#!/usr/bin/env python

# coding: utf-8

# # \*\*Lab 2.2: Preprocessing\*\*

#

# <hr>

#

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

# In this lab, we will download a dataset, analyze it, and clean it for use in future labs.

#

# ### \*\*Pandas Library\*\*

# It is considered the most popular data analysis library in Python.

# It performs all its operations using a \*"DataFrame"\* object.

#

# It allows, among other operations:

#

# \* Loading and storing data in different formats (csv, tsv, xlsx, txt...).

# \* Manipulating rows, columns, and cells.

# \* Filtering or grouping content.

# \* Performing intersection, concatenation, or merging of multiple DataFrames.

#

# To install it:

# In[1]:

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

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

# <strong>NOTE:</strong> The exclamation mark before the code indicates to Jupyter that this is not Python and should be executed in the terminal. This allows us to install libraries directly from the notebook.

# </div>

# <hr>

#

# ## \*\*2. Exploratory Data Analysis (EDA)\*\*

#

# The main objectives of this analysis are:

# \* Understand the data we will be working with.

# \* Clean the dataset:

# \* Remove empty rows or columns.

# \* Eliminate inconsistent values.

#

# Our dataset contains information about a race from the 2023 Formula 1 season.

# Next, we will download the data and analyze it using the Pandas library.

# In[2]:

import pandas as pd

url\_data = 'https://raw.githubusercontent.com/AIC-Uniovi/Sistemas-Inteligentes/refs/heads/main/datasets/f1\_23\_monaco.csv'

data = pd.read\_csv(url\_data)

# \*\*Description of Dataset Columns\*\*

#

# | Column | Description |

# |----------------------|------------|

# | `Time` | Total elapsed time in the session. |

# | `Driver` | Three-letter driver code. |

# | `DriverNumber` | Driver's race number. |

# | `LapTime` | Total lap time. |

# | `LapNumber` | Lap number in the session. |

# | `Stint` | Current stint number (period between pit stops). |

# | `PitOutTime` | Time when the driver exited the pit lane. |

# | `PitInTime` | Time when the driver entered the pit lane. |

# | `Sector1Time` | Time recorded in the first sector of the lap. |

# | `Sector2Time` | Time recorded in the second sector of the lap. |

# | `Sector3Time` | Time recorded in the third sector of the lap. |

# | `SpeedI1` | Speed measured at the first detection point. |

# | `SpeedI2` | Speed measured at the second detection point. |

# | `SpeedFL` | Speed at the finish line. |

# | `SpeedST` | Maximum speed in the sector. |

# | `IsPersonalBest` | Indicates if the lap is the driver's personal best (`True`/`False`). |

# | `Compound` | Type of tire compound used. |

# | `TyreLife` | Number of laps the tire has been used. |

# | `FreshTyre` | Indicates if the tire was new at the start of the lap (`True`/`False`). |

# | `Team` | Driver's team name. |

# | `LapStartTime` | Lap start time within the session. |

# | `LapStartDate` | Exact date and time of lap start. |

# | `TrackStatus` | Track condition during the lap (e.g., yellow flag, green flag, etc.). |

# | `Position` | Driver's position at the end of the lap. |

# | `Deleted` | Indicates if the lap was deleted (`True`/`False`). |

# | `DeletedReason` | Reason why the lap was deleted (if applicable). |

# | `IsAccurate` | Indicates if the lap data is accurate (`True`/`False`). |

# ### \*\*Basic Operations\*\*

# In[3]:

data

# In[4]:

# Column names

data.columns

# In[5]:

# Column types

data.dtypes

# In[6]:

# Number of columns

len(data.columns)

# In[7]:

# Number of rows

len(data)

# In[8]:

# Get basic statistics for the entire dataset

data.describe()

# In[9]:

# Find columns with missing values

data.isnull().any()

# In[10]:

# Display the first 5 rows

data.head(5)

# In[11]:

# Display the last 5 rows

data.tail(5)

# In[12]:

# Access a specific column

data['Driver']

# In[13]:

# Get multiple statistics for a column

data['Stint'].describe()

# In[14]:

# Operations on numeric columns

data['LapNumber'] + 1

# In[15]:

# View unique (non-repeating) values of a column

data['Team'].unique()

# In[16]:

len(data['Team'].unique())

# In[17]:

# Access multiple columns

data[['Driver', 'Team']]

# In[18]:

# Get a list of values from a column and access aneElement

data['Team'].values[180]

# In[19]:

# Access row 1280, column 1 (zero-indexed)

data.iloc[1280, 1]

# In[20]:

# Sort by the 'LapTime' column

data.sort\_values(['LapTime'])

# <div class='alert alert-block alert-warning'>

# <strong>NOTE:</strong> The previous operations are not 'inplace', meaning they do not modify the DataFrame, they only query it.

