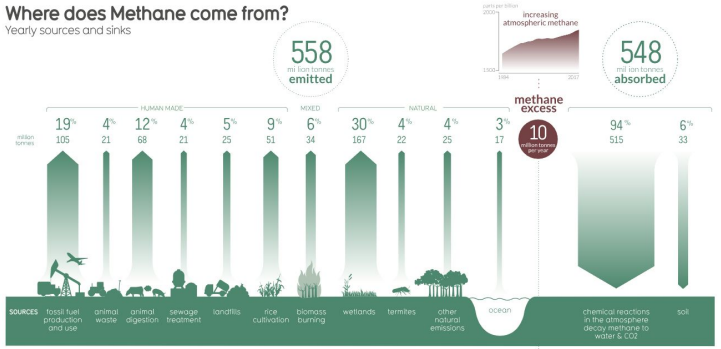
Visualización de datos 01

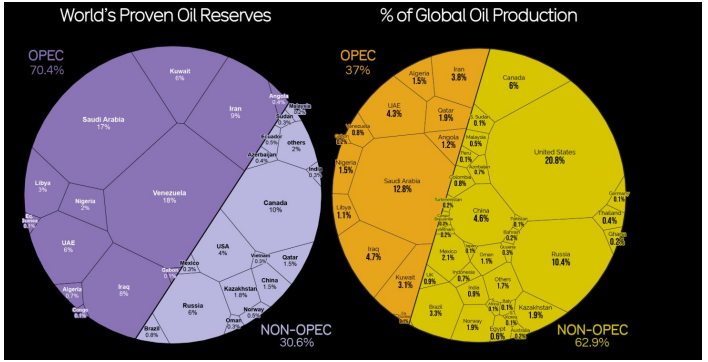
Ainoa Iglesias Dasilva
alu0101164403@ull.edu.es
Análisis de Datos Masivos, 2025

Resumen lectura y Plataformas de visualización

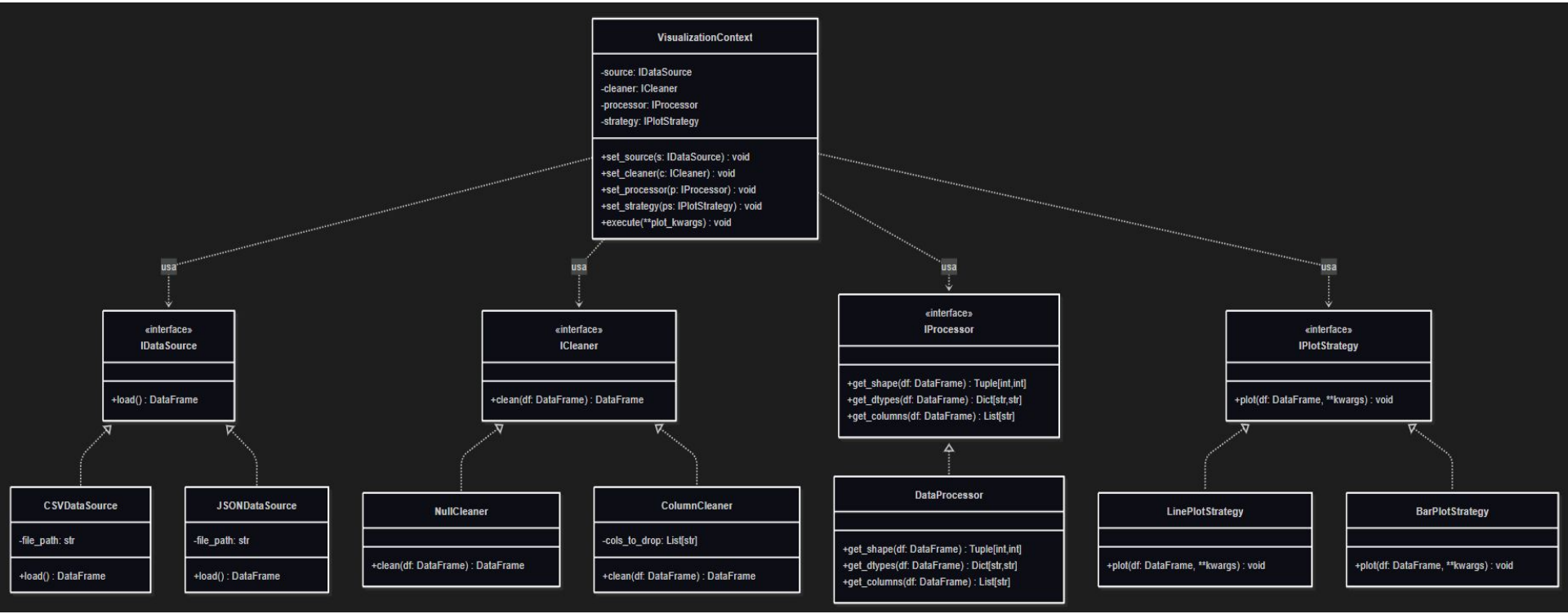
- La visualización de datos facilita la compresión de la información.
- Las imágenes permiten identificar patrones y relaciones más fácil que los textos y tablas.
- Los científicos no dedican tiempo a esta parte, y generan visualizaciones erróneas y engañosas.
- Según estudios, hay gráficas más intuitivas que otras.
- Una buena visualización se basa en saber escoger la gráfica según lo que se quiere mostrar.



