TP FINAL

Ejercicio 1
Buscar un dataset (Que no sea los trabajados en clases)

Ejercicio 2
Realizar una introduccion al dataset de que se trata,definir sus variables (Diccionario de datos)

Ejercicio 3
Identificar el tipo de variable,decide justificando su respuesta.

Ejercicio 4
Detectar Valores Ausente y Valores Atipicos .Decidir si eliminarlos y el por que de la eleccion.

Ejercicio 5
Realizar un analisis univariado y en base a esos graficos,sacar conclusiones.

Ejercicio 6
Realizar analisis de matriz corelacion y explicar que variable estan correlacionadas\

Ejercicio 7
Sobre el Dataset Elegido explique si se puede reducir las dimensiones y que representa esas nuevas variables.

TP FINAL Primera parte

Vamos a cubrir los primeros 4 puntos del tp.

Elegimos el data set de la formula 1.

Lo saque de Kaggle.

```
#librerias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
```

```
#Datasetelegido
formula1 = pd.read_excel('/content/Formula1.xlsx')
formula1
```

	resultId	raceId	driverId	constructorId	number	grid	position	position
0	1	18	1	1	22.0	1	1	
1	2	18	2	2	3.0	5	2	
2	3	18	3	3	7.0	7	3	
3	4	18	4	4	5.0	11	4	
4	5	18	5	1	23.0	3	5	
...	...	...	...	...	...	...	...	
25655	25661	1086	825	210	20.0	13	16	
25656	25662	1086	848	3	23.0	17	17	
25657	25663	1086	849	3	6.0	19	18	
25658	25664	1086	852	213	22.0	16	19	
25659	25665	1086	822	51	77.0	8	20	

25660 rows × 21 columns

Antes de describir las variables vamos a hacer un proceso de limpieza y ordenamiento del data set para el trabajo

```
#Usamos esto para ver mas detallado las variables
```

```
Lista_variables = formula1
print(Lista_variables.columns.tolist())
```

```
['resultId', 'raceId', 'driverId', 'constructorId', 'number', 'grid', 'position', 'positionText', 'positionOrder', 'points', 'laps'
```

```
#vamos a limpiar datos nulos.
#Vamos a corroborar si tiene datos nulos
formula1_2 = formula1.dropna(inplace=True)
print(formula1_2)
```

```
None
```

```
#Otra forma de indentificarlo
null_counts = formula1.isna().sum()
print(null_counts)
```

```
resultId          0.0
raceId            0.0
driverId          0.0
constructorId     0.0
number            0.0
grid              0.0
position          0.0
positionText      0.0
positionOrder     0.0
points            0.0
laps              0.0
time              0.0
milliseconds      0.0
fastestLap        0.0
rank              0.0
fastestLapTime    0.0
fastestLapSpeed   0.0
statusId          0.0
Unnamed: 18       0.0
Unnamed: 19       0.0
Unnamed: 20       0.0
dtype: float64
```

Como no tenemos columnas con datos nulos pero si tenemos 3 columnas las cuales vamos a borrar porque estan vacias que son las Unnamed: 18 Unnamed: 19 Unnamed: 20

```
quitar_columnas = ['Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20']
formula1 = formula1.drop(columns=quitar_columnas)
formula1
```

	resultId	raceId	driverId	constructorId	number	grid	position	position
0	1	18	1	1	22.0	1	1	
1	2	18	2	2	3.0	5	2	
2	3	18	3	3	7.0	7	3	
3	4	18	4	4	5.0	11	4	
4	5	18	5	1	23.0	3	5	
...	...	...	...	...	...	...	...	
25655	25661	1086	825	210	20.0	13	16	
25656	25662	1086	848	3	23.0	17	17	
25657	25663	1086	849	3	6.0	19	18	
25658	25664	1086	852	213	22.0	16	19	
25659	25665	1086	822	51	77.0	8	20	

