Portafolio de evidencias

Lección	Semana	Tema	Estado
1	27-01-2022	Instalación Anaconda y Jupyter	Lección Activa
2	03-02-2022	Metodología Data Science	Lección Activa
3	10-02-2022	Análisis Estadística Descriptiva	No hubo Lección
4	17-02-2022	Carga de Set de Datos para Análisis	Lección Activa
5	24-02-2022	Procesamiento de Datos y Gestión de Valores perdidos	Lección Activa
6	03-03-2022	Estandarización y normalización de los datos	Lección Activa: Inicia Portafolio
7	10-03-2022	Normalización de Datos	Lección Activa
8	17-03-2022	Librería Numpy	Lección Activa
9	24-03-2022	Análisis Exploratorio de Datos (Sesión Asincrona)	No hubo Lección
10	31-03-2022	Análisis Exploratorio de Datos	Lección Activa
11	07-04-2022	Aclarar dudas del trabajo final	Lección Activa
12	14-04-2022	Semana Santa	No hubo clase
13	21-04-2022	Anova	Lección Activa
14	28-04-2022	Entrega Portafolio	Lección Activa: Fin Portafolio
15	05-05-2022		Lección Activa: Fin del Curso

➢ Semana 03-03-2022

Síntesis

Procesamiento de los datos, se refiere a los pasos aplicados para la minería de datos y se dividen en dos categorías:

1- Seleccionar objetos de datos y atributos para el análisis
2- Crear/ Combinar los atributos

Pasos para la implementación de métodos de Procesamiento de Datos en Python:

1- Importación de la Biblioteca
2- Importación del conjunto de datos
3- Manejo de datos faltantes
4- Manejo de datos categóricos
5- Dividir el conjunto de datos en conjuntos
6- Escalado de características

El binning de datos, que es una técnica para el procesamiento de datos. Se pueden seguir dos enfoques.

1 – Convierte los datos numéricos en datos categóricos

2 – Realiza el muestreo de datos, reduciendo el número de muestras

Es muy útil cuando se necesita desratización.

Desarrollo de Actividades

Importar Bibliotecas se importan bibliotecas como Pandas, Numpy vistas en clases, adicionalmente se investiga sobre Marplotlib para el manejo de gráficos, scipy para herramientas y algoritmos matemáticos y seaborn para visualización de datos estadísticos.

El binning de datos es un tipo de procesamiento de datos, un mecanismo que incluye también el tratamiento de los valores faltantes, el formato, la normalización y la estandarización. Se puede utilizar para:

Convertir numérico a categórico incluye binning por distancia y por frecuencia

Reducir los valores numéricos incluye la cuantificación (o muestreo)

El binning es una técnica para el suavizado de dato:

```
min_value = df['Cupcake'].min()
max_value = df['Cupcake'].max()
print(min_value)
print(max_value)
```

Lo que da como resultado:

```
4
100
```

Por lo cual se utiliza para calcular el rango de cada intervalo, es decir el valor mínimo y máximo de cada intervalo.

Agrupamiento por frecuencia calcula el tamaño de cada bin de modo que cada bin contiene el mismo número de observaciones, para esto se utiliza la función de Python

pandasqcut()precision

Muestreo es otra técnica de binning de datos, permite reducir el número de muestras, agrupando valores similares o valores contiguos.

Lecciones Aprendidas

Se continua con la clase de programación en Jupyter, justamente esta lección aprendimos como los datos que el código detecta como una variable ya sea numero entero, decimales, texto; se deben verificar para realizar el cambio si es necesario, por ejemplo, un dato que sea un numero con decimales debería ser tomado como un float, si este se muestra como un object el mismo se debe cambiar a float que significa numero con decimales.

Por ejemplo, para revidar la información de los datos se utiliza el nombre de la variable que se definió más .info ()

dfAutos.info ()

De igual forma en la programación del código se visualiza como desde una base de datos en Jupyter se logra visualizar las medidas estadísticas de una base como son la media, mínimo, máximo, cuartiles, entre otras, para ello se aplica la fórmula:

dfAutos1.describe ()

Se aprende a **unir** archivos de bases separadas, pero homologas en campos para dejar una sola y lograr trabajar con todos los datos unionArchivos= [dfAutos1], [dfAutos2]

En bases muy grandes es difícil ver todas las filas de la base para revisar la base por lo que se utiliza la función replace que ayuda a remplazar los datos y pasarlos a Nan, con el objetivo de contar los valores Nan, con el objetivo final de imputar valores o eliminar filas o columnas.

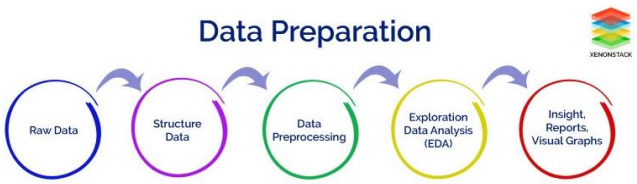
Eso lo realiza con un ciclo **for** column.

Imputar valores numéricos con la media
Se debe cambiar a un dato tipo float, para calcular la media de los datos de una columna. Para remplazar los valores Nan por el cálculo de la media en el set de datos.

Imputar valores String
Se busca el valor más común de una columna, con el fin de sustituir los valores Nan, por este valor común.

Eliminar Filas
Si una columna tiene más del 90% de valores perdidos se elimina o bien una fila que la mayoría de sus datos son valores perdidos. Para ello se utiliza la función dropna. Específicamente los valores Nan y se resetea el orden del set de datos para que el conteo quede consecutivo.