# </div>

# In[21]:

# Get the maximum, mean, and minimum number of laps a set of tires was used

mean\_life = data['TyreLife'].mean()

min\_life = data['TyreLife'].min()

max\_life = data['TyreLife'].max()

print(min\_life, mean\_life, max\_life)

# In[22]:

# Change the type of a series of columns to int

data[['DriverNumber', 'LapNumber', 'Stint', 'TyreLife', 'TrackStatus', 'Position']] = data[['DriverNumber', 'LapNumber', 'Stint', 'TyreLife', 'TrackStatus', 'Position']].astype(int)

# In[23]:

# Add a new column

data['New\_Column\_1'] = 1 # All rows will have the same value

data['New\_Column\_2'] = list(range(len(data))) # New column created from a list of values (as many as rows)

data['New\_Column\_3'] = data['Stint'] + 1 # New column created from another column

# In[24]:

# Delete columns

data = data.drop(columns = ['New\_Column\_1', 'New\_Column\_2', 'New\_Column\_3'])

# This is equivalent to:

# data.drop(columns = ['New\_Column\_1', 'New\_Column\_2', 'New\_Column\_3'], inplace = True)

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

# <b>Exercise:</b> Get the number of unique drivers.

# </div>

# In[25]:

# Your code here

len(data['Driver'].unique())

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

# <b>Exercise:</b> Change the type of the columns: 'Time', 'LapTime', 'PitOutTime', 'PitInTime', 'Sector1Time', 'Sector2Time', 'Sector3Time', and 'LapStartTime' to <a href="https://pandas.pydata.org/docs/reference/api/pandas.to\_timedelta.html"><i>timedelta</i></a>.

# </div>

# In[26]:

time\_columns = ['Time', 'LapTime', 'PitOutTime', 'PitInTime', 'Sector1Time', 'Sector2Time', 'Sector3Time', 'LapStartTime']

# Your code here

for col in time\_columns:

data[col] = pd.to\_timedelta(data[col])

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

# <b>Exercise:</b> Change the type of the column: 'LapStartDate' to <a href="https://pandas.pydata.org/docs/reference/api/pandas.to\_datetime.html"><i>datetime</i></a>.

# </div>

# In[27]:

# Your code here

data['LapStartDate']=pd.to\_datetime(data['LapStartDate'])

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

# <b>Exercise:</b> Which driver has been the best in the first sector? And in the second sector?

# </div>

# In[28]:

# Your code here

data.sort\_values("Sector1Time").Driver.values[0]

# In[29]:

# Your code here

data.sort\_values("Sector2Time").Driver.values[0]

# ### \*\*Data Filtering\*\*

# In[30]:

# Get the value of a specific cell

data.loc[572, 'Team']

# In[31]:

# Get the laps of drivers whose team is 'Ferrari'

data\_ferrari = data.loc[data['Team'] == 'Ferrari']

data\_ferrari

# In[32]:

# Get all laps 1 and 2 of the drivers

data.loc[data['LapNumber'] <= 2]

# In[33]:

# Get lap 10 of the drivers from 'Ferrari'

data\_ferrari\_10 = data.loc[(data['LapNumber'] == 10) & (data['Team'] == 'Ferrari')]

data\_ferrari\_10

# In[34]:

# Get the laps of the drivers 'SAI' or 'LEC'

data.loc[(data['Driver'] == 'SAI') | (data['Driver'] == 'LEC')]

# In[35]:

# Another option for getting the laps of the drivers 'SAI' or 'LEC'

data.loc[data['Driver'].isin(['SAI', 'LEC'])]

# In[36]:

# Get laps from teams that contain 'Bull'

data.loc[data['Team'].str.contains('Bull')]

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

# <b>Exercise:</b> Fix the NaT values in the times of the columns 'Sector1Time' and 'LapTime'.

# </div>

# In[37]:

# Your code here

#print(data["Sector1Time"].isna )

#print(data["LapTime"].isna)

#Solucion dada mal

#data["Sector1Time"]=data["LapTime"]-data["Sector2Time"]-data["Sector3Time"]

#data.loc[572,'LapTime']=data.loc[572, "Sector1Time"] + data.loc[572, "Sector2Time"] + data.loc[572, "Sector3Time"]

#Solución real

aux=data[data['Sector1Time'].isna()]

for index, rows in aux.iterrows():

data.at[index, "Sector1Time"]=rows["LapTime"]-rows["Sector2Time"]-rows["Sector3Time"]

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

# <b>Exercise:</b> Get the average lap time of 'AlphaTauri' drivers between laps 1 and 20 (inclusive).

# </div>

# In[38]:

# Your code here

aux=data.loc[(data["Team"]=="AlphaTauri") & (data["LapNumber"]<=20) & (data["LapNumber"]>=1)]

print(aux["LapTime"].mean())

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

# <b>Exercise:</b> What was the maximum speed at the finish line for Alonso? And for Verstappen? In which laps?

# </div>

# In[39]:

alonso\_speed = data.loc[data["Driver"] == "ALO"]

alonso\_maxSP = alonso\_speed["SpeedFL"].max()

alonso\_aux\_lap = data.loc[(data["Driver"] == "ALO") & (data["SpeedFL"] == alonso\_maxSP)]

alonso\_lap = alonso\_aux\_lap["LapNumber"].values[0]

print(f"Alonso speed: {alonso\_maxSP}, Alonso lap: {alonso\_lap}")