- **Tableau Public:** visualizaciones de datos curiosos de cualquier tipo.
- **World Bank Open Data:** amplia colección de datos sobre desarrollo global.
- **Our World in Data:** visualizaciones interactivas sobre una variedad de temas relacionados con el bienestar global.
- **Information is beautiful:** presenta proyectos personalizados con un diseño brillante y minimalista.

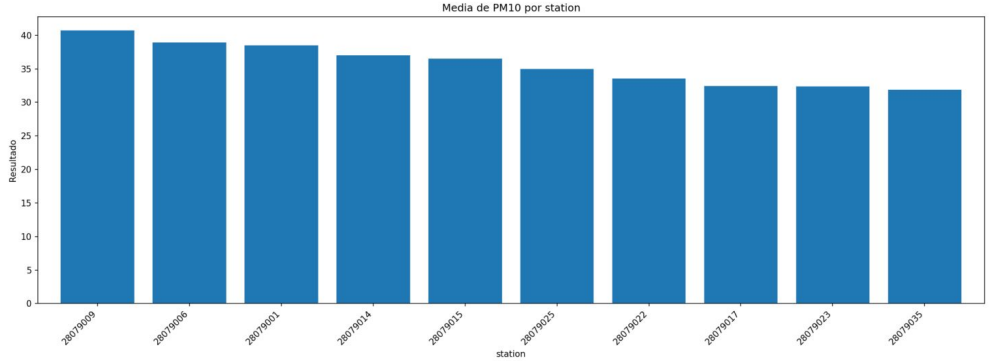


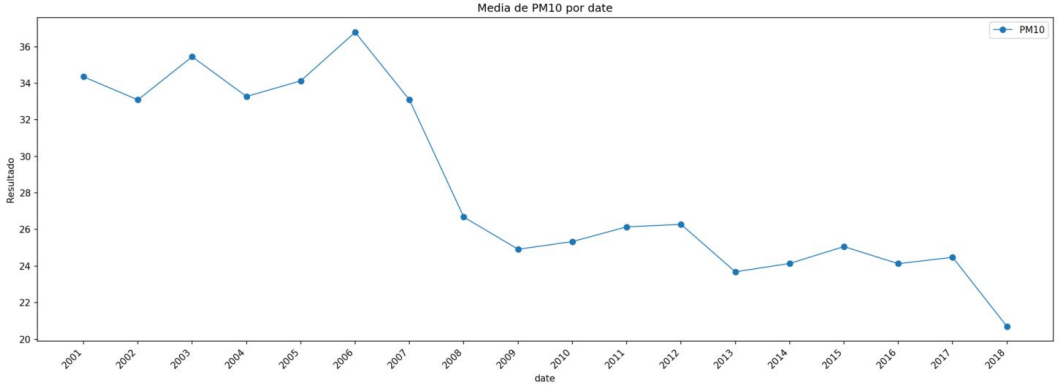
Diagrama


VisualizationContext
-source: IDataSource
-cleaner: ICleaner
-processor: IProcessor
-strategy: IPlotStrategy
+set_source(s: IDataSource) : void
+set_cleaner(c: ICleaner) : void
+set_processor(p: IProcessor) : void
+set_strategy(ps: IPlotStrategy) : void
+execute(**plot_kwargs) : void
usa
usa
usa
usa
«interface»
IDataSource
+load() : DataFrame
«interface»
ICleaner
+clean(df: DataFrame) : DataFrame
«interface»
IProcessor
+get_shape(df: DataFrame) : Tuple[int,int]
+get_dtypes(df: DataFrame) : Dict[str,str]
+get_columns(df: DataFrame) : List[str]
«interface»
IPlotStrategy
+plot(df: DataFrame, **kwargs) : void
