Tabla de contenido

Atribución - No comercial - Compartir igual:

1. Objetivo de la unidad 3

Comprender el manejo de bases de datos en Python a través de librerías especializadas, para realizar un análisis exploratorio de datos, que permita obtener estadísticas descriptivas y representar los datos mediante visualizaciones claras y comprensibles, que faciliten la toma de decisiones.

2. Tema 2: Análisis Exploratorio de Datos - Ejemplo

2.1 Introducción

En este documento, se ilustra una metodología para realizar un análisis exploratorio de datos utilizando el lenguaje de programación Python. A lo largo de la exposición, se mostrarán ejemplos prácticos de código Python que permitirán explorar la estructura, distribución y relaciones dentro del conjunto de datos anexo a la Unidad 3 de nuestro curso. Esta aproximación no solo ofrecerá una comprensión profunda de los datos, sino que también facilitará la identificación de patrones y tendencias relevantes para la toma de decisiones informadas.

Objetivos

1. Comprender la estructura de los datos.
2. Identificar la distribución de las variables.
3. Detectar valores atípicos.
4. Explorar relaciones entre variables.
5. Visualizar patrones y tendencias.
6. Evaluar la calidad de los datos.
7. Preparar los datos para análisis posteriores.
8. Generar ideas para futuras investigaciones.

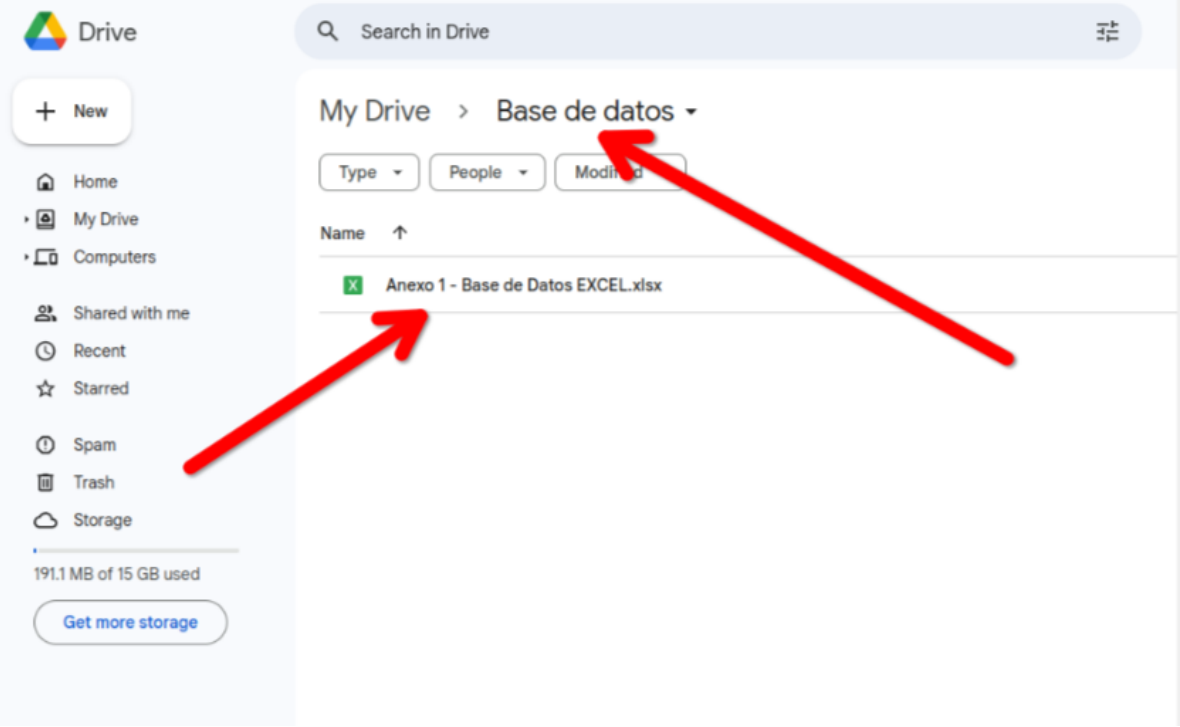
2.2 Análisis exploratorio de datos

Atribución - No comercial - Compartir igual:

Primer paso

Para iniciar este ejercicio de análisis exploratorio de datos, se requiere:

1. Descargar el archivo excel llamado "Anexo 1 - Base de Datos EXCEL.xlsx", que se encuentra anexo a la Unidad 3 del curso.
2. Subir el archivo excel a su Google Drive en una carpeta llamada "Bases de datos".
3. Procedemos a ingresar a Google Colab https://colab.google/



Carga de la Base de Datos en Google Colab

Ahora es tiempo de cargar nuestra base de datos como un DataFrame en Google Colab. Para ello usamos la librería "Pandas".

Es importante que replique el código en su propio Notebook de Google Colab. El ejecutar este paso, la plataforma Google Colab solicita permisos para acceder a Google Drive. Es totalmente seguro.

Nota: La biblioteca pandas es una herramienta de manipulación y análisis de datos de código abierto para el lenguaje de programación Python. Proporciona estructuras de datos de alto rendimiento y fáciles de usar, como DataFrame y Series, que permiten trabajar con datos de manera eficiente y flexible. Pandas es ampliamente utilizada en ciencia

Atribución - No comercial - Compartir igual: CC BY NC ND

de datos, análisis financiero, ingeniería y otras disciplinas donde se requiere el manejo y análisis de datos tabulares.

