2.3.1 - Entendiendo los datos

December 20, 2020

1 Análisis exploratorio de los datos

(Fuente:https://www.proglobalbusinesssolutions.com/six-steps-in-crisp-dm-the-standard-data-mining-process/)

La recopilación inicial de datos y cifras se realiza a partir de todas las fuentes disponibles. En la fase para **entender los datos** se examinan las propiedades del set de datos que se tiene. Luego, la calidad de la información se verifica mediante respuestas a ciertas preguntas relevantes sobre la integridad y precisión del material.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

1.1 Carga de datos

[6]:
```
# Carga de datos
data_frame = pd.read_csv("Food_Preference.csv")
data_frame.head()
```

[6]:
```
                    Timestamp Participant_ID  Gender Nationality  Age  \
0  2019/05/07 2:59:13 PM GMT+8         FPS001    Male      Indian   24
1  2019/05/07 2:59:45 PM GMT+8         FPS002  Female      Indian   22
2  2019/05/07 3:00:05 PM GMT+8         FPS003    Male      Indian   31
3  2019/05/07 3:00:11 PM GMT+8         FPS004  Female      Indian   25
4  2019/05/07 3:02:50 PM GMT+8         FPS005    Male      Indian   27

                Food              Juice Dessert
0  Traditional food        Fresh Juice   Maybe
1      Western Food  Carbonated drinks     Yes
2      Western Food        Fresh Juice   Maybe
3  Traditional food        Fresh Juice   Maybe
4  Traditional food        Fresh Juice   Maybe
```

1.1.1 Consideraciones

1. El set de datos está formado por filas y columnas. Las filas corresponden a las observaciones y las columnas son las características (features, variables).
2. Cada característica puede ser numérica o catgórica; particularmente, muchos algoritmos requieren que las columnas con las cuales se quiere trabajar deben ser numéricas. Lo anterior lleva a un proceso, eventualmente, de transformación de los datos.

1.2 Es hora de comenzar con la exploración de los datos ...

1.3 Tamaño del dataset

[15]:
```
data_frame.shape
```

[15]:
```
(288, 8)
```

1.4 Tipos de datos de las columnas

[10]:
```
# Revisión de los tipos de datos
data_frame.dtypes
```

[10]:
```
Timestamp         object
Participant_ID    object
```

```
Gender            object
Nationality       object
Age                int64
Food              object
Juice             object
Dessert           object
dtype: object
```

1.4.1 Tratamiento de valores nulos

[113]:
```
# Valores nulos
for feature in data_frame.columns:
    print('Total de valores nulos de', feature, '=', data_frame[feature].isna().
 ↪sum())
```

```
Total de valores nulos de Timestamp = 0
Total de valores nulos de Participant_ID = 0
Total de valores nulos de Gender = 4
Total de valores nulos de Nationality = 0
Total de valores nulos de Age = 0
Total de valores nulos de Food = 0
Total de valores nulos de Juice = 0
Total de valores nulos de Dessert = 0
```

1.4.2 Técnicas de tratamiento de nulos

1. Eliminar las observaciones (opción simple)
2. Imputación (mejor opción). Acá se rellenan los valores vacíos con algún valor, puede ser el promedio del valor de la columna. Puede ser el valor más repetido en el caso de una variable categórica.
3. Una extensión de la imputación. Se agrega una columna indicando que el valor ha sido reemplazado, de esta forma, se mantiene "identificados" a los valores nulos

[114]:
```
# Aplica la técnica 1 de tratamiento de nulos
data_frame = data_frame.dropna()
data_frame.shape
```

[114]: (284, 8)

1.5 Exploración de valores

1.5.1 Algunas estadísticas

[115]:
```
data_frame.describe()
```

[115]:
```
              Age
count  284.000000
mean    30.654930
std     11.244501
min      8.000000
25%     24.000000
50%     28.000000
75%     37.000000
max     80.000000
```

La función anterior obtiene las estadísticas de una sola columna (en este caso hay 8), aparece solo esa porque es la única columna numérica.