25660 rows × 18 columns

```
print(formula1.dtypes)
```

```
resultId            int64
raceId              int64
driverId            int64
constructorId       int64
number            float64
grid                int64
position           object
positionText       object
positionOrder       int64
points              int64
```

```
laps                int64
time               object
milliseconds       object
fastestLap         object
rank               object
fastestLapTime     object
fastestLapSpeed    object
statusId            int64
dtype: object
```

```
print(formula1.describe())
```

```
            resultId        raceId      driverId  constructorId        number  \
count  25660.000000  25660.000000  25660.000000   25660.000000  25654.000000
mean   12831.305105    527.495830    258.558496      48.299532     17.740742
std     7408.686039    296.776908    265.708300      59.354624     15.042254
min        1.000000      1.000000      1.000000       1.000000      0.000000
25%     6415.750000    291.000000     56.000000       6.000000      7.000000
50%    12830.500000    511.000000    160.000000      25.000000     15.000000
75%    19245.250000    777.000000    358.000000      58.000000     24.000000
max    25665.000000   1086.000000    855.000000     214.000000    208.000000

                grid  positionOrder        points          laps      statusId
count  25660.000000   25660.000000  25660.000000  25660.000000  25660.000000
mean      11.187256      12.892673      1.989049     45.936204     17.634528
std        7.251983       7.721729      8.479032     29.867844     26.161012
min        0.000000       1.000000      0.000000      0.000000      1.000000
25%        5.000000       6.000000      0.000000     22.000000      1.000000
50%       11.000000      12.000000      0.000000     52.000000     11.000000
75%       17.000000      18.000000      2.000000     66.000000     14.000000
max       34.000000      39.000000    814.000000    200.000000    141.000000
```

Descripcion de las variables (son un total de 18):

resultId (int64): Esta columna es un identificador único para los resultados.

raceId (int64):Esta columna es un identificador único para las carreras.

driverId (int64): Esta columna es un identificador para los pilotos

constructorId (int64): Esta columna es un identificador para los constructures de la Formula1 (ferrari, redbull, etc.)

number (float64): Esta columna es un identificador para los autos.

grid (int64): Esta columna representa la posición en la que los autos comenzaron la carrera en la parrilla.

position (object): Al ser una columna que posiblemente almacena información variada como la posición de un auto (por ejemplo, "1", "2", "DNF")

positionText (object): Similar a position, es probable que almacene información diversa relacionada con la posición

positionOrder (int64): Esta columna representa el orden de posicion de los autos

points (int64): Esta columna muestra el valor de los puntos obtenidos mediante el resultado final de las carreras.

laps (int64): Esta columna representa la cantidad de vueltas dentro de la pista en la que hayan corrido.

time (object): Esta columna almacena el tiempo que le saco por vuelta un piloto a otro.

milliseconds (object): Similar a time, esta columna también parece contener información de tiempo en milisegundos.

fastestLap (object): Esta columna contiene informacion sobre las vueltas mas rapidas.

rank (object): Esta columna contiene informacion del rango en el que terminaron los pilotos.

fastestLapTime (object): Almacena el tiempo de la vuelta más rápida.

fastestLapSpeed (object): Puede contener información de velocidad en forma de número o texto, por lo que se considera como un tipo de objeto.

statusId (int64): Parecido a la columna rank contiene el resultado final de pilotos en las carreras.

En resumen: tenemos un total de 18 variables

1. Variables numericas:resultd,raceid,driverid,constructorid,number,grid,positionorder,points,laps,statusid.
2. Variables categoricas:position,positiontext,time,miliseconds,fastestlap,rank,fastestlaptime,fastestlapseed.

Valores ausentes ya corrobaramos que no tiene y borramos las columnas necesarias.

Valores Atipicos: vamos a corroborar una de las columnas, en este caso la columna "points"

```
# Calcular los cuantiles y el rango intercuartil (IQR)
q1 = formula1['points'].quantile(0.25)
q3 = formula1['points'].quantile(0.75)
```