Link: https://pandas.pydata.org/

```
# Importamos las librerías necesarias en este paso
from google.colab import drive
import pandas as pd

# Montar Google Drive
drive.mount('/content/drive')

# Especifica la ruta del archivo Excel en Google Drive
ruta_archivo = "/content/drive/MyDrive/Base de datos/Anexo 1 - Base de Datos EXCEL.xlsx"

# Carga el archivo Excel en un DataFrame
df = pd.read_excel(ruta_archivo, sheet_name=1, header=3, index_col=0)

# Muestra las primeras filas del DataFrame para verificar que se haya cargado correctamente
df.head()
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

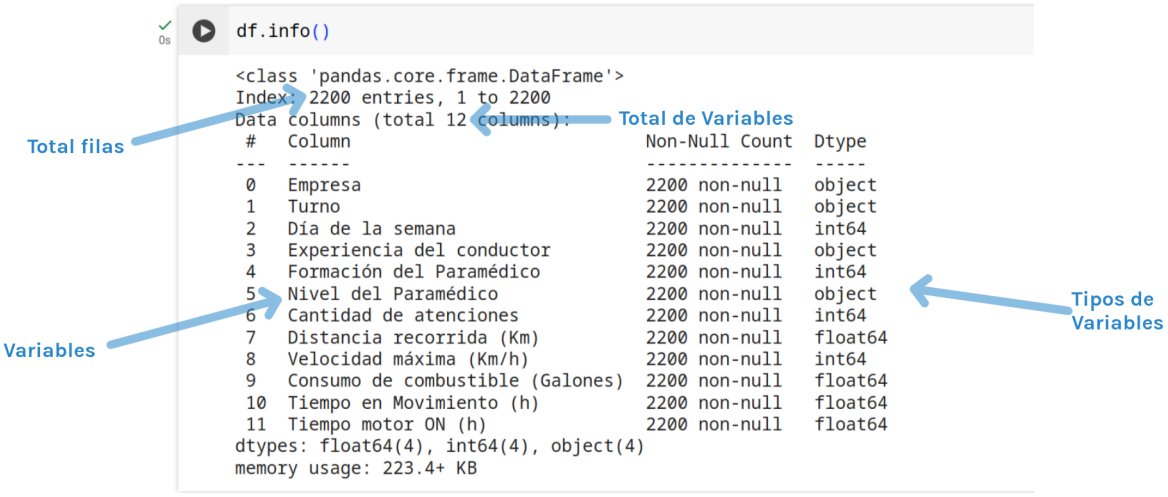
```
Mounted at /content/drive
```

Servicio	Empresa	Turno	Día de la semana	Experiencia del conductor	Formación del Paramédico	Nivel del Paramédico	Cantidad de atenciones	Distancia recorrida (Km)	Velocidad máxima (Km/h)	Consumo de combustible (Galones)	Tiempo en Movimiento (h)	Tiempo motor ON (h)
1	TAC	D	5	Principiante	4	I	8	78.4	82	3.1	2.8	3.08
2	TAD	N	4	Principiante	3	B	5	46.2	89	1.9	2.1	2.31
3	TAB	N	4	Experto	4	B	6	62.4	99	2.5	2.6	4.68
4	TAB	D	2	Intermedio	3	B	7	70.2	95	3.0	2.6	3.12
5	TAC	N	7	Principiante	2	I	8	75.6	87	2.8	2.8	3.64

Información básica del DataFrame

La función **df.info()** en pandas sirve para proporcionar una visión general concisa de la estructura y la composición de un DataFrame. Al llamar a **df.info()**, se mostrará la siguiente información:

Atribución - No comercial - Compartir igual: CC BY NC ND



Estadísticas Descriptivas del DataFrame

La función **df.describe()** en pandas proporciona un resumen estadístico de las columnas numéricas de un DataFrame.

Nota: Al llamar a **df.describe()**, se calcularán las siguientes estadísticas descriptivas para cada columna numérica:

1. **Recuento (count):** El número de observaciones no nulas en la columna.
2. **Media (mean):** El promedio de los valores en la columna.
3. **Desviación estándar (std):** La medida de dispersión de los valores alrededor de la media.
4. **Valor mínimo (min):** El valor más bajo en la columna.
5. **Percentiles (25%, 50%, 75%):** Los valores que dividen los datos ordenados en cuatro partes iguales. El percentil 50% es la mediana.
6. **Valor máximo (max):** El valor más alto en la columna.

Una vez aplicado el comando, la siguiente imagen muestra el resultado obtenido:

```
df.describe()
```

	Día de la semana	Formación del Paramédico	Cantidad de atenciones	Distancia recorrida (Km)	Velocidad máxima (Km/h)	Consumo de combustible (Galones)	Tiempo en Movimiento (h)	Tiempo motor ON (h)
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	4.001364	2.500000	6.854091	68.443636	90.110000	2.852545	2.801455	4.052614
std	1.993965	1.118288	1.126425	10.784850	5.071718	0.538903	0.297187	0.770218
min	1.000000	1.000000	4.000000	41.800000	74.000000	1.300000	1.900000	2.310000
25%	2.000000	1.750000	6.000000	60.000000	87.000000	2.500000	2.600000	3.480000
50%	4.000000	2.500000	7.000000	67.500000	90.000000	2.800000	2.800000	4.030000
75%	6.000000	3.250000	8.000000	75.600000	93.000000	3.200000	3.000000	4.620000
max	7.000000	4.000000	10.000000	101.500000	107.000000	4.600000	3.800000	6.480000

Atribución - No comercial - Compartir igual: CC BY NC ND

La función **df.describe(include=['object'])** en pandas proporciona un resumen estadístico de las columnas que contienen datos de tipo objeto (es decir, variables categóricas o de texto) en un DataFrame.

Nota: Al llamar a esta función, se calcularán las siguientes estadísticas descriptivas para cada columna de tipo objeto:

1. **Recuento (count):** El número de observaciones no nulas en la columna.
2. **Recuento único (unique):** El número de valores únicos en la columna.
3. **Valor superior (top):** El valor más frecuente en la columna (la moda).
4. **Frecuencia del valor superior (freq):** La frecuencia del valor más frecuente en la columna.

Una vez aplicado el comando, la siguiente imagen muestra el resultado obtenido:

```
df.describe(include=['object'])
```

	Empresa	Turno	Experiencia del conductor	Nivel del Paramédico
count	2200	2200	2200	2200
unique	5	2	3	3
top	TAC	D	Intermedio	B
freq	596	1114	1200	1200

Matriz de Correlación

La siguiente tabla proporciona un código para crear la matriz de correlación entre las variables numéricas de un DataFrame:

```
# Creamos un DataFrame adicional que excluye las variables
categóricas, conteniendo solo variables numéricas.
# Utilizamos el método drop() para eliminar las columnas
especificadas del DataFrame original 'df'.
# El parámetro axis=1 indica que estamos eliminando columnas
(variables), no filas.
corr_numericas = df.drop(['Empresa', 'Turno', 'Experiencia del
```

Atribución - No comercial - Compartir igual:

```
conductor', 'Nivel del Paramédico'], axis=1).corr()

# Calculamos la matriz de correlaciones entre las variables
numéricas en el DataFrame 'corr_numericas'.
# Utilizamos el método corr() para calcular las correlaciones entre
todas las pares de variables numéricas en el DataFrame.
# Esto nos proporciona una visión general de cómo están relacionadas
las variables numéricas entre sí.
corr_numericas.corr()
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

	Día de la semana	Formación del Paramédico	Cantidad de atenciones	Distancia recorrida (Km)	Velocidad máxima (Km/h)	Consumo de combustible (Galones)	Tiempo en Movimiento (h)	Tiempo motor ON (h)
Día de la semana	1.000000	-0.136213	-0.423926	-0.422935	-0.157032	-0.400615	-0.427857	-0.408130
Formación del Paramédico	-0.136213	1.000000	-0.405710	-0.402780	-0.205684	-0.393870	-0.440057	-0.370927
Cantidad de atenciones	-0.423926	-0.405710	1.000000	0.998866	-0.411638	0.963128	0.814883	0.437132
Distancia recorrida (Km)	-0.422935	-0.402780	0.998866	1.000000	-0.423140	0.968231	0.820998	0.444406
Velocidad máxima (Km/h)	-0.157032	-0.205684	-0.411638	-0.423140	1.000000	-0.397693	-0.447540	-0.361252
Consumo de combustible (Galones)	-0.400615	-0.393870	0.963128	0.968231	-0.397693	1.000000	0.755224	0.381278
Tiempo en Movimiento (h)	-0.427857	-0.440057	0.814883	0.820998	-0.447540	0.755224	1.000000	0.718925
Tiempo motor ON (h)	-0.408130	-0.370927	0.437132	0.444406	-0.361252	0.381278	0.718925	1.000000

Interpretación:

La matriz de correlaciones es una herramienta útil para comprender las relaciones entre las variables en un conjunto de datos.

Los valores de la matriz varían entre -1 y 1, donde 1 indica una correlación positiva perfecta, -1 indica una correlación negativa perfecta, y 0 indica ausencia de correlación. Un valor cercano a 1 o -1 sugiere una relación fuerte entre las variables, mientras que un valor cercano a 0 indica una relación débil o nula.

Es importante recordar que la correlación no implica causalidad, por lo que es necesario realizar análisis adicionales para establecer relaciones causales entre las variables

Correlaciones más fuertes

Atribución - No comercial - Compartir igual: CC BY NC ND

La siguiente tabla muestra el código utilizado para generar una lista de los 15 índices de correlaciones más grandes en nuestro conjunto de datos.

```
# Obtenemos la matriz de correlaciones entre las variables numéricas
# corr_numericas es la matriz de correlaciones previamente calculada
# Utilizamos unstack() para convertir la matriz en una serie
# Luego, ordenamos los valores de mayor a menor (ascending=False)
max_corr = corr_numericas.unstack().sort_values(ascending=False)

# Imprimimos las variables que tienen las correlaciones más altas
# Filtramos aquellos valores que no son 1.0 (correlación perfecta con sí mismos)
# Imprimimos los 15 primeros valores
print("\nVariables con las correlaciones más altas (excluyendo correlación perfecta):")
print(max_corr[max_corr != 1.0].head(15))
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

```
Variables con las correlaciones más altas (excluyendo correlación perfecta):
Cantidad de atenciones            Distancia recorrida (Km)            0.966409
Distancia recorrida (Km)          Cantidad de atenciones              0.966409
Consumo de combustible (Galones)  Distancia recorrida (Km)            0.833070
Distancia recorrida (Km)          Consumo de combustible (Galones)    0.833070
Cantidad de atenciones            Consumo de combustible (Galones)    0.806123
Consumo de combustible (Galones)  Cantidad de atenciones              0.806123
Distancia recorrida (Km)          Tiempo en Movimiento (h)            0.651875
Tiempo en Movimiento (h)          Distancia recorrida (Km)            0.651875
                                  Cantidad de atenciones              0.636389
Cantidad de atenciones            Tiempo en Movimiento (h)            0.636389
Tiempo motor ON (h)               Tiempo en Movimiento (h)            0.555573
Tiempo en Movimiento (h)          Tiempo motor ON (h)                 0.555573
Consumo de combustible (Galones)  Tiempo en Movimiento (h)            0.544471
Tiempo en Movimiento (h)          Consumo de combustible (Galones)    0.544471
Tiempo motor ON (h)               Distancia recorrida (Km)            0.367795
dtype: float64
```

Interpretación:

Estos resultados muestran las correlaciones más altas entre las variables numéricas en tu conjunto de datos. Cada línea indica la correlación entre dos variables.

Atribución - No comercial - Compartir igual: CC BY NC ND

Por ejemplo: La primera línea indica que la variable "Cantidad de atenciones" y la variable "Distancia recorrida (Km)" tienen una correlación de 0.966409, lo que sugiere una relación muy fuerte entre la cantidad de atenciones y la distancia recorrida.

Estas correlaciones pueden proporcionar información útil sobre cómo las variables están relacionadas entre sí en tu conjunto de datos.

Histograma de variables numéricas

La siguiente tabla muestra el código utilizado para generar histogramas para cada una de las variables numéricas:

```
# Importa la librería matplotlib, especializada en gráficos
import matplotlib.pyplot as plt
# Selecciona solo las variables numéricas del DataFrame
variables_numericas = df.select_dtypes(include=['int64', 'float64'])
# Configura el tamaño de la figura
plt.figure(figsize=(10, 6))

# Itera sobre cada variable numérica y crea histogramas
for i, col in enumerate(variables_numericas.columns):
    # Divide la figura en subgráficos
    plt.subplot(2, len(variables_numericas.columns)//2, i+1)
    # Crea un histograma para la variable actual
    plt.hist(df[col], bins=20, color='skyblue', edgecolor='black')
    # Añade un título al histograma
    plt.title(col)
    # Etiqueta el eje x con el nombre de la variable
    plt.xlabel(col)
    # Etiqueta el eje y con 'Frecuencia'
    plt.ylabel('Frecuencia')
# Ajusta el diseño de la figura para que no haya superposiciones
plt.tight_layout()
# Muestra la figura
plt.show()
```

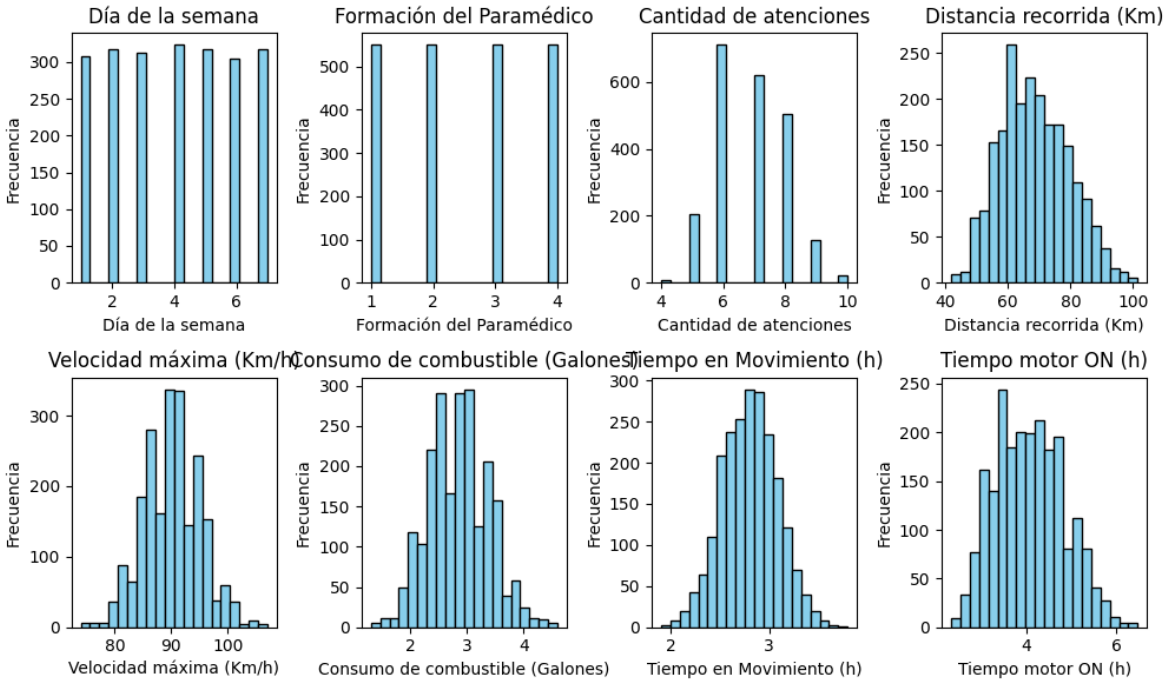
Matplotlib:

Atribución - No comercial - Compartir igual: CC BY NC ND

Es una biblioteca de visualización de datos en Python que proporciona herramientas para crear gráficos estáticos, interactivos y animados. Es ampliamente utilizada en el campo de la ciencia de datos, la ingeniería, la investigación académica y más debido a su flexibilidad y facilidad de uso.

Link: https://matplotlib.org/

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:



Interpretación:

1. **Forma de la distribución:** Observa la forma general del histograma. ¿Es simétrico, sesgado a la izquierda o sesgado a la derecha?
2. **Centralidad:** La ubicación de la media y la mediana en el histograma puede dar una idea de la centralidad de los datos.
3. **Dispersión:** Se puede inferir observando la amplitud de la distribución en el histograma. Una distribución más amplia sugiere una mayor variabilidad en los datos.
4. **Valores atípicos:** Son valores inusualmente altos o bajos en la distribución. Estos se representan como barras individuales o picos en el extremo de la distribución en el histograma.

Atribución - No comercial - Compartir igual:

5. **Modos:** Es el valor que ocurre con mayor frecuencia. Puedes identificar los modos en un histograma observando los picos o las áreas de mayor concentración de barras.
6. **Número de intervalos (bins):** La elección del número de intervalos en el histograma puede influir en la interpretación de la distribución. Un número demasiado bajo de intervalos puede ocultar detalles importantes en la distribución, mientras que un número demasiado alto puede generar ruido y hacer que la interpretación sea más difícil.

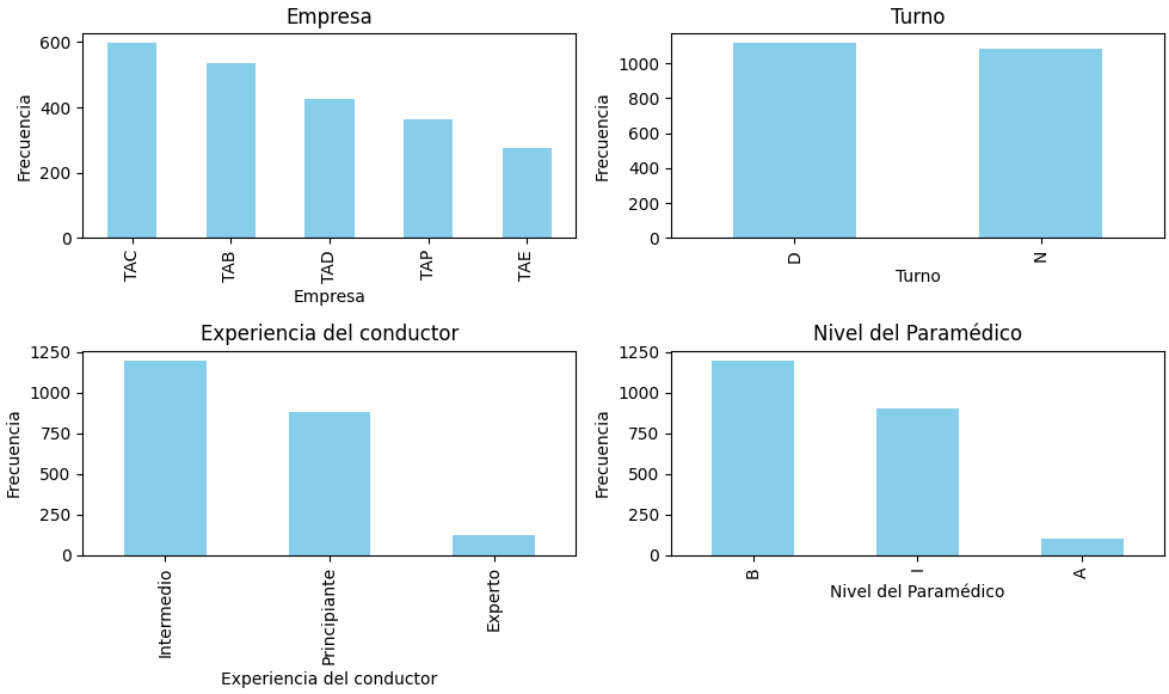
Diagrama de barras de variables categóricas

La siguiente tabla muestra el código utilizado para generar diagramas de barras para cada una de las variables categóricas:

```
# Selecciona solo las variables categóricas del DataFrame
variables_categoricas = df.select_dtypes(include=['object'])
# Configura el tamaño de la figura
plt.figure(figsize=(10, 6))

# Itera sobre cada variable categórica y crea gráficos de barras
for i, col in enumerate(variables_categoricas.columns):
    # Divide la figura en subgráficos para cada variable categórica
    plt.subplot(2, len(variables_categoricas.columns)//2, i+1)
    # Crea un gráfico de barras para la frecuencia de valores de la
variable categórica
    df[col].value_counts().plot(kind='bar', color='skyblue')
    # Añade un título al gráfico con el nombre de la variable
    plt.title(col)
    # Etiqueta el eje x con el nombre de la variable categórica
    plt.xlabel(col)
    # Etiqueta el eje y con 'Frecuencia'
    plt.ylabel('Frecuencia')
# Ajusta el diseño de la figura para evitar superposiciones
plt.tight_layout()
# Muestra la figura
plt.show()
```

Atribución - No comercial - Compartir igual: CC BY NC ND

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:



Interpretación:

1. **Frecuencia de valores:** La altura de cada barra representa la frecuencia de valores en una categoría específica.
2. **Comparación de categorías:** Los diagramas de barras permiten comparar la frecuencia de valores entre diferentes categorías.
3. **Ordenamiento:** En algunos casos, es útil ordenar las barras de manera ascendente o descendente según la frecuencia de los valores. Esto puede revelar patrones o tendencias en los datos.
4. **Identificación de valores dominantes:** Busca las barras más altas. Estas representan los valores más frecuentes en los datos y pueden proporcionar información sobre las categorías más comunes.
5. **Análisis de distribución:** Los diagramas de barras también pueden mostrar la distribución de los valores a lo largo de diferentes categorías.
6. **Comunicación efectiva:** Los diagramas de barras son una forma efectiva de comunicar información a audiencias no técnicas. Son

Atribución - No comercial - Compartir igual:

fáciles de entender y pueden utilizarse para resumir datos de manera clara y concisa.

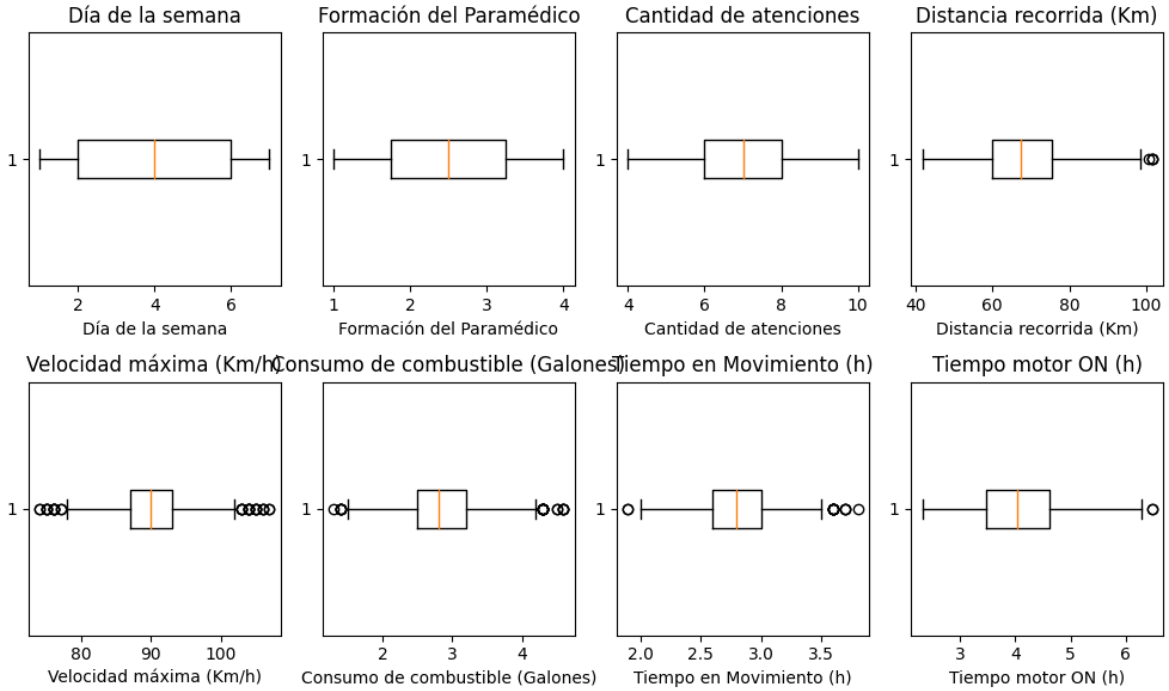
Diagramas tipo cajas

La siguiente tabla muestra el código utilizado para generar un diagrama tipo caja para cada una de las variables numéricas:

```
import pandas as pd
import matplotlib.pyplot as plt

# Selecciona solo las variables numéricas
variables_numericas = df.select_dtypes(include=['int64', 'float64'])

# Configura el tamaño de la figura
plt.figure(figsize=(10, 6))

# Itera sobre cada variable numérica y crea boxplots
for i, col in enumerate(variables_numericas.columns):
    plt.subplot(2, len(variables_numericas.columns)//2, i+1)
    plt.boxplot(df[col], vert=False)
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('')

# Ajusta el diseño de la figura
plt.tight_layout()

# Muestra la figura
plt.show()
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

Atribución - No comercial - Compartir igual:



Interpretación:

1. **Identificación de la Mediana:** La línea dentro de la caja (color naranja en el gráfico) representa la mediana (Q2) de los datos. Es un indicador de la tendencia central.
2. **Dispersión de los Datos:** El tamaño de la caja (Q3 - Q1) muestra el rango intercuartílico (IQR), que mide la dispersión central de los datos. Un IQR grande indica mayor dispersión y viceversa.
3. **Simetría y Sesgo:** Si la mediana está centrada en la caja, los datos son simétricos. Si la mediana está más cerca de Q1 o Q3, los datos pueden estar sesgados. Si uno de los bigotes es significativamente más largo que el otro, esto indica sesgo en la distribución de los datos.
4. **Valores Atípicos:** Los puntos fuera de los bigotes se consideran valores atípicos. Estos pueden ser casos extremos que requieren una mayor investigación.

Diagrama de líneas

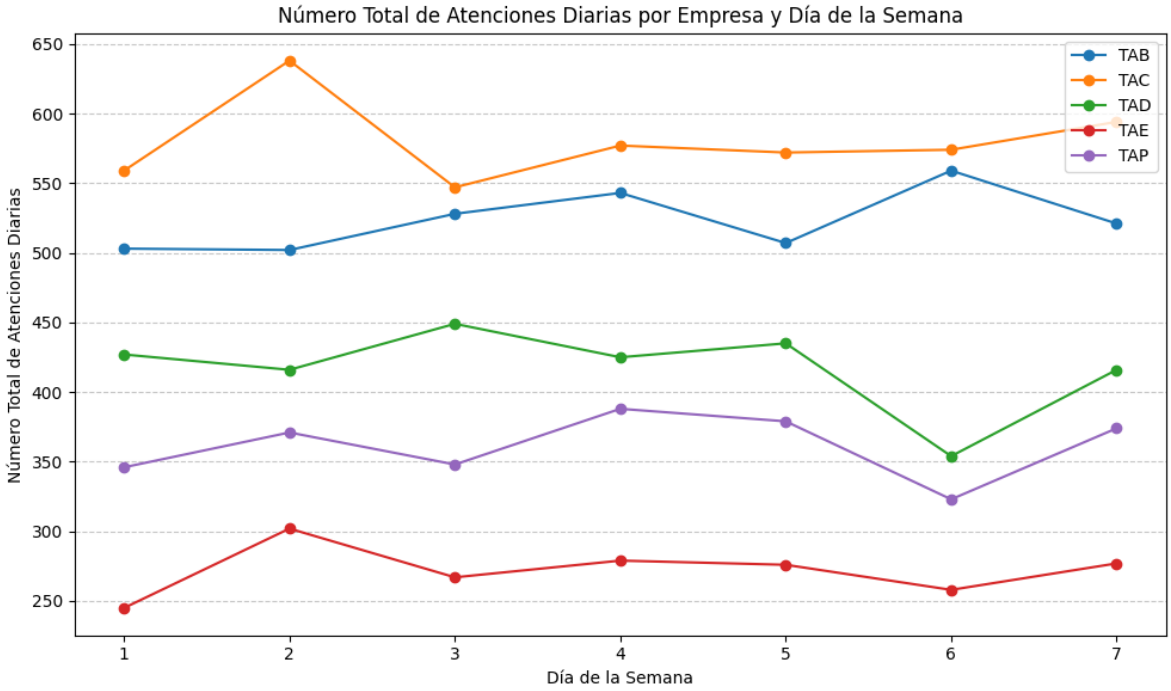
La siguiente tabla muestra el código utilizado para generar un diagrama de líneas en el que se compara la cantidad de atenciones por cada una de las empresas de ambulancias y en cada uno de los días:

Atribución - No comercial - Compartir igual:

```
# Agrupar por empresa y día de la semana, y contar la cantidad de atencones diarias
total_entregas_por_empresa_y_dia = df.groupby(['Empresa', 'Día de la semana'])['Cantidad de atenciones'].sum().unstack()

# Crear el gráfico de líneas para cada empresa
plt.figure(figsize=(10, 6))
for empresa in total_entregas_por_empresa_y_dia.index:
    plt.plot(total_entregas_por_empresa_y_dia.columns, total_entregas_por_empresa_y_dia.loc[empresa], marker='o', label=empresa)

# Agregar etiquetas y título
plt.title('Número Total de Atenciones Diarias por Empresa y Día de la Semana')
plt.xlabel('Día de la Semana')
plt.ylabel('Número Total de Atenciones Diarias')
plt.xticks(total_entregas_por_empresa_y_dia.columns) # Etiquetas del eje x como los días de la semana
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.legend()

# Mostrar el gráfico
plt.show()
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

Atribución - No comercial - Compartir igual: CC BY NC ND



Interpretación:

1. **Tendencias Temporales:** En este caso, el eje x representa el tiempo (días), puedes observar cómo cambian los valores de la variable a lo largo del tiempo. Ver un aumento o disminución constante en las líneas podría indicar una tendencia temporal.
2. **Comparación de Series:** Dado que hay múltiples líneas en el gráfico, puedes comparar las tendencias entre ellas.
3. **Puntos de Inflexión:** Observa los puntos donde la línea cambia de dirección. Estos puntos pueden indicar cambios significativos en la variable que estás analizando.
4. **Variabilidad y Estacionalidad:** Si hay fluctuaciones en las líneas, especialmente si se repiten en intervalos regulares, esto podría indicar variabilidad o estacionalidad en los datos.
5. **Valores Atípicos:** Observa cualquier valor que se aleje significativamente de la tendencia general de la línea. Estos valores atípicos podrían indicar eventos inusuales o errores en los datos que requieren una investigación adicional.

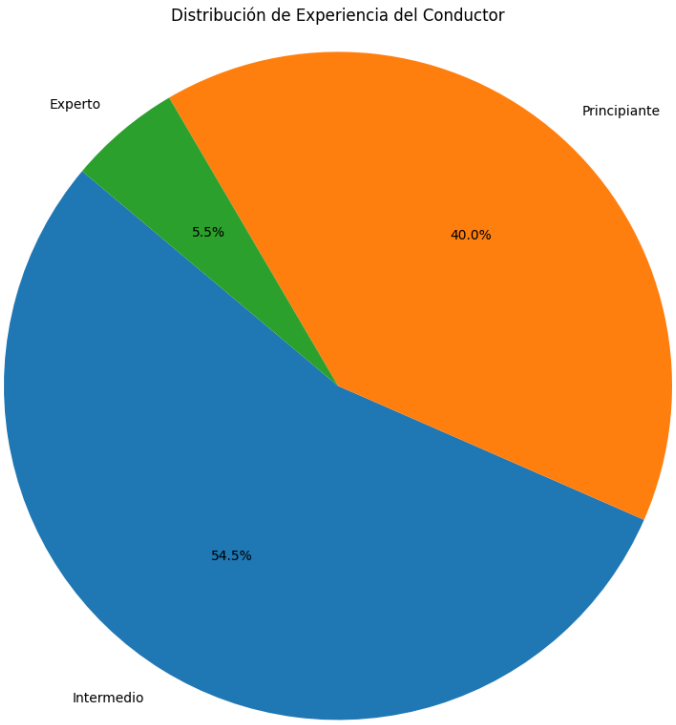
Diagrama tipo pastel

La siguiente tabla muestra el código utilizado para generar un diagrama tipo pastel para observar la distribución de la experiencia del conductor a lo largo de toda la base de datos:

Atribución - No comercial - Compartir igual: CC BY NC ND

```
# Contamos los valores de la variable "Experiencia del conductor"
conteo_experiencia = df['Experiencia del conductor'].value_counts()

# Creamos el diagrama de pie
plt.figure(figsize=(8, 8))
plt.pie(conteo_experiencia, labels=conteo_experiencia.index,
autopct='%1.1f%%', startangle=140)

# Agregamos título
plt.title('Distribución de Experiencia del Conductor')

# Mostramos el diagrama de pie
plt.axis('equal') # Para asegurar que el gráfico de pie sea circular
plt.tight_layout()
plt.show()
```

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:



Atribución - No comercial - Compartir igual: CC BY NC ND

Interpretación:

1. **Proporción de Categorías:** Cada sector del diagrama de pie representa una categoría de la variable categórica. La proporción del área de cada sector en relación con el círculo completo indica la proporción de datos que pertenecen a esa categoría en el conjunto de datos total.
2. **Categorías Dominantes:** Los sectores más grandes en el diagrama de pie representan las categorías que tienen una mayor proporción de datos en el conjunto de datos total.
3. **Categorías Menos Representadas:** Los sectores más pequeños en el diagrama de pie representan las categorías que tienen una menor proporción de datos en el conjunto de datos total.
4. **Comparación Relativa:** Puedes comparar fácilmente la proporción de diferentes categorías entre sí utilizando el diagrama de pie.
5. **Resumen Visual:** El diagrama de pie proporciona un resumen visual claro y conciso de la distribución de una variable categórica. Es especialmente útil cuando hay un número limitado de categorías y quieres obtener una impresión rápida de su distribución relativa.

Gráfico de dispersión

La siguiente tabla muestra el código utilizado para generar un gráfico de dispersión para cinco variables numéricas escogidas:

```
# Importamos la librería Seaborn, la cual es una biblioteca de
visualización en Python
# basada en matplotlib
import seaborn as sns


# Seleccionamos las variables numéricas a incluir en el pair plot
variables_numericas = ['Distancia recorrida (Km)', 'Velocidad máxima
(Km/h)', 'Consumo de combustible (Galones)',
'Tiempo en Movimiento (h)', 'Tiempo motor ON (h)']

# Creamos el pair plot utilizando Seaborn
# El pair plot muestra la relación entre todas las combinaciones de
```

Atribución - No comercial - Compartir igual: CC BY NC ND

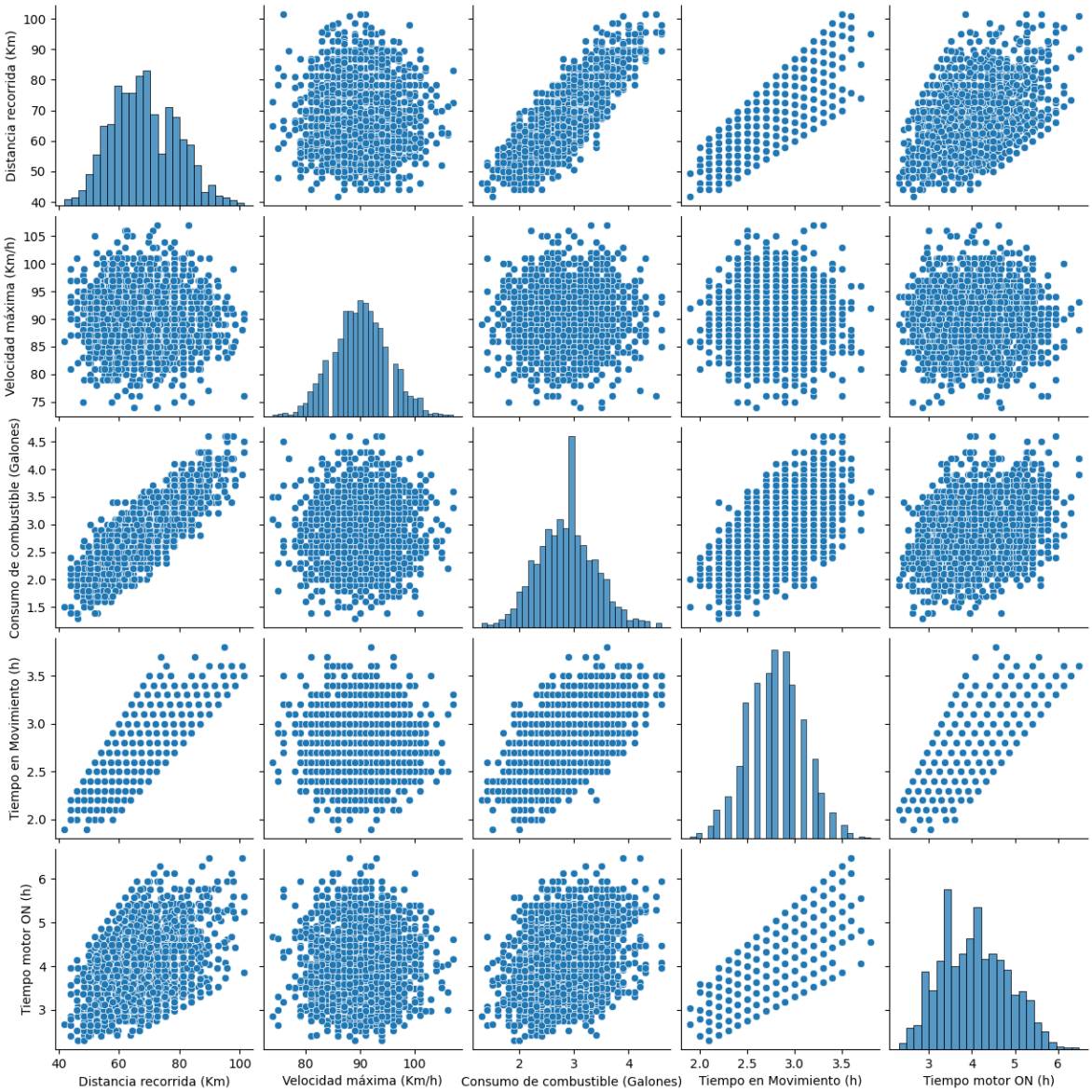
```
variables numéricas en un solo panel de gráficos
sns.pairplot(df[variables_numericas])

# Mostramos el gráfico
plt.show()
```

Seaborn: Es una biblioteca de visualización de datos en Python que se basa en Matplotlib y proporciona una interfaz de alto nivel para crear gráficos estadísticos atractivos y descriptivos. Seaborn se utiliza comúnmente en entornos de análisis de datos y ciencia de datos para explorar y visualizar conjuntos de datos de manera efectiva.

Link: https://seaborn.pydata.org/

Si todo el código ingresado es correcto, al hacer clic en el botón play se deberá visualizar el siguiente resultado:

Atribución - No comercial - Compartir igual:



Interpretación:

1. Diagonal Principal: La diagonal principal del pair plot muestra histogramas de las distribuciones univariadas de cada variable numérica. Esto te da una idea de la distribución de cada variable por sí sola.
2. Triángulo Superior e Inferior: El triángulo superior y el triángulo inferior del pair plot muestran gráficos de dispersión de todas las combinaciones posibles de pares de variables numéricas. Cada punto en un gráfico de dispersión representa una observación en tus datos.

Atribución - No comercial - Compartir igual: CC BY NC ND

3. Tendencias Generales: Observa la dirección general de las tendencias en los gráficos de dispersión. Si los puntos tienden a seguir una línea diagonal de arriba a abajo, indica una correlación positiva entre las variables. Si los puntos tienden a alejarse de la línea diagonal, indica una correlación negativa. Si no hay una tendencia clara, podría indicar una falta de relación entre las variables.
4. Correlación: Presta atención a la distribución de puntos en los gráficos de dispersión y a la concentración de puntos alrededor de la línea diagonal. Una mayor concentración de puntos cerca de la línea diagonal indica una correlación más fuerte entre las variables.

2.3 Resumen

En el análisis exploratorio de datos se han obtenido hallazgos significativos sobre la base de datos investigada. En la primer parte correspondiente a métricas descriptivas, se tienen:

1. Nuestra base de datos consta de 12 variables (8 numéricas y 4 categóricas) y un total de 2200 entradas o filas.
2. Se observan métricas descriptivas para las variables numéricas. Estas métricas incluyen: la media, mínimo, máximo, desviación estándar, etc.
3. Se estableció que la empresa de ambulancias "TAC" es quien lidera en la cantidad de servicios con un total de 596.
4. Algunas de las variables numéricas poseen una fuerte correlación positiva.

¿Cuáles otras conclusiones significativas puedes obtener?

3. Bibliografía

- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 92-111). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1
- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 125-131). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1
- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 141-145). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1

Atribución - No comercial - Compartir igual: CC BY NC ND

- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 150-162). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1
- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 239-249). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1
- ✓ Nelli, F. (2018). Python Data Analytics: With Pandas, NumPy and Matplotlib. Apress, Berkeley, CA, (pp. 276-296). https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-1-4842-3913-1
- ✓ Gaitan, R. (2023). *Uso de las Librerías Pandas y Numpy en Python*. [Objeto_virtual_de_Informacion_OVI]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/54808
- ✓ Vargas, M. A. (2022). *Carga de Datos en Python desde Excel*. [Objeto_virtual_de_Informacion_OVI]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/49755

Atribución - No comercial - Compartir igual: CC BY NC ND