La interpretación:

1. Hay 284 valores en la columna
2. El promedio corresponde a 30.65
3. La desviación estándar es de 11.24, lo que quiere decir, la edad de las observaciones varía dentro del intervalo [19.41, 41.89] [mean - std; mean + std]
4. Valor mínimo de la columna es 8
5. El 25% de las observaciones es menor a 24
6. El 50% de las observaciones es menor a 28
7. El 75% de las observaciones es menor a 37
8. El valor máximo de la columna es 80

1.5.2 Una alternativa

Lo anterior igual se puede lograr de forma individual

[116]:
```
print('Mínimo:',data_frame['Age'].min())
print('Máximo:',data_frame['Age'].max())
print('Promedio:',data_frame['Age'].mean())
print('STD:',data_frame['Age'].std())
print(data_frame.Age.quantile([.25, .5, .75]))
print('*******')
print('Mínimo:',min(data_frame['Age']))
print('Máximo:',max(data_frame['Age']))
```

```
Mínimo: 8
Máximo: 80
Promedio: 30.654929577464788
STD: 11.244501302221419
```

```
0.25    24.0
0.50    28.0
0.75    37.0
Name: Age, dtype: float64
*******
Mínimo: 8
Máximo: 80
```

[117]:
```
data_frame.describe(include="all")
```

[117]:

	Timestamp	Participant_ID	Gender	Nationality	\
count	284	284	284	284	
unique	281	284	2	26	
top	2019/05/10 1:07:43 AM GMT+8	FPS209	Female	Indian	
freq	2	1	165	238	
mean	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	

	Age	Food	Juice	Dessert
count	284.000000	284	284	284
unique	NaN	2	2	3
top	NaN	Traditional food	Fresh Juice	Maybe
freq	NaN	234	252	122
mean	30.654930	NaN	NaN	NaN
std	11.244501	NaN	NaN	NaN
min	8.000000	NaN	NaN	NaN
25%	24.000000	NaN	NaN	NaN
50%	28.000000	NaN	NaN	NaN
75%	37.000000	NaN	NaN	NaN
max	80.000000	NaN	NaN	NaN

[118]:
```
# Considerando solo las columnas de tipo object
import numpy as np
data_frame.describe(include=[np.object])
```

[118]:

	Timestamp	Participant_ID	Gender	Nationality	\
count	284	284	284	284	
unique	281	284	2	26	
top	2019/05/10 1:07:43 AM GMT+8	FPS209	Female	Indian	
freq	2	1	165	238	

	Food	Juice	Dessert

```
count                284          284     284
unique                 2            2       3
top     Traditional food  Fresh Juice   Maybe
freq                 234          252     122
```

1.6 Agrupaciones de datos

[119]:
```
# Obtener total de pedidos considerando tipo de comida y género
data_frame.groupby('Gender')['Food'].value_counts()
```

[119]:
```
Gender  Food
Female  Traditional food    144
        Western Food         21
Male    Traditional food     90
        Western Food         29
Name: Food, dtype: int64
```

[120]:
```
# Obtener total de alternativas de postre por género
data_frame.groupby('Gender')['Dessert'].value_counts()
```

[120]:
```
Gender  Dessert
Female  Maybe      72
        Yes        58
        No         35
Male    Yes        52
        Maybe      50
        No         17
Name: Dessert, dtype: int64
```

[121]:
```
# Para comprobar los resultados anteriores
data_frame.groupby(data_frame.Gender).count()
```

[121]:
```
        Timestamp  Participant_ID  Nationality  Age  Food  Juice  Dessert
Gender
Female        165             165          165  165   165    165      165
Male          119             119          119  119   119    119      119
```

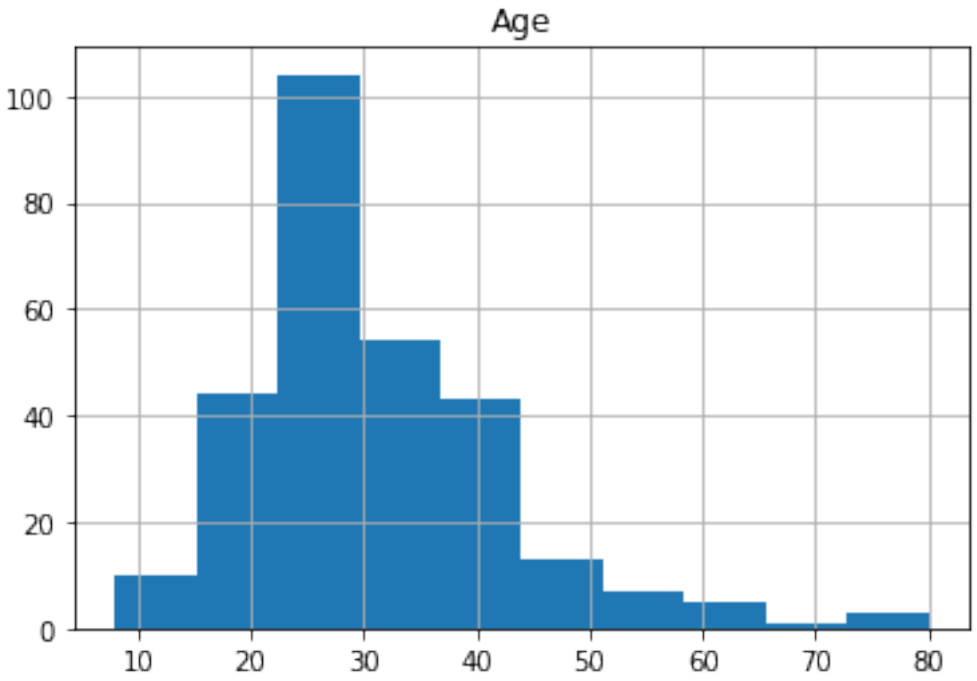
[122]:
```
# ¿Quiénes piden más postres, los hombres o las mujeres?
data_frame[data_frame.Dessert == 'Yes'].groupby('Gender')['Dessert'].count()
```