CSVDataSource
-file_path: str
+load() : DataFrame
JSONDataSource
-file_path: str
+load() : DataFrame
NullCleaner
+clean(df: DataFrame) : DataFrame
ColumnCleaner
-cols_to_drop: List[str]
+clean(df: DataFrame) : DataFrame
DataProcessor
+get_shape(df: DataFrame) : Tuple[int,int]
+get_dtypes(df: DataFrame) : Dict[str,str]
+get_columns(df: DataFrame) : List[str]
LinePlotStrategy
+plot(df: DataFrame, **kwargs) : void
BarPlotStrategy
+plot(df: DataFrame, **kwargs) : void

Herramientas y visualizaciones

- Python 3
- Pandas: para carga y manipulación de datos.
- Matplotlib: para visualizaciones.
- abc(abstract base classes): definición de interfaces
- Jupyter Notebook: entorno presentación.
- Patrón de diseño
- Mermaid para diagrama





https://github.com/AinoaIglesias/ADM_24-25

Visualización de datos 02

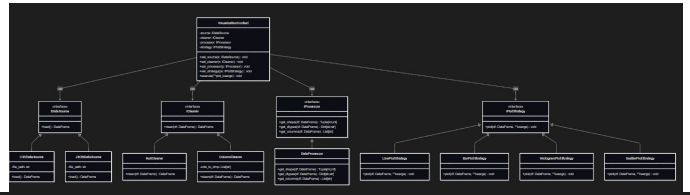
Ainoa Iglesias Dasilva
alu0101164403@ull.edu.es
Análisis de Datos Masivos, 2025

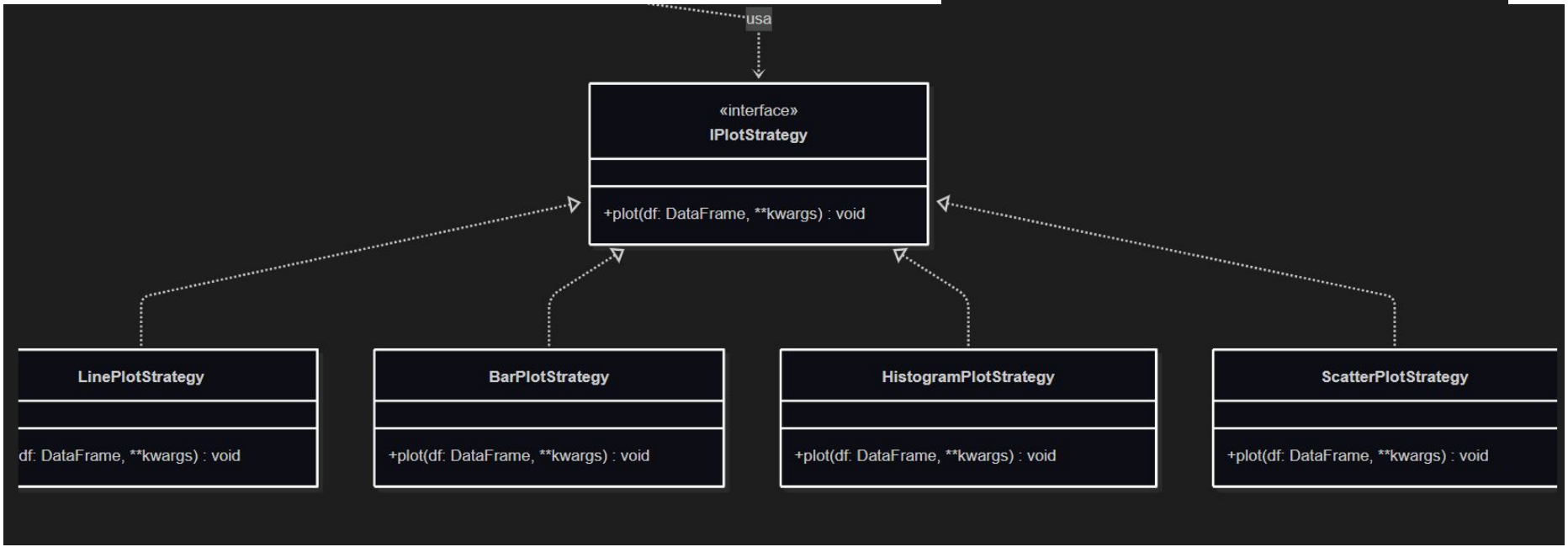
Conclusiones lectura Why scientists need to be better at data visualization