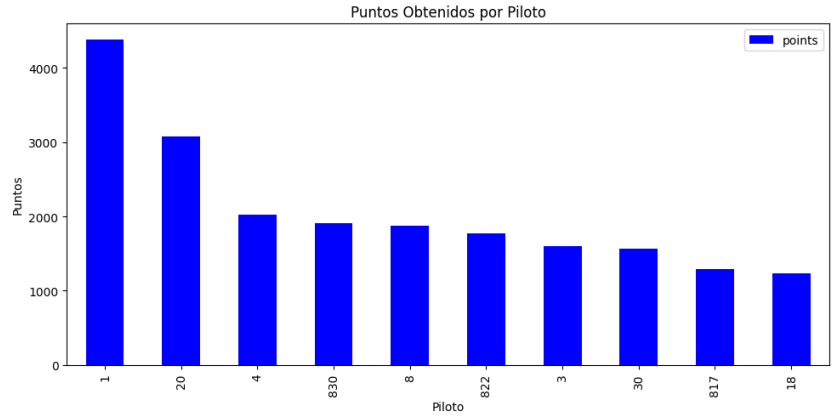
```
iqr = q3 - q1

# Calcular los límites para identificar valores atípicos
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Identificar valores atípicos
outliers = formula1[(formula1['points'] < lower_bound) | (formula1['points'] > upper_bound)]
print(outliers)
```

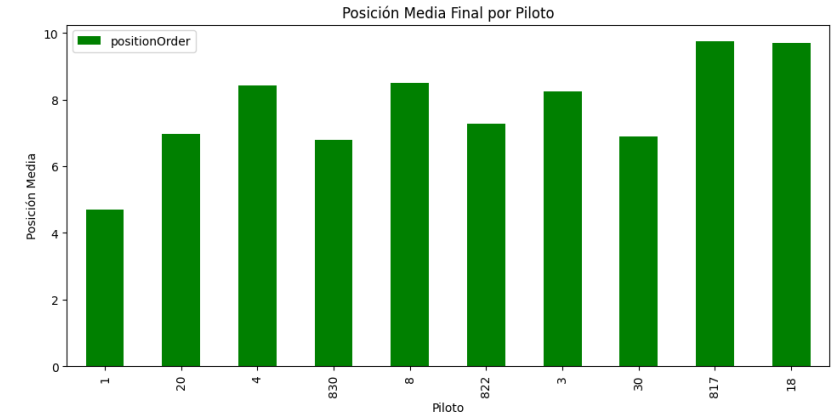
Lo que pude lograr entender con este analisis es que no debemos de borrar ni modificar el resultado del mismo.

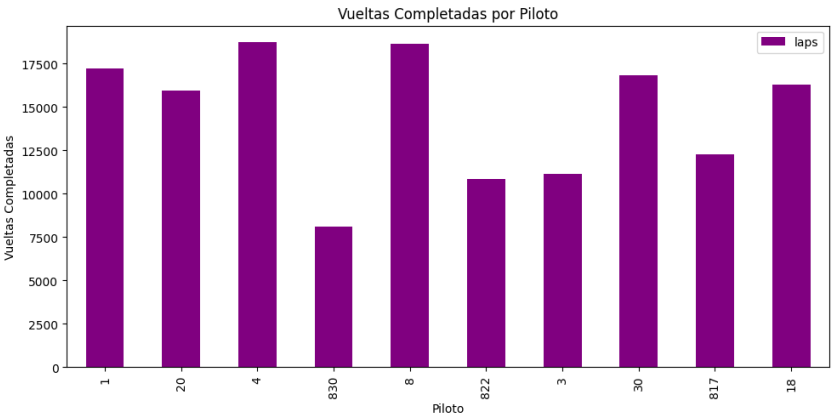
Segunda parte del tp final

En esta parte vamos a cubrir todo lo que tenga que ver con los graficos y sus analisis, cubriendo asi los puntos 5,6 y 7.

```
# Agrupar por piloto y calcular las estadísticas
pilotos = formula1.groupby('driverId').agg({
    'points': 'sum',
    'positionOrder': 'mean',
    'laps': 'sum'
}).reset_index()

# Ordenar por puntos en orden descendente
pilotos = pilotos.sort_values(by='points', ascending=False)

# Mostrar solo los primeros N pilotos con más puntos
N = 10
Mejores_pilotos = pilotos.head(N)

# Gráfico de barras comparando puntos, posición media y vueltas completadas por piloto
fig, axes = plt.subplots(3, 1, figsize=(10, 15))
Mejores_pilotos.plot(x='driverId', y='points', kind='bar', ax=axes[0], color='blue')
Mejores_pilotos.plot(x='driverId', y='positionOrder', kind='bar', ax=axes[1], color='green')
Mejores_pilotos.plot(x='driverId', y='laps', kind='bar', ax=axes[2], color='purple')

axes[0].set_title('Puntos Obtenidos por Piloto')
axes[0].set_ylabel('Puntos')
axes[0].set_xlabel('Piloto')

axes[1].set_title('Posición Media Final por Piloto')
axes[1].set_ylabel('Posición Media')
axes[1].set_xlabel('Piloto')

axes[2].set_title('Vueltas Completadas por Piloto')
axes[2].set_ylabel('Vueltas Completadas')
axes[2].set_xlabel('Piloto')

plt.tight_layout()
plt.show()
```