[122]:
```
Gender
Female    58
Male      52
Name: Dessert, dtype: int64
```

Claramente la pregunta anterior es más fácil de responder utilizando gráficos ...

1.7 También es posible graficar ...

[123]:
```
# Visualizamos de forma rápida las características de entrada, eliminando las␣
 ↪columnas que no interesan
data_frame.drop(['Timestamp', 'Participant_ID'],1).hist()
plt.show()
```

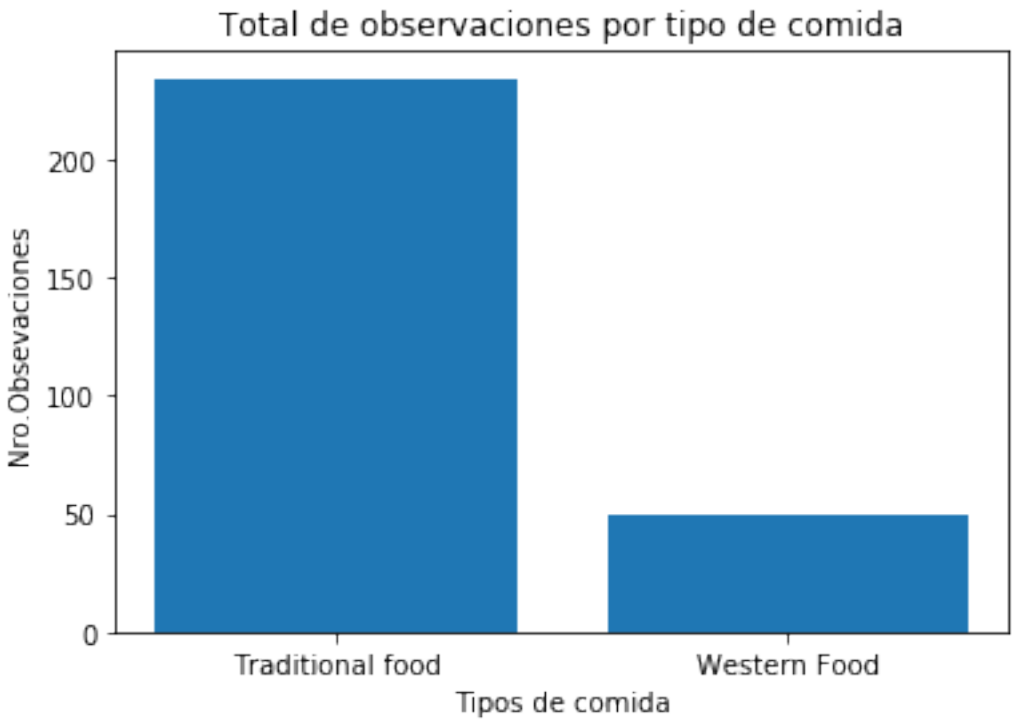


Nuevamente, solo considera las columnas que son numéricas. La pregunta válida acá es, ¿cómo se puede graficar por ejemplo, la columna Food?