# In[40]:

VER\_speed = data.loc[data["Driver"] == "VER"]

VER\_maxSP = VER\_speed["SpeedFL"].max()

VER\_aux\_lap = data.loc[(data["Driver"] == "VER") & (data["SpeedFL"] == VER\_maxSP)]

VER\_lap = alonso\_aux\_lap["LapNumber"].values[0]

print(f"VER speed: {VER\_maxSP}, VER lap: {VER\_lap}")

# ### \*\*Data Grouping\*\*

# In[41]:

# Get the number of drivers per team

data.groupby('Team')['Driver'].nunique().reset\_index()

# In[42]:

# List of drivers by team

data.groupby('Team')['Driver'].unique().reset\_index()

# In[43]:

# Another option to get the list of drivers by Team

data.groupby('Team')['Driver'].apply(lambda x: list(set(x))).reset\_index()

# In[44]:

# Number of laps per driver ordered from highest to lowest

data.groupby('Driver')['LapNumber'].max().sort\_values(ascending = False).reset\_index()

# In[45]:

# Another option to get the Number of laps per driver ordered from highest to lowest

data.groupby('Driver')['LapNumber'].size().sort\_values(ascending = False).reset\_index()

# In[46]:

# Average speed at the finish line for each team

data.groupby('Team')['SpeedFL'].mean().sort\_values(ascending = False).reset\_index()

# In[47]:

# Another option that allows customizing the name of the new column and creating multiple at once

data.groupby('Team').agg(AvgFlSpeed = ('SpeedFL', 'mean')).sort\_values('AvgFlSpeed', ascending = False).reset\_index()

# In[48]:

# The Pivot Table allows grouping data in a more complex way.

# In this example, it shows for each Driver from each Team, the Number of Laps they did with each compound, as well as the totals for rows and columns (margins).

data.pivot\_table(index = ['Team', 'Driver'], columns = ['Compound'], values = 'LapNumber', aggfunc = 'count', fill\_value = 0, margins = True, margins\_name = 'Total')

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

# <b>Exercise:</b> Get the number of laps annulled for each driver. Order from highest to lowest.

# </div>

# In[49]:

# Your code here

#data.groupby('Driver')['LapNumber'].max().sort\_values(ascending = False).reset\_index()

laps\_annulled\_by\_driver = data.groupby('Driver')['Deleted'].sum().reset\_index()

laps\_annulled\_by\_driver = laps\_annulled\_by\_driver.sort\_values(by='Deleted', ascending=False).reset\_index(drop=True)

laps\_annulled\_by\_driver

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

# <b>Exercise:</b> Get the number of pit stops made by each driver.

# </div>

# In[50]:

# Your code here

#data.groupby('Driver')['LapNumber'].max().sort\_values(ascending = False).reset\_index()

pitstops = data.dropna(subset=['PitOutTime', 'PitInTime'])

pitstops\_driver=pitstops.groupby('Driver').size().reset\_index(name='Pitstops').sort\_values(by='Pitstops', ascending=False)

pitstops\_driver

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

# <b>Exercise:</b> Create a table where the rows show the teams and drivers, and the columns represent the first 10 laps. The lap time should be displayed in seconds (using time.dt.total\_seconds()).

# </div>

# In[51]:

# Your code here

first\_10\_laps = data[data['LapNumber'] <= 10]

first\_10\_laps['LapTimeSeconds'] = first\_10\_laps['LapTime'].dt.total\_seconds()

lap\_times\_table = first\_10\_laps.pivot\_table(

index=['Team', 'Driver'],

columns='LapNumber',

values='LapTimeSeconds',

aggfunc='mean'

)

lap\_times\_table

# ### \*\*Final Cleanup and Storage of the DataFrame\*\*

#

# To ensure that this dataset is ready for future use in practice, we will remove certain rows and columns that won't contribute useful information to the problems we'll solve.

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

# <b>Exercise:</b> Remove all rows where 'TrackStatus' is not equal to 1 and those that correspond to a pit stop. The latter will have a value in 'PitOutTime' or 'PitInTime'. Sort by 'Time' in ascending order and reset the index using `reset\_index(drop=True)` to reassign the row indices.

# </div>

# In[52]:

# Your code here

data=data[data['TrackStatus']== 1 & data['PitInTime'].isnull() & data['PitOutTime'].isnull()]

data.sort\_values('Time', ascending=True).reset\_index(drop=True)

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

# <b>Exercise:</b> Finally, remove the columns "Deleted", "DeletedReason", "IsAccurate", "TrackStatus", "PitOutTime", and "PitInTime".

# </div>

# In[53]:

# Your code here

cols = ["Deleted", "DeletedReason", "IsAccurate", "TrackStatus", "PitOutTime", "PitInTime"]

data.drop(columns=cols)

# Once this phase of analysis and cleaning of the dataset is completed, we will store the Pandas DataFrame in a `Pickle` file.

# It is possible to store it as `CSV` or `XLSX`, but these formats do not preserve the column types. This would mean that when loading it in the future, we would have to re-cast (`.astype()`) each column.

# In[54]:

data.to\_pickle('f1\_23\_monaco.pkl')