Evidencias



Ejercicio en clases

1.3 Revision de Encabezados

```
In [9]: dfAutos1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 26 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   symboling    99 non-null     int64
 1   normalized   99 non-null     object
 2   make         99 non-null     object
 3   fuel         99 non-null     object
 4   aspiration   99 non-null     object
 5   doors        99 non-null     object
```

```
In [12]: dfAutos1.describe()
```

Out[12]:

	symboling	base	length	width	heingt	curb	engine-size	compression	horsepower	peak	City	Higway
count	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000
mean	0.979798	98.016162	171.728283	65.760606	53.063636	2485.606061	127.141414	9.711212	101.939394	5214.141414	25.626263	31.121212
std	1.087778	6.375715	13.895558	2.522665	2.350183	607.860615	52.132095	3.438604	38.868911	467.190933	7.537100	7.907982
min	-1.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	48.000000	4200.000000	13.000000	16.000000
25%	0.000000	94.400000	160.750000	63.850000	50.800000	1980.000000	92.000000	8.500000	69.000000	5000.000000	19.000000	25.000000
50%	1.000000	96.300000	172.400000	65.200000	53.300000	2380.000000	110.000000	9.000000	88.000000	5200.000000	25.000000	31.000000
75%	1.500000	100.500000	177.800000	66.500000	54.500000	2817.000000	136.000000	9.400000	121.000000	5500.000000	31.000000	37.000000
max	3.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.000000	22.700000	262.000000	6000.000000	49.000000	54.000000

```
In [13]: dfAutos2.describe()
```

Out[13]:

	symboling	base	length	width	heingt	curb	engine-size	bore	stroke	compression	City	Higway
count	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000	98.000000
mean	0.846939	98.807143	175.189796	65.865306	54.200000	2584.489796	125.377551	3.388367	3.172449	10.549082	25.204082	30.846939
std	1.310953	5.307148	10.013717	1.648561	2.404806	411.035736	29.413341	0.263113	0.352180	4.356249	5.445057	5.764005

1.3 Union de Archivos

In [14]:
```
unionarchivos = [dfAutos1,dfAutos2] #Crea una lista de listas
dfSetAutos = pd.concat(unionarchivos)
dfSetAutos.shape
```

Out[14]: (197, 26)

2.1 Identificacion de valores faltantes o perdidos y estandarizacion

dfSetAutos

In [15]:
```
dfSetAutos.replace("?",np.nan,inplace = True)
```

In [16]:
```
dfSetAutos
```

Out[16]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepowe
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	11
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	11
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	15
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	10
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	11

2.2 Contar los NaN en el set de datos

In [17]:
```
valoresPerdidos = dfSetAutos.isnull()
valoresPerdidos.head(5)
```

Out[17]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	peak	F
0	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	F
1	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	F
2	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	F

5 rows × 26 columns

2.3 Imputar valores numericos con la media

In [19]:
```
avg_norm = dfSetAutos["normalized"].astype("float").mean(axis=0)
print ("media del normalized: ",avg_norm)
```

media del normalized: 123.6025641025641

In [20]:
```
dfSetAutos["normalized"].replace(np.nan,124, inplace = True)
dfSetAutos.head(5)
```

Out[20]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	peak
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500

2.4 Imputacion de Valores STRING

In [21]:
```
dfSetAutos["doors"].value_counts().idxmax()
```

Out[21]: 'four'

In [22]:
```
dfSetAutos["doors"].replace(np.nan, "four", inplace = True)
dfSetAutos.head (5)
```

Out[22]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	peak
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500

5 rows × 26 columns

2.5 Eliminar filas con valores o registros perdidos

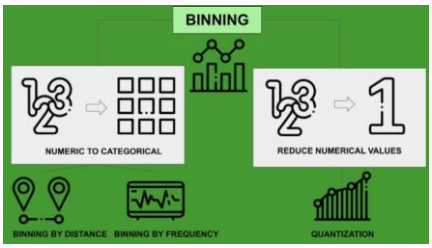
```
In [23]: dfSetAutos.dropna(subset = ['price'],axis = 0,inplace = True)
         dfSetAutos.reset_index(drop=True,inplace=True)

In [24]: dfSetAutos.head(3)
```

Out[24]:

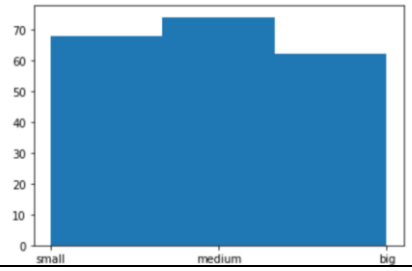
	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	peak
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000

3 rows × 26 columns

```
In [25]: valoresPerdidos = dfSetAutos.isnull()
         valoresPerdidos.head(5)

         for column in valoresPerdidos.columns.values.tolist():
```



Histograma



➢ Semana 10-03-2022

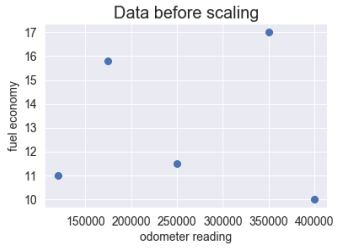
Síntesis
Normalización de datos es una práctica común en el aprendizaje automático que consiste en transformar columnas numéricas a una escala común. Las características con valores más altos dominaran el proceso de inclinación La normalización de datos transforma los datos multisecular a la misma escala. Después de la normalización, todas las variables tienen una influencia similar en el modelo, mejorando la estabilidad y el rendimiento del algoritmo de aprendizaje. Existen múltiples técnicas de normalización: 1- El escalado absoluto máximo 2- La escala de características min-máx.

3- El método z-score
4- El escalado robusto

Desarrollo de Actividades

El siguiente marco de datos contiene las entradas (variables independientes) de un modelo de regresión múltiple para predecir el precio de un automóvil de segunda mano: (1) la lectura del odómetro (km) y (2) la economía de combustible (km / l).

	odometer_reading	fuel_economy
0	120000	11.0
1	250000	11.5
2	175000	15.8
3	350000	17.0
4	400000	10.0



Como puede observar, la lectura del odómetro varía de 120000 a 400000, mientras que la economía de combustible varía de 10 a 17. El modelo **de regresión lineal múltiple** ponderará la variable de lectura del odómetro más que el atributo de economía de combustible debido a sus valores más altos. Sin embargo, no significa que el atributo de lectura del odómetro sea más importante como predictor. Para resolver este problema, tenemos que **normalizar** los valores de ambas variables. ♥

El escalado absoluto máximo

La **escala absoluta máxima** reescala cada característica **entre -1 y 1** dividiendo cada observación por su valor absoluto máximo.

$$x_{scaled} = \frac{x}{max(|x|)}$$

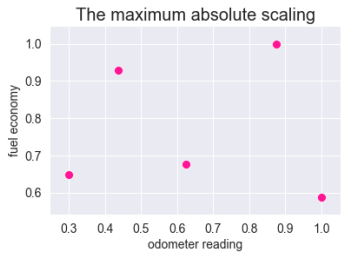
Podemos aplicar el **escalado absoluto máximo** en **Pandas** utilizando los métodos **.max()** y **.abs(),** como se muestra a continuación.

	odometer_reading	fuel_economy
0	0.3000	0.647059
1	0.6250	0.676471
2	0.4375	0.929412
3	0.8750	1.000000
4	1.0000	0.588235

Alternativamente, podemos usar la biblioteca **Scikit-learn** para calcular la **escala absoluta máxima**. Primero, creamos una abs_scaler con la clase **MaxAbsScaler**. Luego, usamos el **método de ajuste** para aprender los parámetros requeridos para escalar los datos (el **valor absoluto máximo** de cada característica). Finalmente, transformamos los datos utilizando esos parámetros.

	odometer_reading	fuel_economy
0	0.3000	0.647059
1	0.6250	0.676471
2	0.4375	0.929412
3	0.8750	1.000000
4	1.0000	0.588235

Como puedes observar, obtenemos los mismos resultados utilizando **Pandas** y **Scikit-learn**. El siguiente gráfico muestra los datos transformados después de realizar el escalado absoluto máximo.



La escala de características min-max

El **enfoque min-max** (a menudo llamado **normalización**) reescala la función a un **rango fijo** de **[0,1]** restando el valor mínimo de la función y luego dividiendo por el rango.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Podemos aplicar la **escala min-max** en **Pandas** usando los métodos. min **()** y. max **()**.

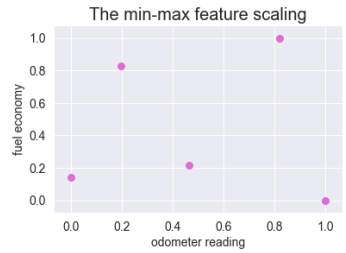
	odometer_reading	fuel_economy
0	0.000000	0.142857
1	0.464286	0.214286
2	0.196429	0.828571
3	0.821429	1.000000
4	1.000000	0.000000

Alternativamente, podemos usar la clase **MinMaxScaler** disponible en la biblioteca **Scikit-learn**. Primero, creamos un **objeto escalador**. Luego, **ajustamos** los parámetros del escalador, lo que significa que calculamos el valor mínimo y máximo para cada característica. Finalmente, **transformamos** los datos utilizando esos parámetros.

	odometer_reading	fuel_economy
0	0.000000	0.142857
1	0.464286	0.214286
2	0.196429	0.828571
3	0.821429	1.000000
4	1.000000	0.000000

Adicionalmente, podemos obtener los valores mínimos y máximos calculados por la función **fit** para normalizar los datos con los atributos **data_min_** y **data_max_**.

El siguiente gráfico muestra los datos después de aplicar la escala de características min-max. Como puede observar, esta técnica de normalización reescala todos los valores de características para que estén dentro del rango de [0, 1].



Como puedes observar, obtenemos los mismos resultados utilizando **Pandas** y **Scikit-learn**. Sin embargo, si desea realizar muchos pasos de transformación de datos, se recomienda utilizar **MinMaxScaler** como entrada en un constructor **de pipeline** en lugar de realizar la normalización con **Pandas**.

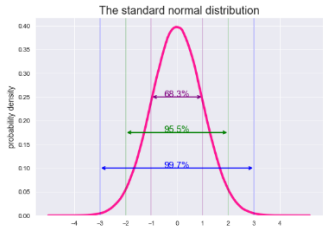
Además, es importante tener en cuenta que la **escala absoluta máxima** y la escala **mín-máx** son muy **sensibles a los valores atípicos** porque un solo valor atípico puede influir en los valores mínimos y máximos y tener un gran efecto en los resultados.

El método z-score

El **método z-score** (a menudo llamado **estandarización**) transforma los datos en una distribución con una **media de 0** y una **desviación estándar de 1**. Cada valor estandarizado se calcula restando la **media** de la característica correspondiente y luego dividiendo por la **desviación estándar**.

$$x_{std} = \frac{x-\mu}{\sigma}$$

A diferencia **de la escala min-max**, la **puntuación z** no reescala la función a un rango fijo. La **puntuación z** suele oscilar entre **-3,00 y 3,00** (más del 99% de los datos) si la entrada se distribuye normalmente. Sin embargo, los valores estandarizados también pueden ser más altos o más bajos, como se muestra en la imagen a continuación.



Es importante tener en cuenta que **las puntuaciones z** no se distribuyen necesariamente normalmente. Simplemente escalan los datos y siguen la misma distribución que la entrada original. Esta distribución transformada tiene una **media de 0** y una **desviación estándar de 1** y va a ser la **distribución normal estándar** (ver la imagen de arriba) sólo si la función de entrada sigue una distribución normal.

Podemos calcular la **puntuación z** en **Pandas** usando los métodos **.mean()** y **std()**.

	odometer_reading	fuel_economy
0	-1.189512	-0.659120
1	-0.077019	-0.499139
2	-0.718842	0.876693
3	0.778745	1.260647
4	1.206628	-0.979081

Alternativamente, podemos usar la clase **StandardScaler** disponible en la biblioteca **Scikit-learn** para realizar la puntuación z. Primero, creamos un objeto **standard_scaler**. Luego, calculamos los parámetros de la transformación (en este caso la **media** y la **desviación estándar**) utilizando el método. fit **()**. A continuación, llamamos al método. transform**()** para aplicar la estandarización al marco de datos. El método. transform **()** utiliza los parámetros generados a partir del método. fit **()** para realizar la puntuación z.

	odometer_reading	fuel_economy
0	-1.329915	-0.736918
1	-0.086110	-0.558055
2	-0.803690	0.980173
3	0.870664	1.409446
4	1.349051	-1.094646

Para simplificar el código, hemos utilizado el **método .fit_transform()** que combina ambos métodos (fit y transform) juntos. Como se puede observar, los resultados difieren de los obtenidos utilizando **Pandas**. La función **StandardScaler** calcula la **desviación estándar de la población** donde la suma de cuadrados se divide por **N** (número de valores en la población).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(y_i-\mu)^2}{N}}$$

Por el contrario, el **método .std()** calcula la **desviación estándar de la muestra** donde el denominador de la fórmula es **N-1** en lugar de **N**.

$$s = \sqrt{\frac{\sum_{i=1}^{N}(y_i-\bar{y})^2}{N-1}}$$

Para obtener los mismos resultados con Pandas, establecemos el parámetro **ddof** igual a 0 (el valor predeterminado es ddof=1) que representa el divisor utilizado en los cálculos (**N-ddof**).

```
odometer_reading    104517.941044
fuel_economy             2.795425
dtype: float64
```

Lecciones Aprendidas

La normalización de datos consiste en transformar columnas numéricas a una escala común.

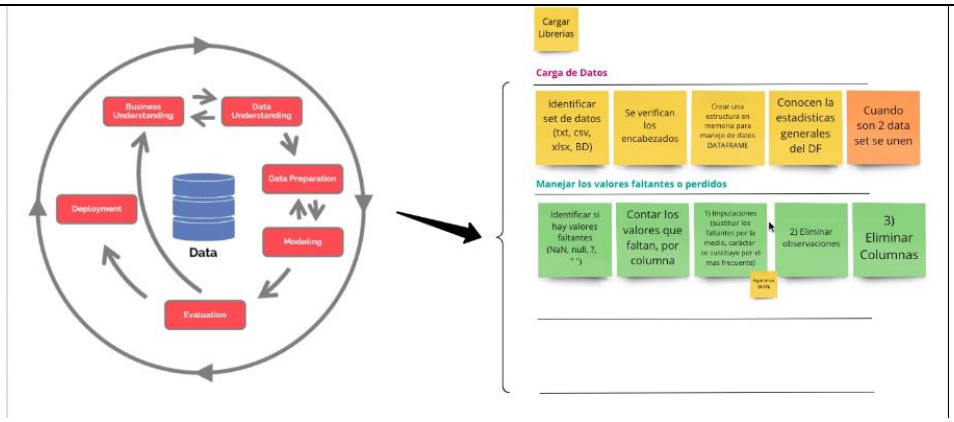
Con Pyhton se puede implementar la normalización de datos de una manera sencilla.

La biblioteca de Pandas contiene múltiples métodos integrados para calcular las funciones estadísticas descriptivas más comunes que hacen que las técnicas de normalización de datos sean realmente fáciles de implementar.

También se puede utilizar la librería Scikit-Learn para transformar los datos en una escala común. En esta biblioteca, los métodos de escalado más frecuentes ya están implementados.

En la lección de esta semana se realizó un repaso de los temas vistos en la clase anterior, se visualizó todos los pasos por detalle que se realizan para lograr cargar datos, imputar datos, contar valores perdidos y como se visualiza en la imagen en evidencia, en la cual se visualiza los pasos.

Evidencias



1.1 Importancion de Libreria

```
In [1]: #Se realiza la importancion de las librerias necesarias para el EDA
        import pandas as pd                 #para el manejo del Data Frame
        import numpy as np                  #para el manejo de estructuras de datos como las listas
        import matplotlib.pyplot as plt     #para el manejo de los graficos
```

➢ Semana 17-03-2022

Síntesis
Librería Numpy es una librería en la que se define un de dato que representa matrices multidimensionales, equivalentes a las matrices del R. Además, incluye algunas funcionalidades básicas para trabajar con ellas. Numpy es una librería con una amplia tradición en Python, es estable y muy rápida.

Desarrollo de Actividades

Numpy es el paquete fundamental para la computación de Pyhton. Es una biblioteca que proporciona un objeto de matriz multidimensional, varios objetos derivados y una variedad de rutinas para operaciones rápidas en matrices que incluye matemáticas, lógicas, manipulación de formas, clasificación, selección, E/s transformadas de Fourier discretas, algebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

Para Instalar Numpy

Si se tiene Python, se puede instalar de la siguiente forma:

- Conda install numpy o Pip install numpy

Como importar Numpy:

Import numpy as np

¿Por qué utilizar Numpy?

Las matrices de Numpy son más rápidas y compactas que las listas de Pyhton. Esto permite que el código se optimice aún más.

¿Qué es una matriz?

Es una estructura de datos central de la biblioteca Numpy. Es una cuadricula de valores y contiene información sobre datos sin procesar, como localizar un elemento y como interpretar un elemento.

```
a = np. array([1, 2, 3, 4, 5, 6])
```

Para Crear una matriz básica

Para crear una matriz Numpy, se puede utilizar la función np.array ()

```
import numpy as np
>>> a = np.array([1, 2, 3])
```

Lecciones Aprendidas

Estandarización de datos

Facilita que los datos de origen sean internamente coherentes; es decir, que cada tipo de datos tenga el mismo tipo de contenido y de formato.

Se debe revisar el tipo de datos de la columna esto debido a que si se tiene que trabajar con una columna que está en object y esta se debería ser un numero para generar un cálculo matemático, se debe convertir a numero entero y así evitar un error, esto se hace mediante el paso de estandarización.

En resumen, es convertir datos a las medidas que más nos convenga y hacerlos comparables, lo recomendable no es modificar datos del set original, sino más crear nuevas medidas con los datos que se desean comprar o calcular.

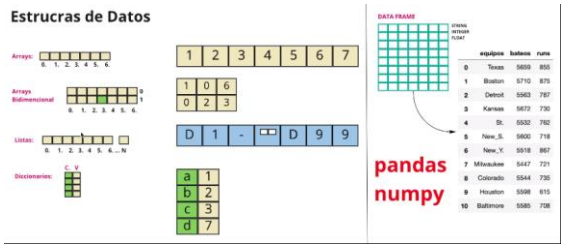
Normalización de Valores

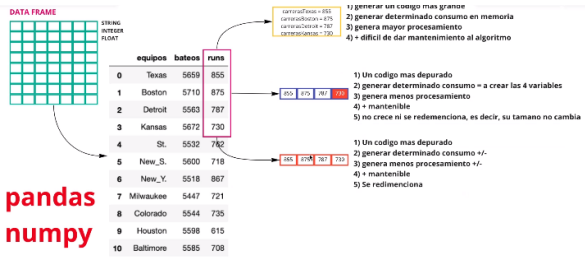
La normalización de valores es hacer que las medidas sean comprables entre sí. Si se tiene 3 datos diferentes entre sí, se crea una medida que logre relacionar y comparar los datos, por ejemplo, realizar el rango entre 0 y 1 de una columna, se puede dividir el dato máximo contra el dato de estudio y así con las otras columnas para lograr compararlas.

Rangos

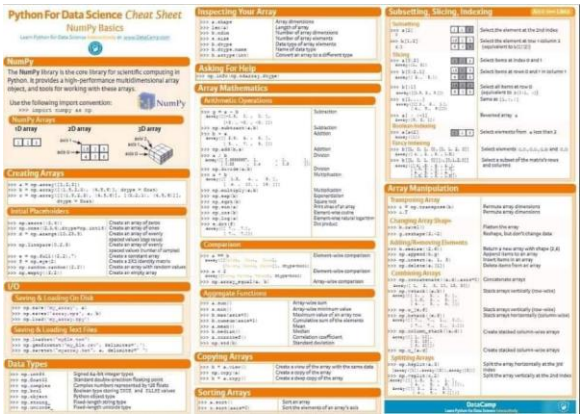
Se requiere datos numéricos, es una forma de agrupar datos para un análisis, para ellos se utilizan categorías, como se ha visualizado de igual forma se deben limpiar datos en la columna que se desea agrupar.

Evidencias





	equipos	bateos	runs
0	Texas	5659	855
1	Boston	5710	875
2	Detroit	5563	787
3	Kansas	5672	730
4	St.	5532	762
5	New_S.	5600	718
6	New_Y.	5518	867
7	Milwaukee	5447	721
8	Colorado	5544	735
9	Houston	5598	615
10	Baltimore	5585	708



Ejercicio en clases

3. Estanderizacion de Datos

1. Verificar el dominio de las variables, se ajusta ese dominio si es necesario
2. Convertir los valores en funcion de su interpretacion
3. Normalizar los datos para que los valores queden entre 0 y 1
4. Crear rangos (Binning)

3.1 Verificar dominio de datos

```
In [26]: dfSetAutos.head(2)
```

Out[26]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	peak
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	500
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	500

2 rows × 26 columns

```
In [27]: dfSetAutos.dtypes
```

```
Out[27]: symboling          int64
         normalized        object
         make              object
         fuel              object
         aspiration        object
```

3.2 Estandarizacion de Valores

In [29]:
```
dfSetAutos.head(5)
```

Out[29]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	engine-size	system	bore	stroke	compression	horsepower	pe
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500

5 rows × 26 columns

In [30]:
```
dfSetAutos["city-L/100km"] = 235.215/dfSetAutos["City"]
dfSetAutos.head(5)
```

Out[30]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	system	bore	stroke	compression	horsepower	peak	City
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	mpfi	2.68	3.47	9.0	154	5000.0	19
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	10.0	102	5500.0	24
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	mpfi	3.19	3.40	8.0	115	5500.0	18

5 rows × 27 columns

In [31]:
```
dfSetAutos["Higway-L/100km"] = 235.215/dfSetAutos["Higway"]
dfSetAutos.head(5)
```

Out[31]:

	symboling	normalized	make	fuel	aspiration	doors	style	wheels	engine	base	...	bore	stroke	compression	horsepower	peak	City	Higway
0	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27
1	3	124	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27
2	1	124	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	2.68	3.47	9.0	154	5000.0	19	26
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	3.19	3.40	10.0	102	5500.0	24	30
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	3.19	3.40	8.0	115	5500.0	18	22

5 rows × 28 columns

3.3 Normalizacion de valores

In [32]:
```
dfTemp = dfSetAutos[["length","width","heingt"]]
dfTemp.head()
```

Out[32]:

	length	width	heingt
0	168.8	64.1	48.8
1	168.8	64.1	48.8
2	171.2	65.5	52.4
3	176.6	66.2	54.3
4	176.6	66.4	54.3

In [33]:
```
dfSetAutos["length-N"] = dfSetAutos["length"]/dfSetAutos["length"].max()
dfSetAutos["width-N"] = dfSetAutos["width"]/dfSetAutos["width"].max()
dfSetAutos["heingt-N"] = dfSetAutos["heingt"]/dfSetAutos["heingt"].max()
dfSetAutos.head(5)
```

In [33]:
```
dfSetAutos["length-N"] = dfSetAutos["length"]/dfSetAutos["length"].max()
dfSetAutos["width-N"] = dfSetAutos["width"]/dfSetAutos["width"].max()
dfSetAutos["heingt-N"] = dfSetAutos["heingt"]/dfSetAutos["heingt"].max()
dfSetAutos.head(5)
```

Out[33]:

ke	fuel	aspiration	doors	style	wheels	engine	base	...	horsepower	peak	City	Higway	price	city-L/100km	Higway-L/100km	length-N	width-N	heingt-N
a-ro	gas	std	two	convertible	rwd	front	88.6	...	111	5000.0	21	27	13495.0	11.200714	8.711667	0.811148	0.890278	0.816054
a-ro	gas	std	two	convertible	rwd	front	88.6	...	111	5000.0	21	27	16500.0	11.200714	8.711667	0.811148	0.890278	0.816054
a-ro	gas	std	two	hatchback	rwd	front	94.5	...	154	5000.0	19	26	16500.0	12.379737	9.046731	0.822681	0.909722	0.876254
udi	gas	std	four	sedan	fwd	front	99.8	...	102	5500.0	24	30	13950.0	9.800625	7.840500	0.848630	0.919444	0.908027
udi	gas	std	four	sedan	4wd	front	99.4	...	115	5500.0	18	22	17450.0	13.067500	10.691591	0.848630	0.922222	0.908027

3.4 Rangos

```
In [39]: bins = np.linspace(min(dfSetAutos["horsepower"]),max(dfSetAutos["horsepower"]),4)
         bins

Out[39]: array([ 48.        , 119.33333333, 190.66666667, 262.        ])

In [40]: group_names = ["bajo","medio","alto"]
         dfSetAutos["horsepower-rangos"] = pd.cut(dfSetAutos["horsepower"], bins, labels=group_names,include_lowest=True)
         dfSetAutos[['horsepower','horsepower-rangos']].head(5)
```

Out[40]:

	horsepower	horsepower-rangos
0	111	bajo
1	111	bajo
2	154	medio
3	102	bajo
4	115	bajo

```
In [41]: dfSetAutos ["horsepower-rangos"].value_counts()

Out[41]: bajo     147
         medio     39
         alto       5
         Name: horsepower-rangos, dtype: int64
```

➢ Semana 31-03-2022

Síntesis

La librería matplotlib. pyplot nos ofrece multitud de opciones a la hora de trabajar con gráficas, desde pintar varias funciones en la misma figura, con diferentes colores o diseños, hasta indicar por ejemplo el máximo de una función.

Matplotlib es una biblioteca de gráficos basada en Python con soporte completo para 2D y soporte limitado para gráficos 3D

Desarrollo de Actividades

La biblioteca se dirige a una amplia gama de casos de uso. Puede incrustar gráficos en el kit de herramientas de interfaz de usuario de su elección, y actualmente admite gráficos interactivos en todos los principales sistemas operativos de escritorio utilizando los kits de herramientas GTK +, Qt, Tk, FLTK, wxWidgets y Cocoa.

Por lo tanto, matplotlib se desarrolló originalmente como una herramienta de visualización EEG / ECoG para esta aplicación GTK +, y este caso de uso dirigió su arquitectura original. matplotlib fue diseñado originalmente para servir a un segundo propósito también: como un reemplazo para la generación de gráficos interactivos impulsados por comandos, algo que MATLAB hace muy bien.

Capa de backend

En la parte inferior de la pila se encuentra la capa de backend, que proporciona implementaciones concretas de las clases de interfaz abstractas:

FigureCanvas encapsula el concepto de una superficie para dibujar (por ejemplo, "el papel").

Renderer hace el dibujo (por ejemplo, "el pincel").

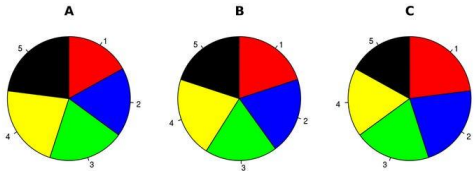
Event controla las entradas del usuario, como los eventos de teclado y ratón.

matplotlib pasa mucho tiempo transformando coordenadas de un sistema a otro. Estos sistemas de coordenadas incluyen:

- datos: los valores originales de los datos sin procesar
- ejes: el espacio definido por un rectángulo de ejes particular
- figura: el espacio que contiene toda la figura
- display: las coordenadas físicas utilizadas en la salida (por ejemplo, puntos en PostScript, píxeles en PNG)

Grafico Circular

Digamos que representan la encuesta de una elección local con cinco candidatos en tres puntos diferentes A, B, un C durante una elección:



No es tan claro a la hora de interpretar y comparar la información entre candidatos.

Los gráficos más elegantes no requieren etiquetados de datos

No debería necesitar tener ningún número extraño en sus datos para hacer su punto.

Entonces, revisemos:

- Siempre que haya similitud en la información disponible, un gráfico circular no es el gráfico correcto para usar.
- Siempre que haya varios (3 o más) puntos de datos diferentes, un gráfico circular no es el gráfico correcto para usar.
- Los gráficos circulares son muy fáciles de abusar.
- Un gráfico circular no es el gráfico correcto para usar si necesita etiquetar cada porcentaje.

EDA

- En estadística, el análisis exploratorio de datos es un enfoque para analizar conjuntos de datos para resumir sus características principales, a menudo con métodos visuales. Se puede usar un modelo estadístico o no, pero principalmente EDA es para ver lo que los datos pueden decirnos más allá del modelado formal o la tarea de prueba de hipótesis.

EDA en Python utiliza la visualización de datos para dibujar patrones y conocimientos significativos. También implica la preparación de conjuntos de datos para el análisis mediante la eliminación de irregularidades en los datos.

En base a los resultados de EDA, las empresas también toman decisiones comerciales, que pueden tener repercusiones más adelante.

- Si EDA no se realiza correctamente, puede obstaculizar los pasos adicionales en el proceso de creación del modelo de aprendizaje automático.
- Si se hace bien, puede mejorar la eficacia de todo lo que hacemos a continuación.

En este artículo veremos sobre los siguientes temas:

- Abastecimiento de datos
- Limpieza de datos
- Análisis univariado
- Análisis bivariado
- Análisis multivariante

1. Abastecimiento de datos

El abastecimiento de datos es el proceso de encontrar y cargar los datos en nuestro sistema. A grandes rasgos hay dos formas en las que podemos encontrar datos.