Podemos observar que en el eje X se encuentran los pilotos, dado que en el dataframe elegido no pudimos obtener los nombres de los mismos.

Recordemos lo siguiente en la formula 1 la distribucion de puntos es la siguiente:

1. 1er lugar son 25 puntos.
2. 2do lugar son 18 puntos.
3. 3er lugar son 15 puntos.
4. 4to ugar son 10 puntos.
5. 5to lugar son 8 puntos.
6. 6to lugar son 6 puntos.
7. 7mo lugar son 5 puntos.
8. 8vo lugar son 3 puntos.
9. 9no lugar son 2 puntos.
10. 10mo lugar son 1 punto.

Aclaracion: si alguno de ellos logra quedarse con la vuelta rapida tendran 1 punto mas, solo aquellos que hayan terminado entre la posicion 1 al 10.

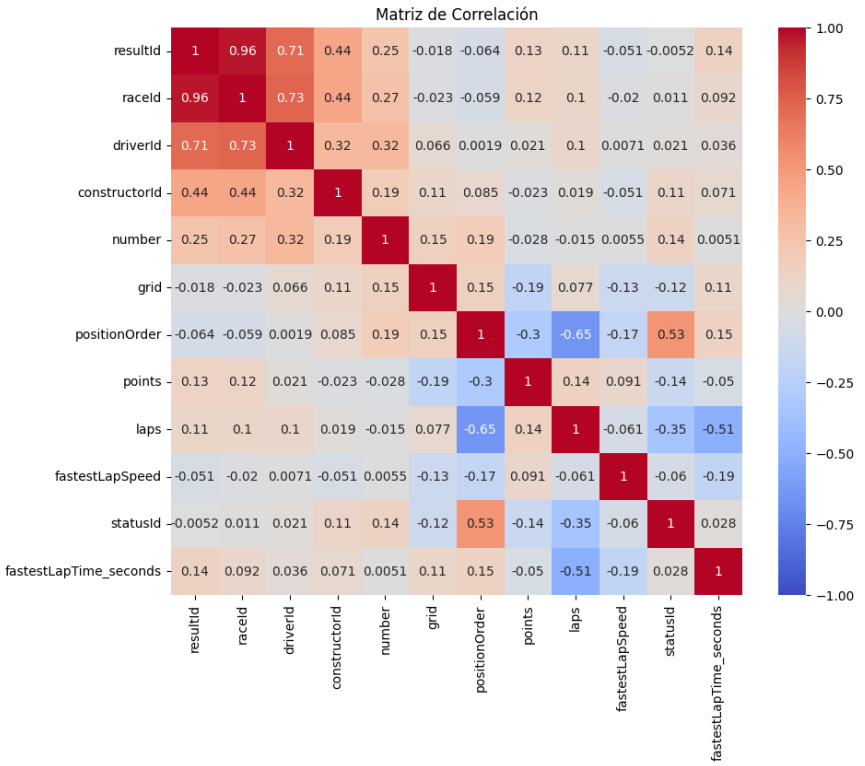
En el grafico que muestra las barras azules podemos ver los puntos obtenidos por los pilotos donde el piloto "1" es quien obtiene mas puntos.

En el grafico que muestra las barras verdes podemos ver la posicion media final por piloto donde el piloto 817 y 18 son quienes obtienen las peores posiciones dado que terminar en una posicion media cercana al 10 es en el ultimo lugar donde se obtienen puntos.

