,1],

driver = filtered\_data['Driver'],

color = filtered\_data['Compound'].apply(lambda x: tire\_colors[x]),

lap\_time = filtered\_data['LapTime'].dt.total\_seconds(),

lap\_number = filtered\_data['LapNumber'],

compound = filtered\_data['Compound'],

tire\_life = filtered\_data['TyreLife'],

)

)

# Customize the information to display when hovering the mouse

TOOLTIPS = """

<div>

<div style="padding-right:5px;padding-left:5px">

<span style="font-size: 20px; color: @color;"><strong>@driver</strong></span>

<ul style="font-size: 15px; margin-left:-20px; margin-top:0px">

<li><strong>Lap: </strong>@lap\_number</li>

<li><strong>Tire: </strong>@compound</li>

<li><strong>Lap Time: </strong>@lap\_time</li>

<li><strong>Tyre life: </strong>@tire\_life</li>

</ul>

</div>

</div>

"""

# Create the plot

p = figure(width = 1000, height = 600, tooltips = TOOLTIPS)

p.scatter(source = source, x = 'x', y = 'y', fill\_color = 'color', line\_color = None, size = 7)

show(p)

# In[ ]: