ANALISIS DE DATOS PARA ESTADISTICAS DE POKEMON

(PYTHON, PANDAS, NUMPY, MATHPLOIT, POWER BI)

exploración, Limpieza, visualización y hallazgos para las estadísticas de Pokémon de primera a sexta generación.

Carlos Mario Hoyos Ríos

Noviembre 2025

Repositorio

https://github.com/ingcarlosmariohoyos/Pokemon_analitycs

Contenido

RESUMEN EJECUTIVO

Este proyecto se desarrolló a partir de un conjunto de datos obtenido en Kaggle que reúne las estadísticas base de los Pokémon desde la primera hasta la sexta generación. A partir de esta información, realicé un proceso completo de exploración, limpieza y análisis utilizando Python, apoyándome en librerías como Pandas, NumPy, Matplotlib y Seaborn. Posteriormente, integré los resultados en Power BI para construir visualizaciones dinámicas que facilitaran una interpretación más clara del comportamiento de los datos.

Aunque el dataset incluye información relacionada con los tipos de los Pokémon, el enfoque principal del análisis se centró en las estadísticas base (HP, Ataque, Defensa, Ataque Especial, Defensa Especial y Velocidad). De esta manera, el objetivo fue identificar patrones, comparaciones relevantes y diferencias significativas entre categorías específicas como legendarios, megaevoluciones y generaciones. Este enfoque permitió obtener una visión más sólida del rendimiento potencial de cada especie sin profundizar directamente en la teoría de tipos del juego.

En conjunto, este proyecto representa un ejercicio sólido de análisis de datos, aplicando buenas prácticas en cada etapa del proceso y aprovechando herramientas ampliamente utilizadas en la industria. Fue una oportunidad para consolidar el uso de Python, mejorar el criterio analítico y construir visualizaciones profesionales en Power BI que resumen de forma clara los principales hallazgos del dataset.

OBJETIVO GENERAL

- Analizar las estadísticas base de los Pokémon de la primera a la sexta generación utilizando Python y Power BI, con el propósito de identificar patrones, diferencias relevantes y tendencias generales que permitan comprender mejor el rendimiento potencial de cada especie dentro del juego.

OBJETIVOS ESPECIFICOS

- Realizar la exploración y limpieza del dataset, garantizando que la información esté completa, coherente y preparada para su análisis.
- Estudiar las estadísticas base (HP, Ataque, Defensa, Ataque Especial, Defensa Especial y Velocidad) para identificar distribuciones, valores sobresalientes y relaciones que aporten a la interpretación del comportamiento de los Pokémon.
- Comparar categorías clave, como legendarios, megaevoluciones y distintas generaciones, con el fin de observar diferencias reales en sus estadísticas y entender cómo se posicionan unas respecto a otras.
- Construir visualizaciones estadísticas en Python que permitan representar los datos de forma clara, facilitando la identificación de patrones y hallazgos importantes.
- Diseñar un panel en Power BI que resuma los resultados principales y permita una lectura más dinámica y visual del análisis completo.
- Documentar cada etapa del proceso, dejando claros los criterios aplicados, los hallazgos obtenidos y las conclusiones que se derivan del estudio del dataset.

METODOLOGIA

Adquisición del dataset:

La adquisición de datos se realizó a través de la plataforma Kaggle. Se utilizo el dataset 'Pokémon with stats' (ID: abcsds/Pokémon).

Autor Original: Alberto Barradas.

URL de la Fuente: https://www.kaggle.com/datasets/abcsds/pokemon

Contenido: El dataset incluye las estadísticas base, tipo, generación y condición de legendario de 800 Pokémon.

Limpieza del dataset

Para empezar, primero es necesario revisar con que datos y tipo contamos en nuestro dataset por lo cual se utiliza el atributo columns de nuestro data set mostrando los siguientes resultados:

```
[9]: #Revision de sus columnas
     df.columns

[9]: Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defense',
            'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary'],
           dtype='object')

[11]: #verificacion de los tipos de datos y su cantidad
      df.info()
```

Luego se procede a limpiar los nulos con la función fillna con la cual se toma como parámetro principal un diccionario creado previamente con el valor que reemplazara las celdas con valores nulos:

```
[2]: #En el notebook anterior se evidencio ue el campo tipo 2 presentaba nulos por lo cual se procede a dar el manejo de estos campos
     #Generamos un diccionario que toma como clave el campo Type 2 y como valor la palabra no aplica para sustituir los campos vacios
     valor_tipo = {"Type 2":"No aplica"}
     #Generamos en dataframe nuevo el cual va aplicar la funcion fillna para reemplazar los campos vacios
     df_sin_na = df.fillna(valor_tipo)
     df_sin_na
```

Se puede evidenciar que ya no se presentan campos nulos en el dataframe

```
[3]: #comprobamos que no queden campos vacios aplicando la funcion isnull y sumando en cada columna cuantos nulos exites en este caso 0
     df_sin_na.isnull().sum()

[3]: #             0
     Name          0
     Type 1        0
     Type 2        0
     Total         0
     HP            0
     Attack        0
     Defense       0
     Sp. Atk       0
     Sp. Def       0
     Speed         0
     Generation    0
     Legendary     0
     dtype: int64
```

Luego se debe dar un nombre mas entendible a los campos o columnas del dataser por lo cual se genera nuevamente un diccionario donde se va a asignar el nuevo nombre que tendrá cada columna:

```
5]: # En este caso se genero otro diccionario llamado columnas donde la clave sera el nombre de la columna actual
    # y el valor sera el nuevo nombre que le daremos a la columna
    Columnas = {
       '#': 'Numero Dex',
       'Name': 'Nombre',
       'Type 1': 'Tipo Primario',
       'Type 2': 'Tipo Secundario',
       'Total': 'Estadisticas Totales',
       'HP': 'Puntos Salud',
       'Attack': 'Ataque',
       'Defense': 'Defensa',
       'Sp. Atk': 'Ataque Especial',
       'Sp. Def': 'Defensa Especial',
       'Speed': 'Velocidad',
       'Generation': 'Generacion',
       'Legendary': 'Legendario'
    }
    # creamos un nuevo dataframe que contendra los datos del dataset pero con el nuevo nombre de las columnas mas claros y para mayor entendimiento
    pokedex = df_sin_na.rename(columns = Columnas)
    pokedex
```

Transformaciones

Luego de tener un estándar listo para campos y eliminar duplicados se procede a categorizar y generar nuevas columnas que aporten más valor al dataframe. Empezando por la división entre Pokémon en estado base y mega evoluciones por lo cual se genera un filtro que permita sacar las mega evoluciones en primera instancia y luego los demás Pokémon excluidos quedaran en el grupo pokedex_base generando dos dataframe de acuerdo a la categoría.

```
[41]: # Ahora se van a categorizar el tipo de fase en la que se encuentra el pokemon primero se revisaran la cantida de megaevoluciones
      es_mega = pokedex['Nombre'].str.contains("mega", case = False)
      # generamos un dataset temporal que nos mostrara solo los pokemon que son megaevoluciones
      megas = pokedex[es_mega]
      print(f"Total de Pokémon Mega encontrados: {len(megas)}")

      Total de Pokémon Mega encontrados: 50

[43]: # Ahora vamos a sacar los que no son megaevoluciones y dejaremos los que estan en su estado base realizando una negacion en base a la variable
      # anterior que nos indicaba cual era megaevolucion
      no_es_mega = ~es_mega
      # En base a esta variable que nos indica cuales no son  megas sacamos la cantidad de pokemon en estado base
      pokedex_base = pokedex[no_es_mega]
      print(f"Total de Pokémon Base (no Mega) encontrados: {len(pokedex_base)}")
```

Eventualidad con Pokemon Meganium:

Posteriormente se encontró que se filtró un Pokémon que no tenía la categoría mega evolución el cual es Meganium por lo cual para solucionar este inconveniente se procede a realizar un filtro por el nombre en el dataframe donde se toma aparte esta fila y posteriormente se vuelve a generar el dataset excluyendo esta fila perteneciente al Pokémon.

```
# Para corregir el error anterior se crea una variable Meganium en donde el datframe buscara por la columna nombre una coincidencia con el nombre del
# Pokemon
Meganium = megas['Nombre'] == 'Meganium'
# creamos una nueva variable que tomara el resultado de la validacion anterior y la tendra a parte para insertarlo en el dataframe correcto
mover_pokemon = megas[Meganium]
# Ahora generaremos nuevamente el dataset con todas las filas que no contenan el nombre Meganium
Megas = megas[~Meganium]
```

Luego esa misma variable que contiene el pokemon lo agregamos a el dataframe pokedex_base que es donde debe pertenecer por lo cual con la función concat agregamos esta variable al dataset y comprobamos que este dentro de el con su correspondiente numero dex el cual es el numero único con el que esta registrado cada pokemon.

```
# por medio de la funcion concat anexamos la fila que tiene los datos del pokemon Meganium a el datframe pokedex_base
Pokedex_Base = pd.concat([pokedex_base, mover_pokemon])
# reseteamos los indices de las filas para ubicar el registro en el nivel que pertenece
Pokedex_Base = Pokedex_Base.reset_index(drop=True)
#hacemos la validacion rapida para ver si quedo agregado al dataframe
Filtro_Pokemon = Pokedex_Base["Nombre"] == "Meganium"
Mostrar_Pokemon = Pokedex_Base[Filtro_Pokemon]
print(Mostrar_Pokemon)
```

```
[15]: #Ordeno el df por el numero dex
      Pokedex_Base_Ordenado = Pokedex_Base.sort_values(by='Numero Dex', ascending=True)

      #reinicio el index del data frame
      Pokedex_Base_Ordenado = Pokedex_Base_Ordenado.reset_index(drop=True)

      #reviso si el pokemon quedo en su posicion correcta
      print("Verificando la posición de Meganium (Número 154):")
      # Mostramos las filas alrededor del 154
      # Evidenciamos que quedo en su numero correcto
      print(Pokedex_Base_Ordenado.loc[150:160, ['Nombre', 'Numero Dex']])
```

```
Verificando la posición de Meganium (Número 154):
         Nombre  Numero Dex
150         Mew         151
151   Chikorita         152
152     Bayleef         153
153    Meganium         154
154   Cyndaquil         155
155     Quilava         156
156  Typhlosion         157
157    Totodile         158
158    Croconaw         159
159  Feraligatr         160
160     Sentret         161
```

Para finalizar se realiza una conversión de estos dataframe el cual venia del dataset pokemon.csv en archivo parquet ya que por su composición es muy eficiente para cargar la información en diferentes fuentes donde se desee trabajar con los datos.

```
•[17]: # Esta linea se utilizo para realizar la conversion de los dataframe a formato parquet con el fin de que sea mas rapido el cargue de informacion
       # y utilizacion de la informacion.

       Pokedex_Base_Ordenado.to_parquet('Pokedex_Base_Ordenado.parquet', index=False)
       Megas.to_parquet('Megas.parquet', index=False)
```

Para dar un mejor entendimiento de los datos se procede a crear una serie de columnas nuevas que permitan analizar de manera mas especifica ciertos aspectos del dataset. Inicialmente se crea una columna llamada región para dar nombre a las generaciones con las cuales se cuentan en este dataset. Se procede a crear un nuevo diccionario con el la clave y el valor de cada región y posteriormente con la transformación map aplicamos esto a cada fila igualando el numero de la generación con la región correspondiente.

```
# Dar nombre a las regiones con el fin de que podamos tener una nueva columna que indique el nombre de cada generacion
# Nos valemos nuevamente de los diccionarios para generar el nombre de las regiones
mapeo_regiones = {
    1:'Kanto',
    2:'Jhoto',
    3:'Hoen',
    4:'Shinoh',
    5:'Unova',
    6:'Kalos',
}
#Se crea una columna llamada region y se le asigna el valor de clave valor del diccinario para darle nombre a las regiones
Pokedex_Base['Region'] = Pokedex_Base['Generacion'].map(mapeo_regiones)
Pokedex_Base
```

Luego se generó una agrupación para poder determinar la cantidad de criaturas por región para ver como esta distribuida la población del dataset generando el siguiente resultado.

```
5]: # para saber el numero de criaturas que hay por region utilizamos groupby por la nueva columna region y un count
    # para saber la cantidad por region con el numero dex el cual representa un numero unico para cada pokemon
    conteo_region = Pokedex_Base.groupby('Region')['Numero Dex'].count()
    print("\n Conteo de Pokémon por Región:")
    print(conteo_region)

     Conteo de Pokémon por Región:
    Region
    Hoen      140
    Jhoto     100
    Kalos      81
    Kanto     151
    Shinoh    115
    Unova     164
    Name: Numero Dex, dtype: int64
```

Siguiente a esto se realizo una clasificación del rol que puede tener un pokemon en el competitivo los cuales seria: Atacante Fisico, Atacante Especial, Tanque Fisico o Especial, Veloz, Muralla en el que cada uno tiene una forma determinada de decidir a que rol permanece

Para atacante físico:

```
# Ahora vamos a realizar una categorizacion de el rol que ocupa cada pokemon

# Atacante Físico
cond_atk = (Pokedex_Base['Ataque'] > Pokedex_Base['Ataque Especial']) & \
           (Pokedex_Base['Ataque'] > Pokedex_Base['Defensa']) & \
           (Pokedex_Base['Ataque'] > Pokedex_Base['Defensa Especial'])
```

Para atacante Especial

```
# Atacante Especial
cond_sp_atk = (Pokedex_Base['Ataque Especial'] > Pokedex_Base['Ataque']) & \
              (Pokedex_Base['Ataque Especial'] > Pokedex_Base['Defensa']) & \
              (Pokedex_Base['Ataque Especial'] > Pokedex_Base['Defensa Especial'])
```

Para tanque:

```
# Tanque (Físico o Especial)
cond_tank_fis = (Pokedex_Base['Defensa'] >= 100) & (Pokedex_Base['Defensa'] > Pokedex_Base['Defensa Especial'])
cond_tank_esp = (Pokedex_Base['Defensa Especial'] >= 100) & (Pokedex_Base['Defensa Especial'] > Pokedex_Base['Defensa'])
```

Para veloz es un calculo especial ya que para determinar si entra dentro de esta categoría debemos promediar la totalidad de las estadísticas, luego comparar si la velocidad es mayor o igual a 100 y si es mayor a el promedio de las estadísticas de esta manera se puede determinar si entra o no en este grupo.

```
# Veloz (*Sweeper*)
# Calculamos el promedio de stats para comparar la velocidad
Pokedex_Base['Avg_Stats'] = Pokedex_Base[['Puntos Salud', 'Ataque', 'Defensa', 'Ataque Especial', 'Defensa Especial', 'Velocidad']].mean(axis=1)
cond_speed = (Pokedex_Base['Velocidad'] >= 100) & (Pokedex_Base['Velocidad'] > Pokedex_Base['Avg_Stats'])
```

Para muralla en diferencia del tanque este se centra en los puntos de salud del Pokémon

```
# 5. Muralla (Alto HP y defensa)
cond_wall = (Pokedex_Base['Puntos Salud'] >= 100) & (Pokedex_Base['Defensa'] + Pokedex_Base['Defensa Especial'] >= 200)
```

Por ultimo por descarte los Pokémon que no cumplen con ninguna de estas categorías se van a ubicar en la categoría de balanceado

```
[14]: # Usamos np.select para aplicar la lógica. 'Balanceado' es el valor por defecto (default).
Pokedex_Base['Rol Pokemon'] = np.select(conditions, choices, default='Balanceado')
```

Para finalizar asignamos el nombre que va llevar en el dataframe cada una de estas categorías.

```
# Definimos las opciones de rol en orden de prioridad
conditions = [cond_wall, cond_tank_fis, cond_tank_esp, cond_speed, cond_atk, cond_sp_atk]
choices = ['Muralla', 'Tanque Físico', 'Tanque Especial', 'Veloz', 'Atacante Físico', 'Atacante Especial']
```

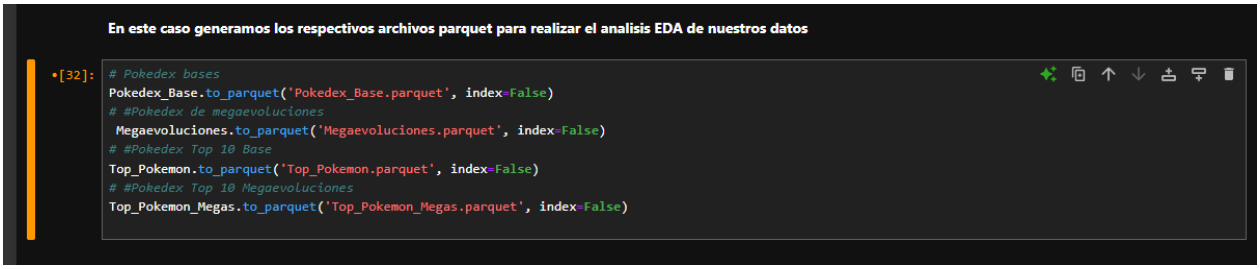
Ahora se genera un dataframe nuevo que nos permita ver el top 10 pokemon con mayor cantidad de estadísticas totales, pero excluyendo los legendarios. Por lo cual se ordena por campo estadísticas totales y luego se toman los primeros 10 registros

```
[19]: # Ahora se va a generar un dataframe con los 10 pokemon con mejores estadisticas
# primero vamos a excluir los legendarios de este dataframe de pokemon base por lo cual generamos un filtro

Filtro_legendarios = Pokedex_Base['Legendario'] == False
# Generamos un nuevo dataframe que contenga los pokemon sin legendarios
Pokedex_Sin_Legendarios = Pokedex_Base[Filtro_legendarios]

# Procedemos a realizar el ordenamiento por la columna estadisticas totales de mayor a menor e indicamos que solo nos muestre los primeros 10
Top_Pokemon = Pokedex_Sin_Legendarios.sort_values(by='Estadisticas Totales', ascending=False).iloc[0:10]
Top_Pokemon
```

Aclaración importante:

Este mismo procedimiento realizado con realizado con los Pokémon Base se va a realizar con el dataframe megaevoluciones por lo cual no se considera relevante anexarlo en este documento ya que es prácticamente el mismo código exceptuando el dataframe que se esta utilizando.

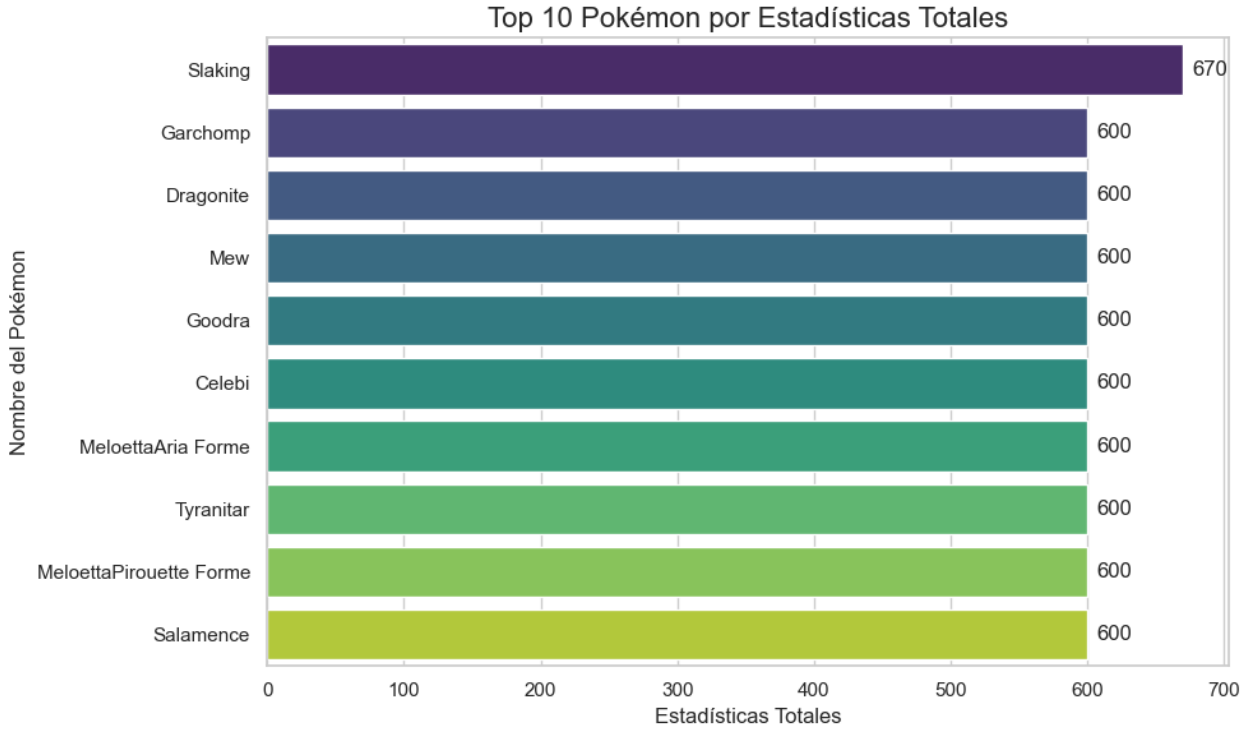
Por ultimo para continuar con el análisis EDA se genera los dataset obtenidos luego de realizar el proceso de análisis de datos que serán útiles para realizar los gráficos estadísticos iniciales.



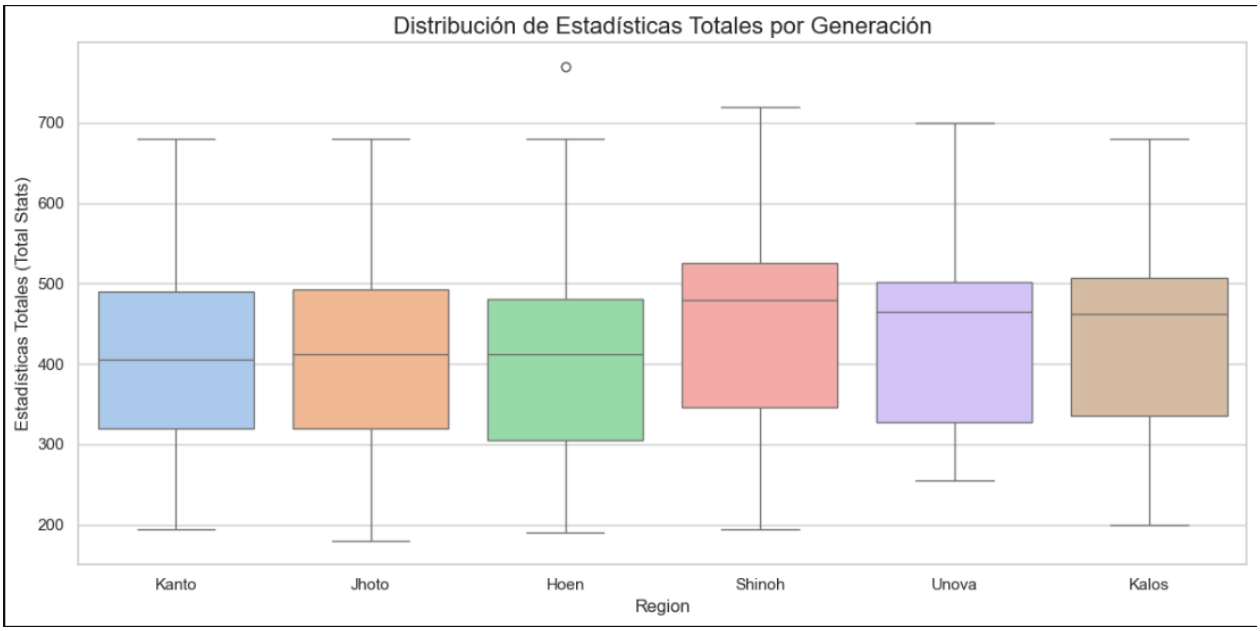
Análisis de Balance de Poder y Diseño del Juego:

Para realizar un análisis de los 10 Pokémon más fuertes de acuerdo con sus estadísticas totales se utilizó un gráfico de barras horizontal el cual permita comparar de acuerdo con su posición y tamaño en que orden se puede ubicar los 10 Pokémon.

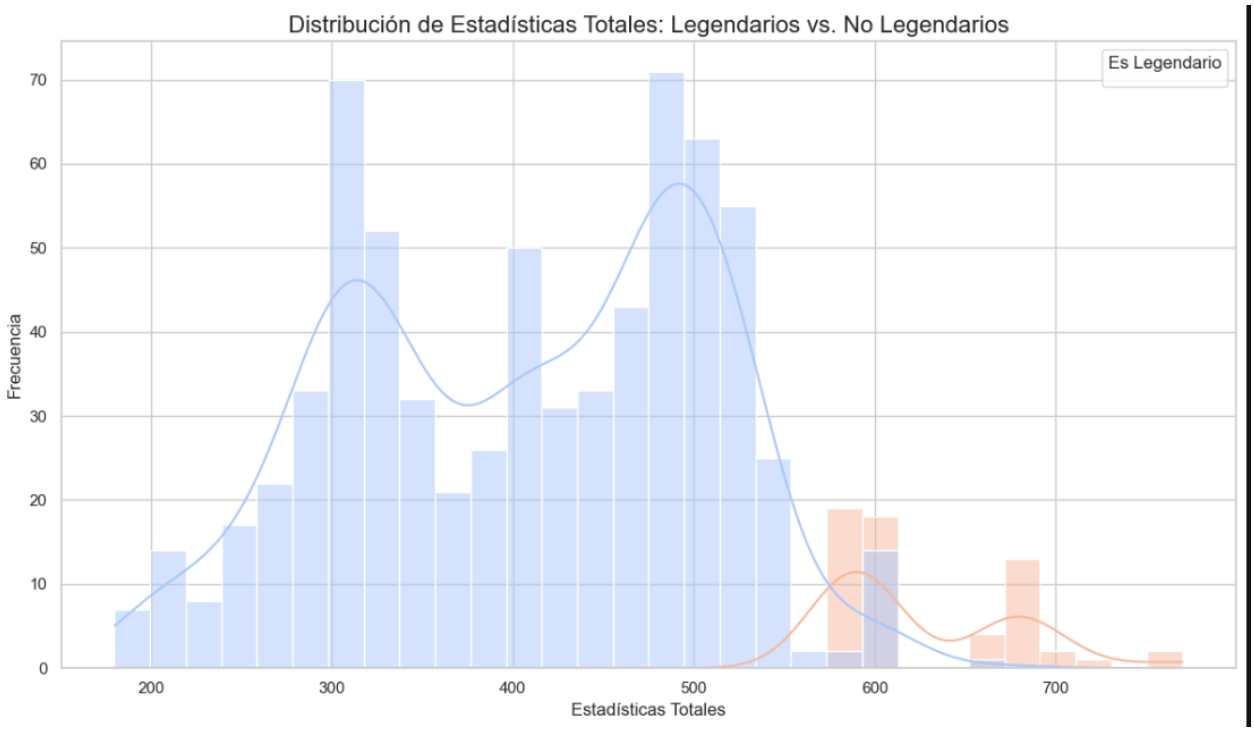
Se puede concluir lo siguiente: El gráfico de barras revela una uniformidad de poder en el Top 10, con la gran mayoría de criaturas agrupadas en 600 Estadísticas Totales. Esta agrupación en el umbral de Poder de Élite indica una regla de diseño estandarizada para Pseudos Legendarios y Míticos. El único valor atípico significativo es Slaking (670), cuya estadística bruta es mitigada por su habilidad, demostrando que el poder bruto (Total estadisticas) no es el único factor determinante. La capacidad de combate entre estos Pokémon de alto nivel se define entonces por variables secundarias pero cruciales, como el Tipo Primario y la distribución de las estadísticas individuales.



Para realizar el análisis del promedio de estadísticas a través de las regiones utilizamos el grafico de cajas el cual permite por medio de la mediana determinar qué región presenta el índice de estadísticas más fuerte. El análisis de la distribución de las Estadísticas Totales (Total estadísticas) a través del diagrama de cajas validó dos tendencias de diseño críticas: primero, se confirmó un **sesgo al alza en el poder promedio** (mediana) de los Pokémon, con **Sinnoh** (4ª Generación) estableciendo la mediana más alta, lo que indica un **incremento progresivo en el poder base** del juego a lo largo del tiempo. Segundo, la existencia de un **valor atípico extremo** en **Hoenn** (el punto sobre 750), generado por las Formas Primigenias, demuestra la necesidad analítica de **segmentar los datos** para diferenciar el poder base del poder potenciado. En conjunto, el gráfico muestra que, si bien el rango de poder total se mantiene amplio en todas las regiones, las nuevas generaciones elevan constantemente el **umbral de poder general** de la base de Pokémon.



El Histograma Superpuesto (Distribución Legendarios vs. No Legendarios) valida que, si bien los Pokémon normales presentan una mayor variabilidad de estadísticas (una curva más amplia), los Legendarios tienen un poder altamente estandarizado y concentrado en el extremo superior del espectro (la curva más estrecha y desplazada a la derecha). Esta distinción de poder se complementa con la tendencia de diseño observada en el Gráfico de Cajas, donde la mediana de Sinnoh demuestra un incremento progresivo en el poder general de las nuevas generaciones. Finalmente, el Gráfico de Dispersión confirma la regla de compensación en el diseño de roles, revelando una correlación positiva general, pero con los Legendarios ocupando las posiciones de mayor poder concentrado en el cuadrante de Ataque y Defensa altos.

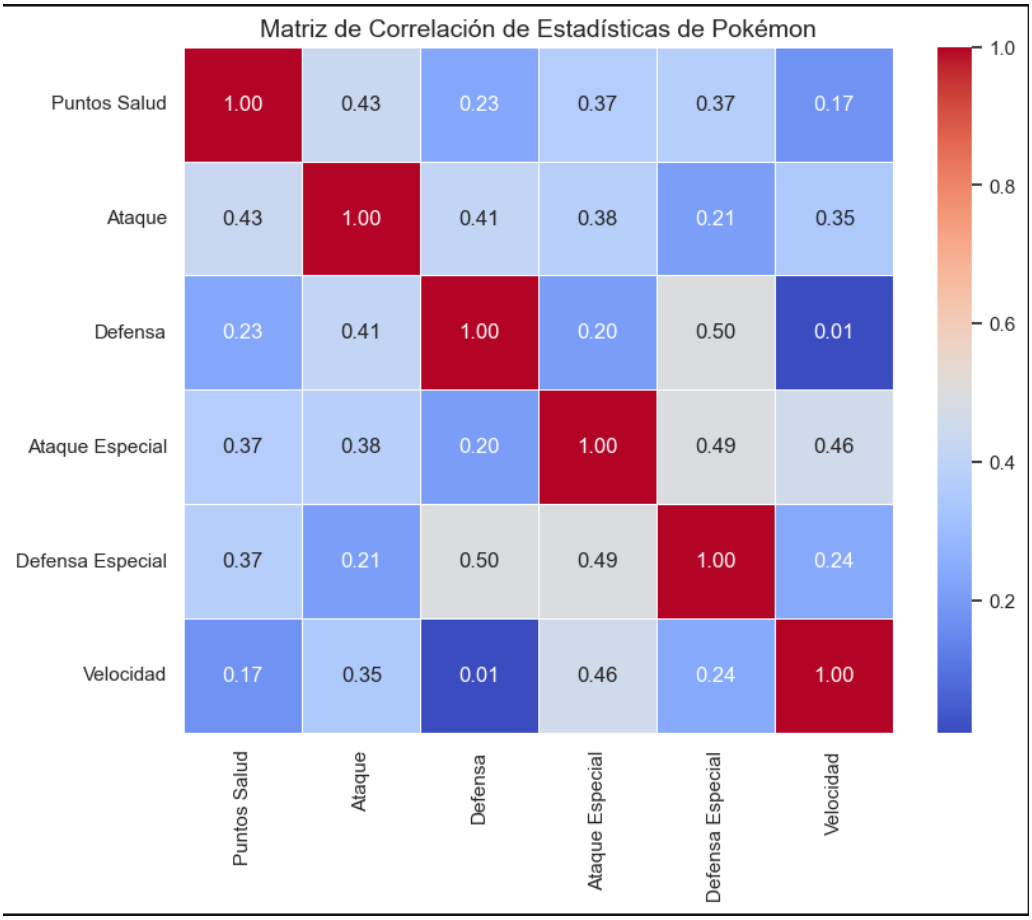


El Mapa de Calor de Correlación reveló que, en general, las estadísticas presentan una correlación positiva, lo cual es esperable debido al factor de poder total del Pokémon (es decir, los Pokémon más fuertes tienen todas las estadísticas más altas). Sin embargo, identificamos una correlación negativa sutil pero significativa (cercana a) entre la Velocidad y las estadísticas defensivas (Defense / HP), lo que confirma la estrategia de balance de juego intencional que compensa la rapidez con la fragilidad, y viceversa.

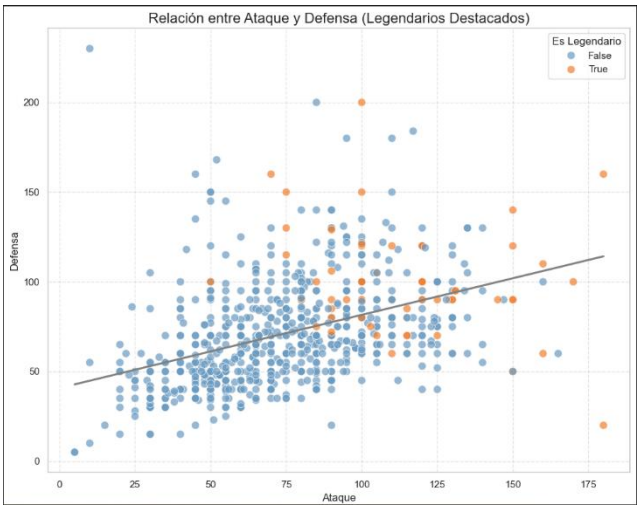
Matriz de Correlación de Pearson:

	Puntos Salud	Ataque	Defensa	Ataque Especial	\
Puntos Salud	1.00	0.43	0.23	0.37	
Ataque	0.43	1.00	0.41	0.38	
Defensa	0.23	0.41	1.00	0.20	
Ataque Especial	0.37	0.38	0.20	1.00	
Defensa Especial	0.37	0.21	0.50	0.49	
Velocidad	0.17	0.35	0.01	0.46	

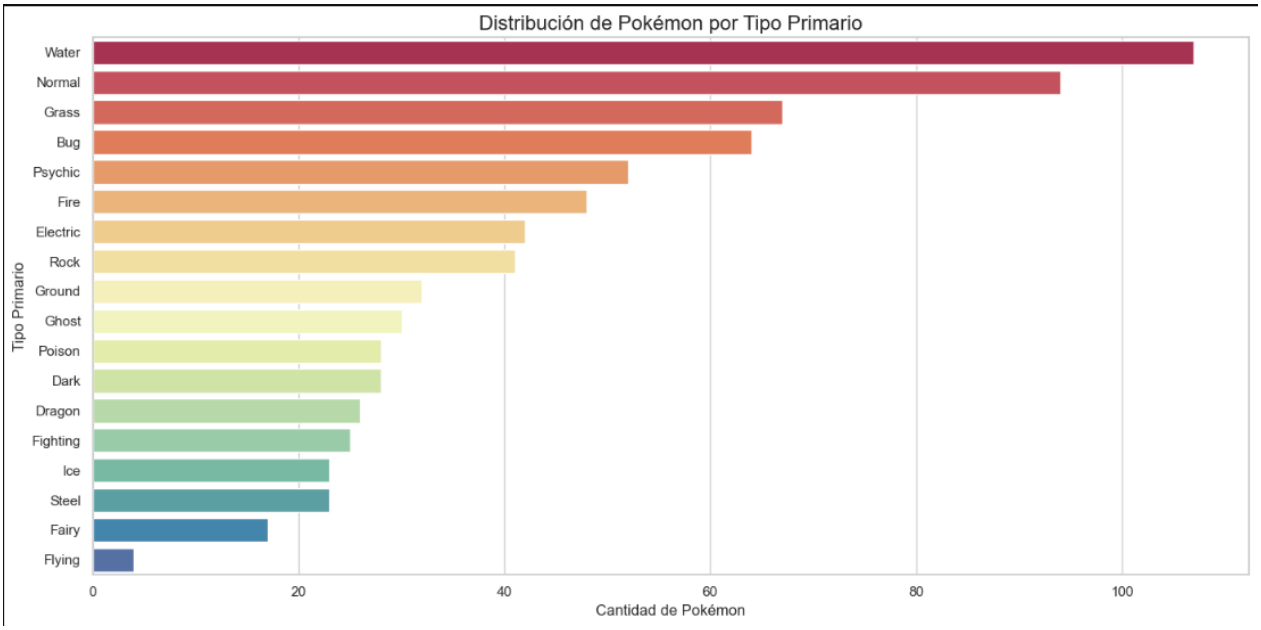
	Defensa Especial	Velocidad
Puntos Salud	0.37	0.17
Ataque	0.21	0.35
Defensa	0.50	0.01
Ataque Especial	0.49	0.46
Defensa Especial	1.00	0.24
Velocidad	0.24	1.00



Este gráfico valida nuestra conclusión de correlación positiva, mostrando que el poder es interdependiente. Sin embargo, su mayor valor es la segmentación: la nube naranja de los Legendarios está desplazada hacia el cuadrante de alto poder, lo que verifica estadísticamente su estatus de élite. La variabilidad de los Legendarios en el cuadrante superior (ej. un Legendario con Ataque 150 y Defensa 50 vs. otro con Ataque 100 y Defensa 150) sugiere que, si bien son poderosos, conservan roles de combate bien definidos y compensatorios.



Esta desigualdad de frecuencia indica una decisión de diseño deliberada que afecta directamente el metajuego competitivo. Aunque el poder individual (Total estadísticas) no se ve comprometido (ya que los tipos raros como Dragón o Hielo suelen tener estadísticas altas), el predominio de tipos como Agua obliga a los jugadores a priorizar la cobertura de tipos en sus equipos, haciendo que los Pokémon de tipo Planta y Eléctrico sean defensivamente cruciales. El juego está desequilibrado en cantidad, lo que define un factor clave en la estrategia competitiva.



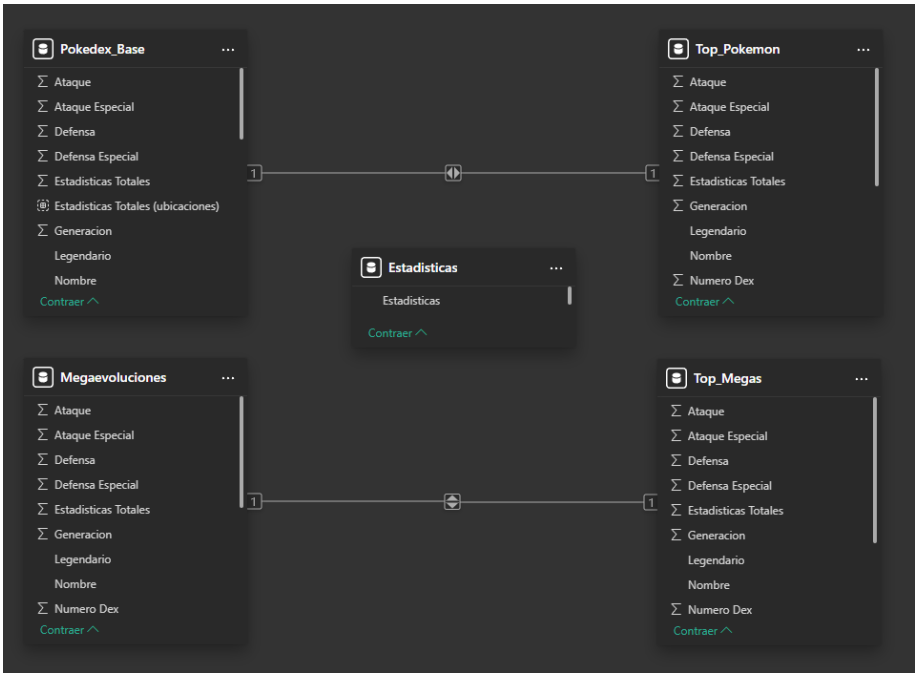
ANALISIS POWER BI

Relación de las tablas en Power Bi:

Para el diseño de los gráficos y el análisis de Power Bi se utilizaron los siguientes recursos generados:

- Pokedex_Base
- Megaevoluciones
- Top_Pokemon
- Top_Megaevoluciones

En este caso solo se realizó una relación entre los nombres de los top y las respectivas tablas de Pokémon ya que no considere necesario relacionarlas de otra forma por que ambas tienen los mismos campos. Por ultimo se creo una tabla auxiliar estadísticas para poder realizar la ejecución de medidas correspondientes a las estadísticas.



Elaboración de visualizaciones:

Para la elaboración de las visualizaciones dentro del reporte en Power BI utilicé diferentes tipos de gráficos y objetos visuales, seleccionados según la naturaleza de cada análisis. En general, se emplearon los siguientes:

- Tarjetas (KPIs): Para mostrar valores puntuales y métricas clave de forma directa y visualmente destacada.
- Gráfico de columnas agrupadas: Utilizado para analizar la distribución de la población de Pokémon y comparar cantidades entre categorías específicas.
- Gráfico de dispersión: Empleado para validar la correlación entre las variables estadísticas base y observar patrones o relaciones relevantes.
- Gráfico de barras agrupadas: Ideal para comparar valores entre diferentes grupos, especialmente al analizar los Pokémon más fuertes o las mega evoluciones.
- Gráfico de líneas: Aunque el dataset no incluye variables temporales, este gráfico se utilizó para observar el comportamiento o la variación de una estadística en función de categorías ordenadas.
- Segmentadores: Incorporados para permitir filtros dinámicos en el reporte, facilitando una exploración más flexible según tipo, generación u otras características.
- Tablas: Incluidas para mostrar de forma detallada la información comparada y permitir una lectura exacta de los valores.
- En conjunto, estas visualizaciones permiten analizar de manera clara la composición del dataset, las diferencias entre generaciones y la relación entre las estadísticas base de los Pokémon.

Medidas

Para la construcción de las visualizaciones en Power BI fue necesario crear un conjunto de **14 medidas en DAX**, con el fin de obtener cálculos más precisos y permitir un análisis dinámico y detallado de las estadísticas base de los Pokémon. Estas medidas se utilizaron especialmente para KPIs, comparaciones entre roles, análisis de promedios y visualizaciones interactivas.

A continuación, se describen las medidas creadas y su propósito dentro del análisis:

- Cantidad Pokémon: Calcula el número total de Pokémon incluidos en el dataset.
- Media Ataque Especial Rol: Obtiene el promedio de la estadística Ataque Especial para el grupo filtrado.
- Media Ataque Rol: Calcula el promedio de la estadística Ataque según el contexto seleccionado.
- Media Defensa Especial Rol: Determina el promedio de la estadística Defensa Especial para el grupo activo.
- Media Defensa Rol: Calcula el promedio de Defensa para el rol o categoría visualizada.
- Media HP Rol: Obtiene el promedio de HP (puntos de salud) dentro del filtro aplicado.
- Media Velocidad Rol: Calcula el promedio de Velocidad para el conjunto seleccionado.
- Megaevoluciones: Indica la cantidad total de megaevoluciones presentes en el dataset.
- Pokémon por Rol: Devuelve el número de Pokémon asociados a cada rol o categoría definida.
- Porcentaje Rol: Calcula el porcentaje que representa cada rol frente al total de Pokémon.
- Promedio Estadísticas Base: Obtiene el promedio general de las estadísticas base de los Pokémon seleccionados.
- Rol Más Común: Identifica el rol predominante dentro del grupo analizado.
- Total Roles: Muestra cuántos roles diferentes existen en el dataset.
- Valor Medio Dinámico: Calcula el promedio agrupado de todas las estadísticas según el filtro o segmentación aplicada.

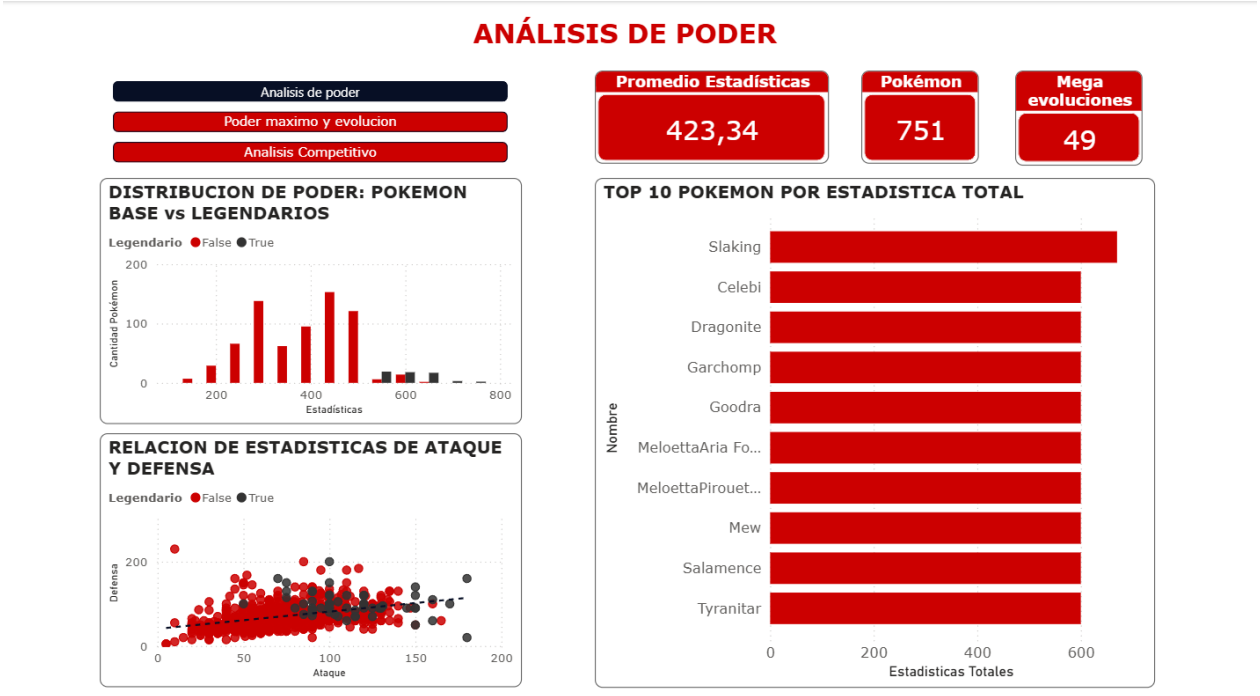
Estas medidas permiten un análisis más flexible y completo, facilitando la comparación entre grupos y la construcción de visualizaciones que responden a las preguntas planteadas en este proyecto.

Análisis de poder:

El análisis de la Distribución de Poder confirma que el promedio general de las estadísticas base se sitúa en 423.34, con la mayoría de los Pokémon base concentrados en el rango de 400 a 450 puntos. Por otro lado, la Distribución de Legendarios (True) muestra un claro sesgo hacia la alta potencia, concentrando el poder en el rango superior de 580 a 770.

El TOP 10 POKÉMON resalta las principales anomalías de poder en la base de datos, identificando a Pokémon no legendarios (pseudo-legendarios como Slaking, Dragonite y Tyranitar) que poseen un índice de poder total comparable o superior al de muchos Pokémon Legendarios.

Finalmente, el gráfico de dispersión de Ataque vs. Defensa sugiere una correlación débil o nula entre ambas estadísticas. Sin embargo, la dispersión evidencia el principio de especialización de rol: los Pokémon que alcanzan valores muy altos en una estadística (ej. Ataque) suelen mostrar un compromiso o *trade-off* con valores inferiores en la otra (Defensa), y viceversa.



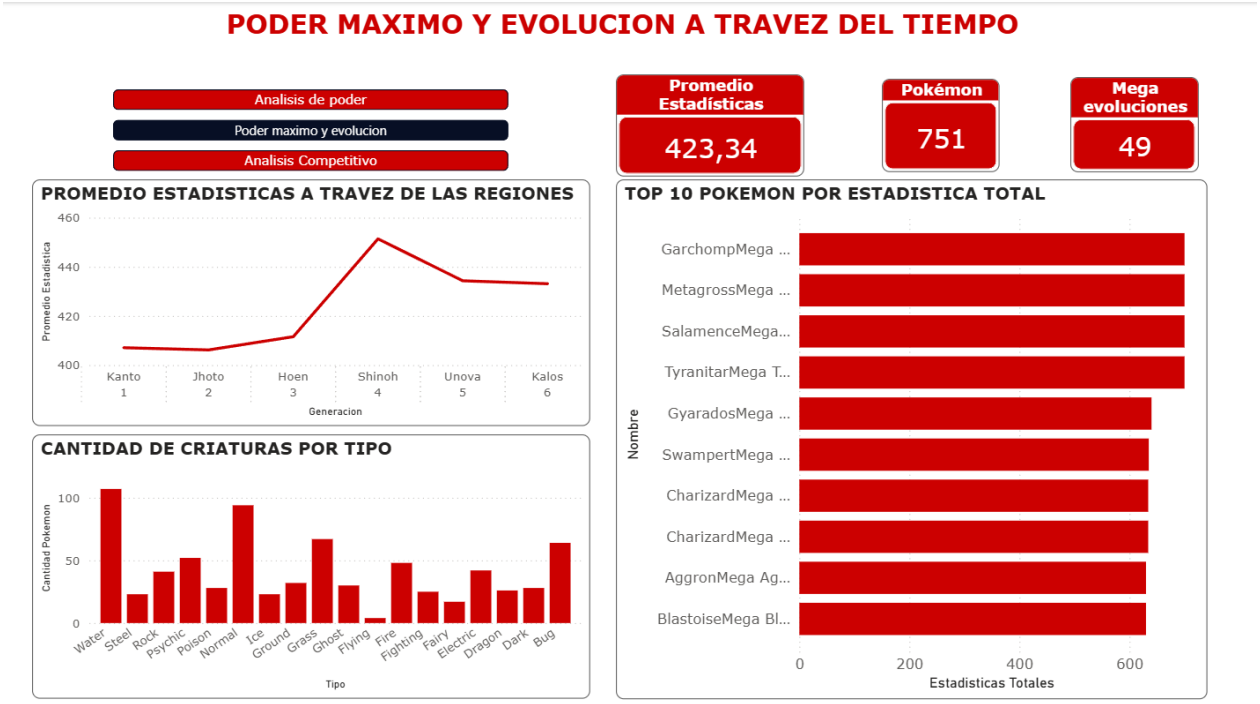
Poder máximo y megaevolucion

El análisis del Promedio de Poder por Generación revela un patrón de inflación de estadísticas a lo largo del tiempo. El poder base promedio se mantuvo relativamente bajo y estable entre Kanto (Gen 1) y Johto (Gen 2), antes de iniciar un aumento constante a partir de Hoenn (Gen 3).

El pico máximo de poder promedio se alcanzó en Sinnoh (Gen 4), lo que indica un cambio significativo en el diseño de las criaturas introducidas en esa generación. Finalmente, se observa una estabilización de la escala de poder en las regiones posteriores (Unova y Kalos).

El TOP 10 de Megaevoluciones confirma que esta mecánica de evolución temporal introduce un nivel de poder extremo, con muchos Pokémon alcanzando o superando la barrera de las 600 Estadísticas Totales, lo que los hace directamente comparables con los Pokémon Legendarios más poderosos.

Por último, la Distribución de Frecuencia por Tipo muestra una disparidad significativa en la cantidad de criaturas disponibles por tipo. El tipo Agua es el más abundante en el juego, mientras que otros tipos clave (como Hada o Hielo) tienen una presencia mucho menor, lo cual afecta inherentemente la diversidad y el balance de estrategias en el juego competitivo.

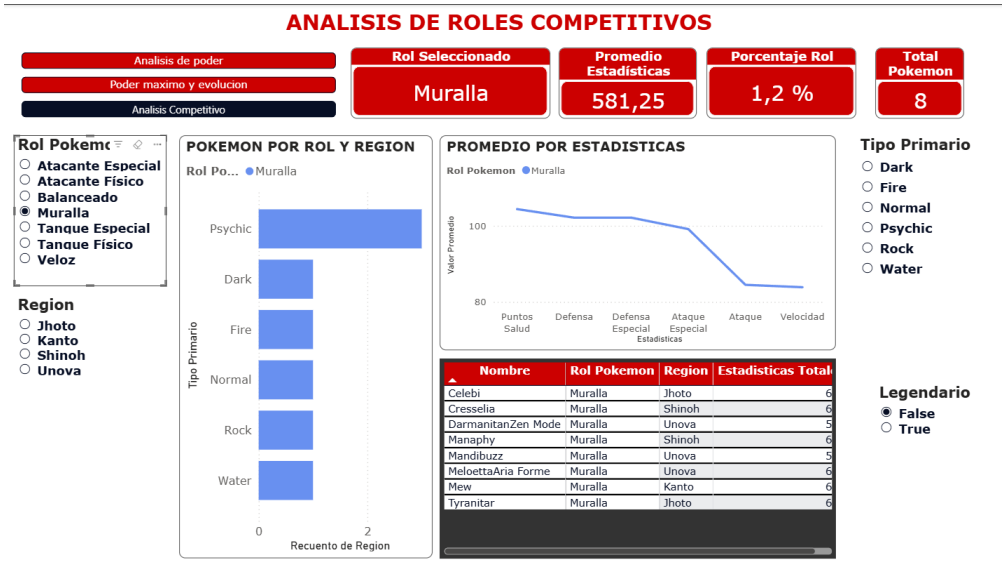


Análisis de roles competitivos

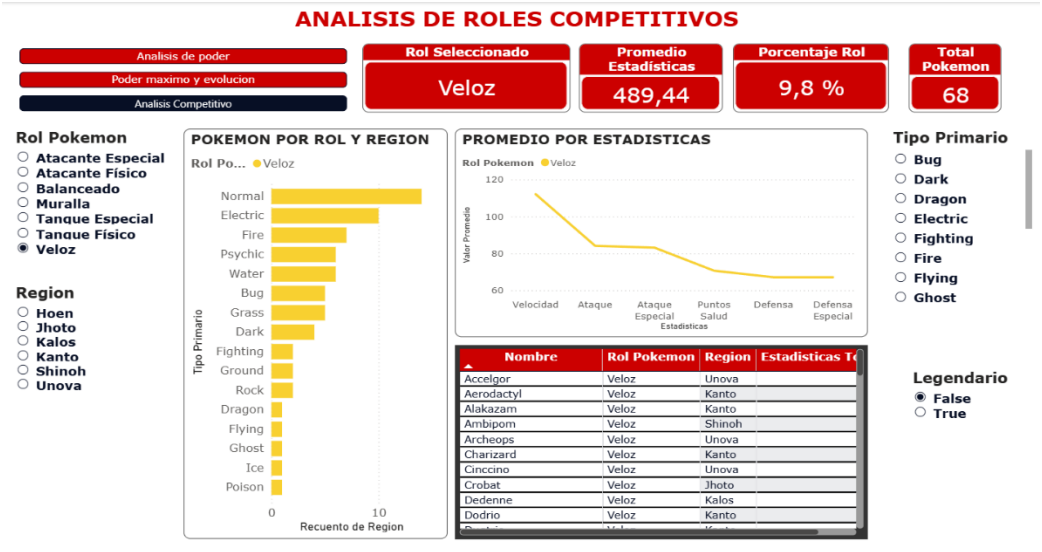
La página de Análisis Competitivo tiene como objetivo principal validar la Ingeniería de Características aplicada, demostrando cómo la clasificación de Rol Pokémon aísla perfiles de estadísticas distintos. Esto se logra mediante la segmentación dinámica por Rol, Región, Tipo y Legendario.

Tomando como ejemplo el rol 'Muralla' (definido por una alta capacidad defensiva y HP), el Gráfico de Perfil (Promedio por Estadísticas) valida la clasificación. El perfil de 'Muralla' muestra un pico en las estadísticas defensivas (Puntos Salud y Defensa), seguido de una compensación directa (*trade-off*) con una caída significativa en las estadísticas ofensivas (Ataque y Velocidad).

Esta relación inversa entre las capacidades defensivas y ofensivas no solo confirma la efectividad de la clasificación, sino que también subraya el principio de especialización de rol que rige el diseño de las criaturas en el juego.



Por otra parte, al tener Pokémon veloces se puede evidenciar que a mayor numero de velocidad el Pokémon deja de ser tan resistente y sus características de ataque suben en su mayoría o se mantienen estables.



CONCLUSIONES

El análisis del ecosistema competitivo de Pokémon demuestra que el balance estratégico se basa en la especialización y no en el mero equilibrio estadístico. La clasificación en Roles Competitivos es fundamental, ya que valida el principio de compensación de estadísticas (trade-off), dotando de una profundidad estratégica única a cada criatura. A esto se suma la disparidad en la distribución de Tipos, lo que obliga a la creación de estrategias altamente especializadas para contrarrestar los tipos más comunes.

Históricamente, se observa una clara inflación de estadísticas y una escalada de poder que alcanzó su pico máximo en Sinnoh (Gen 4), con el poder base promedio estabilizándose en las generaciones posteriores.

Finalmente, la introducción de las Megaevoluciones (Gen 6) redefinió la escala de poder del competitivo al introducir criaturas con estadísticas comparables a las de los Pokémon Legendarios. Este cambio no solo generó nuevas dinámicas estratégicas, sino que también elevó la importancia de la gestión de Tipos y Habilidades durante la batalla.