- Datos privados
- Datos públicos

Datos privados Como su nombre indica, los datos privados son proporcionados por organizaciones privadas. Hay algunas preocupaciones de seguridad y privacidad asociadas a él. Este tipo de datos se utilizan principalmente para el análisis interno de las organizaciones.

Datos públicos Este tipo de datos está disponible para todos. Podemos encontrar esto en sitios web gubernamentales y organizaciones públicas, etc. Cualquier persona puede acceder a estos datos, no necesitamos ningún permiso especial o aprobación.

Limpieza de Datos

Los siguientes son los pasos a seguir al fijar filas y columnas:

- Elimine las filas y columnas de resumen del conjunto de datos.
- Eliminé las filas de encabezado y pie de página en cada página.
- Elimine filas adicionales como filas en blanco, números de página, etc.
- Podemos fusionar diferentes columnas si esto permite una mejor comprensión de los datos.
- Del mismo modo, también podemos dividir una columna en varias columnas en función de nuestros requisitos o comprensión.
- Agregue nombres de columna, es muy importante tener nombres de columna en el conjunto de datos.

Si faltan valores se pueden eliminar los registros faltantes o imputando valores.

También se pueden rellenar los valores que faltan como ‘NaN’ para que, al hacer cualquier análisis estadístico, no afecte el resultado.

Hay dos tipos de valores atípicos:

1. Valores atípicos univariados: Los valores atípicos univariados son los puntos de datos cuyos valores se encuentran más allá del rango de valores esperados basados en una variable.
2. Valores atípicos multivariantes: Al trazar datos, es posible que algunos valores de una variable no se encuentren más allá del rango esperado, pero cuando se trazan los datos con alguna otra variable, estos valores pueden estar lejos del valor esperado.

Estandarización de Valores: tenemos que asegurar que el valor de la misma columna debe estar en la misma escala.

Análisis univariado: Analizar los sobre una variable/ columna de un conjunto de datos.

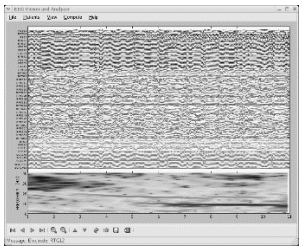
Lecciones Aprendidas

Así es como haremos el Análisis Exploratorio de Datos. El análisis exploratorio de datos (EDA) nos ayuda a mirar más allá de los datos. Cuanto más exploramos los datos, más información obtenemos de ellos. Como analista de datos, casi el 80% de nuestro tiempo se dedicará a comprender los datos y resolver diversos problemas comerciales a través de EDA.+}

Se utiliza una semilla en la biblioteca numpy para utilizar un gráfico establecido.

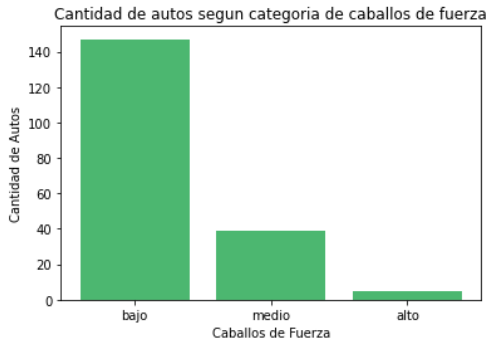
Se trabaja en la clase en la construcción del gráfico de barras como muestra la evidencia

Evidencias



Ejercicio en clases

```
In [45]: eje_x = ('bajo', 'medio', 'alto')
         eje_y = gfl['symboling'].tolist()

         ## Crear la grafica

         plt.bar(eje_x, eje_y, color='#009933', alpha = 0.7)

         #Leyenda en el eje y
         plt.ylabel('Cantidad de Autos')

         #Leyenda en el eje x
         plt.xlabel('Caballos de Fuerza')

         #Titulo al Grafico
         plt.title('Cantidad de autos segun categoria de caballos de fuerza')

         #Cambia el tamaño del grafico
         plt.figure(figsize=(10,30))

         plt.show()
```



➢ Semana 07-04-2022

Síntesis
Python es un gran lenguaje para hacer análisis de datos, principalmente debido al fantástico ecosistema de paquetes python centrados en datos. Pandas es uno de esos paquetes y hace que la importación y el análisis de datos sea mucho más fácil. Pandas groupby se utiliza para agrupar los datos según las categorías y aplicar una función a las categorías. También ayuda a agregar datos de manera eficiente. La función Pandas se utiliza para dividir los datos en grupos en función de algunos criterios. Los objetos de los pandas se pueden dividir en cualquiera de sus ejes. La definición abstracta de agrupación es proporcionar una asignación de etiquetas a los nombres de grupo. dataframe.groupby()

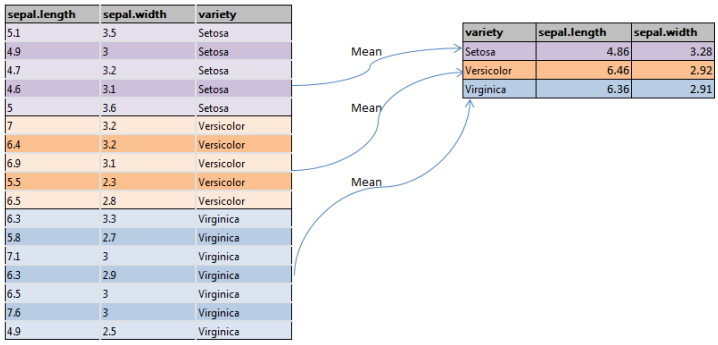
Desarrollo de Actividades
En este tutorial, se centrará en tres conjuntos de datos: 1. El conjunto de datos del Congreso de los Estados Unidos contiene información pública sobre miembros históricos del Congreso e ilustra varias capacidades fundamentales de ..groupby() 2. El conjunto de datos de calidad del aire contiene lecturas periódicas del sensor de gas. Esto le permitirá trabajar con flotadores y datos de series temporales.