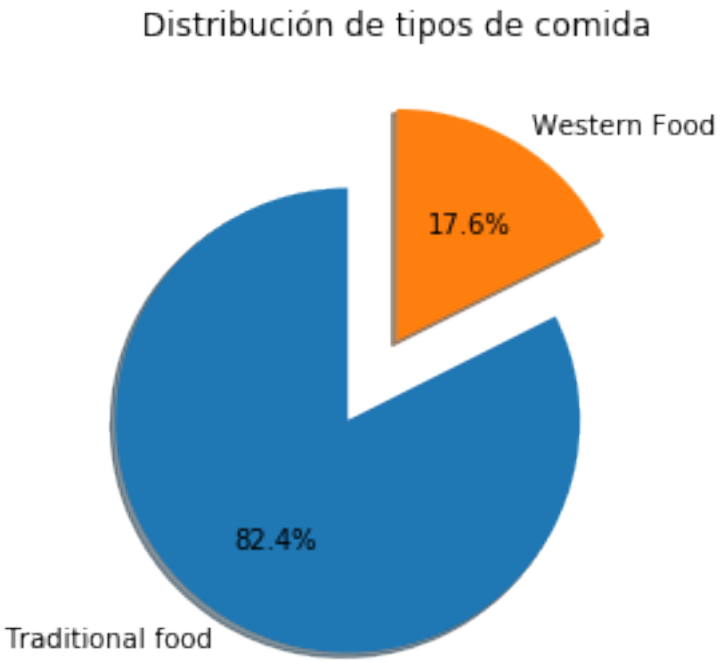
La respuesta es variada a la pregunta, sin embargo, una alternativa es la que se indica a continuación:

[124]:
```
def getGraficoBarras(label_x, label_y, title, feature):
    x_values = data_frame[feature].unique()
    y_values = data_frame[feature].value_counts().tolist()
    plt.bar(x_values, y_values)
    plt.title(title)
    plt.xlabel(label_x)
    plt.ylabel(label_y)
    plt.show()

getGraficoBarras('Tipos de comida', 'Nro.Obsevaciones', "Total de observaciones␣
 ↪por tipo de comida", 'Food')
```

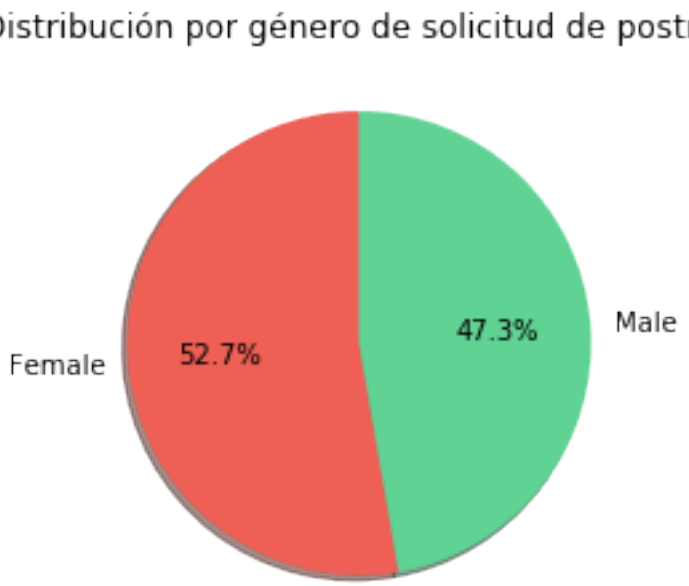


[125]:
```
tipos = data_frame['Food'].unique()
total = data_frame['Food'].value_counts().tolist()
explode = [0.4 if total[0] == max(total) else 0, 0.4 if total[1] == max(total)␣
 ↪else 0] # Destacar algunos
plt.pie(total, labels=tipos, explode = explode, autopct='%1.1f%%', shadow=True,␣
 ↪startangle=90)
plt.title('Distribución de tipos de comida')
plt.show()
```



Ahora que ya se conoce como se genera un gráfico de torta, entonces se puede, responder (gráficamente) a la pregunta acerca de ¿Quiénes piden más postres, los hombres o las mujeres?

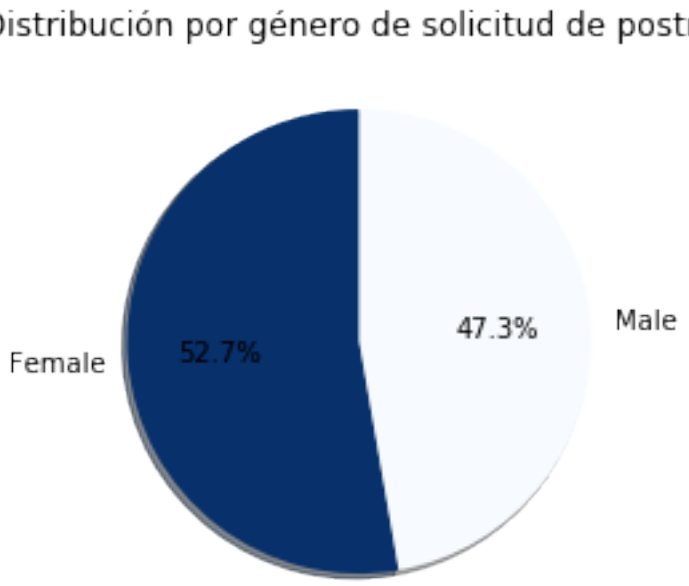
[126]:
```
# Respondiendo, gráficamente a: ¿Quiénes piden más postres, los hombres o las␣
↪mujeres?
df_pie = pd.DataFrame(data_frame[data_frame.Dessert == 'Yes'].
↪groupby('Gender')['Dessert'].count())
colores = ["#EE6055","#60D394"]
plt.pie(np.array(df_pie).ravel(), labels=[df_pie.index[0],df_pie.index[1]],
        autopct='%1.1f%%', shadow=True, startangle=90, colors = colores)
plt.title('Distribución por género de solicitud de postres')
plt.show()
```