En el ultimo grafico de las barras violetras podemos observar las vueltas completadas por piloto donde los pilotos 4 y 8 son quienes han hecho mas vueltas que el resto, esto no significa en la FORMULA 1 que quienes tengan mas vueltas completas sean aquellos que saquen mas puntos o ganen mas carreras ya que eso es dependiendo de otros factores.

```
# Calcular la matriz de correlación
correlation_matrix = formula1.corr()

# Crear un mapa de calor (heatmap) de la matriz de correlación
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Matriz de Correlación')
plt.show()
```

```
<ipython-input-41-c4fb30f8b24f>:2: FutureWarning: The default value of numeric_only i
  correlation_matrix = formula1.corr()
```



Para interpretar la matriz de correlación:

Si un valor es cercano a 1, indica una correlación positiva fuerte. Si un valor es cercano a -1, indica una correlación negativa fuerte. Si un valor es cercano a 0, indica una correlación débil o nula.

Por lo tanto en nuestra matriz de correlacion podemos observar que las variables de resultID, raceID y driverID son aquellas que mejor relacionadas estan, lo identificamos con el color de la matriz donde mas rojo se torne el color del mapa mas fuerte se muestra la relacion.

En el dataframe elegido se puede reducir dimensiones y para ello use el metodo de, PCA (Análisis de Componentes Principales):

PCA es una técnica ampliamente utilizada para reducir dimensiones. Transforma las variables originales en un nuevo conjunto de variables no correlacionadas (componentes principales) que explican la mayor varianza en los datos.

```
# Seleccionar las variables numéricas para el PCA
```

```
numeric_columns = ['resultId', 'raceId', 'driverId','constructorId','points', 'laps','statusId']

# Crear un DataFrame con las variables numéricas
numeric_df = formula1[numeric_columns]

# Normalizar los datos
scaler = StandardScaler()
scaled_numeric_df = scaler.fit_transform(numeric_df)

# Aplicar PCA
pca = PCA()
pca_result = pca.fit_transform(scaled_numeric_df)

# Crear un DataFrame para los resultados de PCA
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i+1}' for i in range(len(numeric_columns))])

# Concatenar los resultados de PCA con el DataFrame original
final_df = pd.concat([formula1, pca_df], axis=1)

# Mostrar el porcentaje de varianza explicada por cada componente
explained_variance_ratio = pca.explained_variance_ratio_
print("Porcentaje de Varianza Explicada por Cada Componente:")
print(explained_variance_ratio)
```

```
Porcentaje de Varianza Explicada por Cada Componente:
[0.41456838 0.20758458 0.13102738 0.1051829  0.08926199 0.04741944
 0.00495533]
```

El resultado final explica lo siguiente:

El primer componente principal (PC1) explica aproximadamente el 41.46% de la varianza total en tus datos.

El segundo componente principal (PC2) explica aproximadamente el 20.76% de la varianza total.

El tercer componente principal (PC3) explica alrededor del 13.10% de la varianza total.

El cuarto componente principal (PC4) explica alrededor de un 10.51% de la varianza total.

El quinto componente principal (PC5) explica alrededor de un 8.92% de la varianza total.

El sexto componente principal (PC6) explica alrededor de un 4.47% de la varianza total.

Por ulimo el septimo componente principal (PC7) explica alrededor de un valor proximo al 0.5& de la varianza total.

Esto muestra cuanto de la varianza total esta siendo explicada por cada componente principal.

La importancia de estos valores radica en entender cuánta información está siendo retenida cuando se reducen las dimensiones de los datos con el PCA. El objetivo del PCA es capturar la mayor cantidad posible de varianza con el menor número de componentes principales. Al observar el porcentaje de varianza explicada, puedes tomar decisiones informadas sobre cuántos componentes retener:

Si se retiensolo unos pocos componentes principales que explican una gran parte de la varianza total, estás reduciendo la dimensionalidad de los datos mientras retienes la mayor cantidad de información relevante.

Si retienes más componentes, estás conservando más detalles, pero también puedes estar reteniendo más ruido en los datos.

En conclusion podemos decir que los primeros 2 componentes es donde mas % se obtienen por ende donde tenemos mas informacion dado que luego vemos como va descendiendo hasta el punto que la Componente 7 no tiene casi informacion.