3. El conjunto de datos del agregado de noticias que contiene metadatos sobre varios cientos de miles de artículos de noticias. Trabajarás con cadenas y harás munging de texto basado en groupby.

Lecciones Aprendidas

Una operación groupby implica alguna combinación de dividir el objeto, aplicar una función y combinar los resultados. Esto se puede utilizar para agrupar grandes cantidades de datos y calcular operaciones en estos grupos.

Evidencias



Podemos agrupar por diferentes niveles de un índice jerárquico utilizando el parámetro *level*:

```
>>> arrays = [['Falcon', 'Falcon', 'Parrot', 'Parrot'],
...           ['Captive', 'Wild', 'Captive', 'Wild']]
>>> index = pd.MultiIndex.from_arrays(arrays, names=('Animal', 'Type'))
>>> df = pd.DataFrame({'Max Speed': [390., 350., 30., 20.]},
...                   index=index)
>>> df
             Max Speed
Animal Type
Falcon Captive      390.0
       Wild         350.0
Parrot Captive       30.0
       Wild          20.0
>>> df.groupby(level=0).mean()
        Max Speed
Animal
Falcon      370.0
Parrot       25.0
>>> df.groupby(level="Type").mean()
         Max Speed
Type
Captive      210.0
Wild         185.0
```

```python
import pandas as pd

dtypes = {
    "first_name": "category",
    "gender": "category",
    "type": "category",
    "state": "category",
    "party": "category",
}
df = pd.read_csv(
    "groupby-data/legislators-historical.csv",
    dtype=dtypes,
    usecols=list(dtypes) + ["birthday", "last_name"],
    parse_dates=["birthday"]
)
```

➢ Semana 21-04-2022

Síntesis

El análisis de la ANOVA o de la varianza de un factor es un método estadístico para examinar las diferencias en las medidas de tres o más grupos. Se emplea cuando tenemos una única variable o *factor* independiente y el objetivo es investigar si las variaciones o diferentes *niveles* de ese factor tienen un efecto medible sobre una variable dependiente.
El ANOVA de un factor es un método estadístico para probar la hipótesis nula (*H0*) de que tres o más medias poblacionales son iguales frente a la hipótesis alternativa (*Ha*) de que al menos una de las medias es diferente.

Desarrollo de Actividades

¿Qué es la prueba ANOVA?
Una prueba de análisis de varianza, o ANOVA, se puede considerar como una generalización de las pruebas t para más de 2 grupos. La prueba t independiente se utiliza para comparar las medias de una afección entre dos grupos. ANOVA se utiliza cuando queremos comparar las medias de una condición entre más de dos grupos.
ANOVA prueba si hay una diferencia en la media en algún lugar del modelo (probando si hubo un efecto general), pero no nos dice dónde está la diferencia (si la hay). Para encontrar dónde está la diferencia entre los grupos, tenemos que realizar pruebas post-hoc.

Para realizar cualquier prueba, primero debemos definir la hipótesis nula y alternativa:

Hipótesis nula – No hay diferencias significativas entre los grupos
Hipótesis alternativa – Hay una diferencia significativa entre los grupos

Básicamente, ANOVA se realiza comparando dos tipos de variación, la variación entre las medias de la muestra, así como la variación dentro de cada una de las muestras. La fórmula mencionada a continuación representa estadísticas unidireccionales de la prueba de Anova.

El resultado de la fórmula ANOVA, la estadística F (también llamada relación F), permite el análisis de múltiples grupos de datos para determinar la variabilidad entre muestras y dentro de las muestras.

En general, si el valor p asociado con el F es menor que 0,05, entonces la hipótesis nula es rechazada y la hipótesis alternativa es apoyada. Si se rechaza la hipótesis nula, podemos concluir que las medias de todos los grupos no son iguales.

Lecciones Aprendidas

El profesor explica para que funciona el ANOVA, básicamente se utiliza para hacer test de cuales variables que tenemos en un set de datos permiten hacer predicciones.
Si en un proyecto se requiere hacer una prueba para validar una hipótesis, es ahí que toma relevancia Anova.

Puede utilizar una variable, dos o más variables que interaccionan entre sí o pueden ser independientes.

Evidencias

$$F = \frac{\text{Explained Variance}}{\text{Unexplained variance}} \text{ or } \frac{\text{Variance between groups}}{\text{Variance within groups}} = \frac{\text{Sum of squares between groups}}{\text{Sum of squares for error}}$$
$$= \frac{MST}{MSE} = \frac{\text{Mean squared error treatments}}{\text{Mean squared error}} = \frac{SS_B/_{DF_T}}{SS_W/_{DF_E}} = \frac{SS_B/_{(k-1)}}{SS_W/_{(N-k)}}$$

Where,

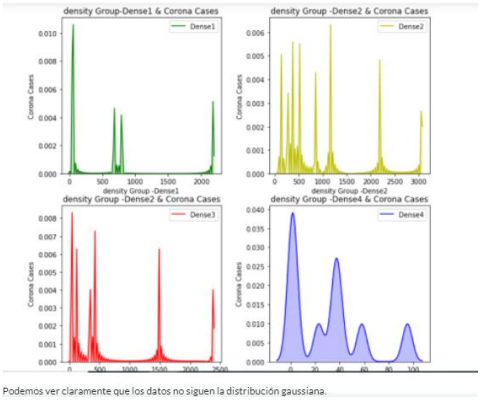
$$SS_B = \sum_i n_i (\bar{y}_i - \bar{y})^2$$

$$SS_w = \sum_{ij} (y_{ij} - \bar{y}_i)^2$$

Where,

y_i – sample mean in the i^{th} group
n_i – number of observation in the i^{th} group
$\bar{y}$ – total mean of the data
k – total number of the groups

y_{ij} – j^{th} observation in the out of k groups
N – Overall sample size



Podemos ver claramente que los datos no siguen la distribución gaussiana.