1.7.1 Manejando la carta de colores

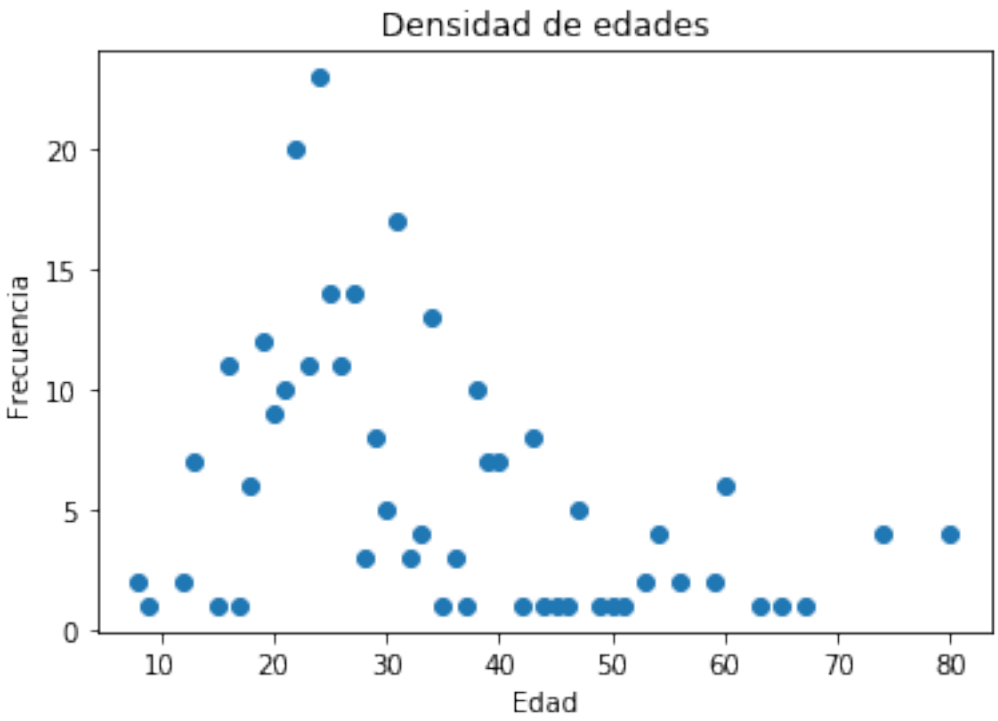
Más información en: https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

[127]:
```
from matplotlib import cm
from matplotlib import colors
df_pie = pd.DataFrame(data_frame[data_frame.Dessert == 'Yes'].
 ↪groupby('Gender')['Dessert'].count())
normdata = colors.Normalize(min(df_pie.Dessert), max(df_pie.Dessert))
colormap = cm.get_cmap("Blues")
colores =colormap(normdata(np.array(df_pie).ravel()))
plt.pie(np.array(df_pie).ravel(), labels=[df_pie.index[0],df_pie.index[1]],
        autopct='%1.1f%%', shadow=True, startangle=90, colors = colores)
plt.title('Distribución por género de solicitud de postres')
plt.show()
```



[129]:
```
# Gráfico de densidad de edades

x_values = data_frame.Age.unique()
y_values = data_frame.Age.value_counts().tolist()
plt.scatter(x_values, y_values, marker='o');
#plt.plot(x_values, y_values)
plt.title('Densidad de edades')
plt.xlabel('Edad')
plt.ylabel('Frecuencia')
plt.show()
```


Densidad de edades
Frecuencia
Edad
0
5
10
15
20
10
20
30
40
50
60
70
80