- El uso de escalas de colores es clave.
- Errores comunes:
  - Escala arcoiris
  - Demasiados colores
  - Contraste simultáneo
- Causas que prolongan estos errores:
  - Falta formación
  - Software
  - Revistas, blogs…

- Tener en cuenta:
  - Resaltar solo lo importante
  - Escalas de color como *viridis* o *cividis*

Diagrama

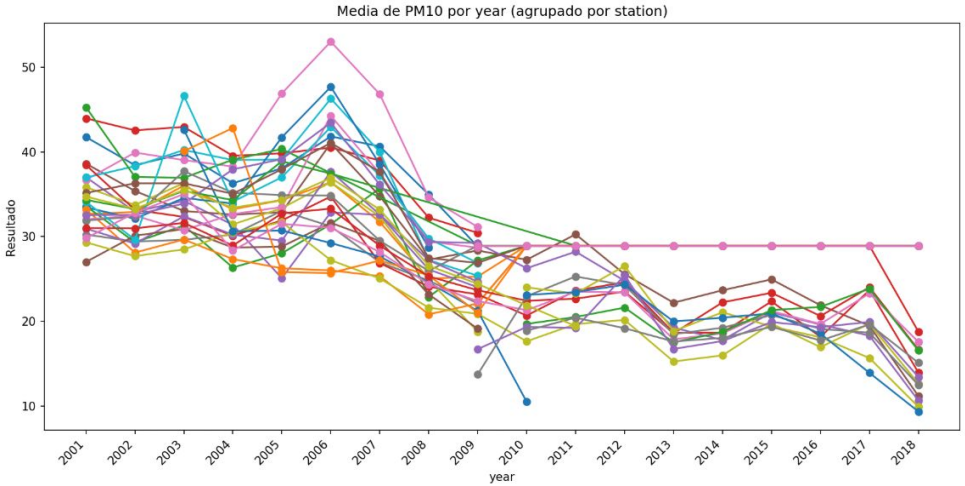


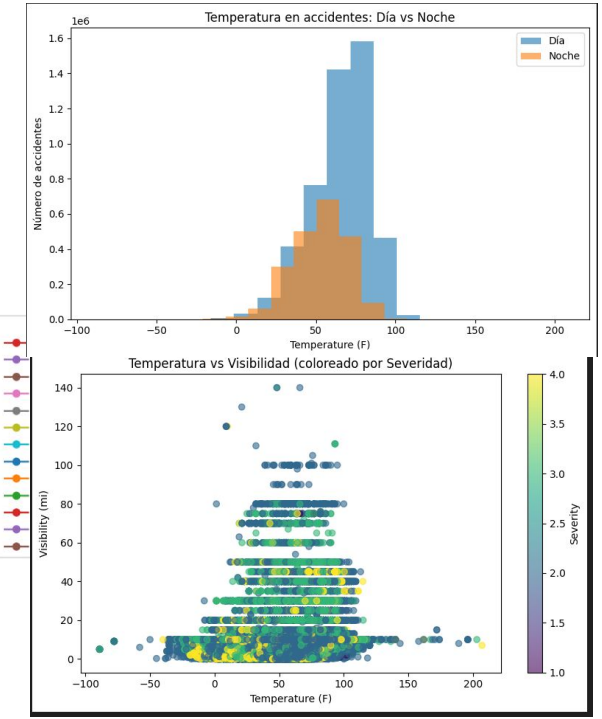

usa
«interface»
IPlotStrategy
+plot(df: DataFrame, **kwargs) : void
LinePlotStrategy
df: DataFrame, **kwargs) : void
BarPlotStrategy
+plot(df: DataFrame, **kwargs) : void
HistogramPlotStrategy
+plot(df: DataFrame, **kwargs) : void
ScatterPlotStrategy
+plot(df: DataFrame, **kwargs) : void

https://github.com/AinoaIglesias/ADM_24-25

Herramientas y visualizaciones

- Python 3
- Pandas: para manipulación de datos.
- Matplotlib / Seaborn / Plotly: para visualizaciones.
- Jupyter Notebook: entorno presentación.
- Patrón de diseño
- ipywidgets

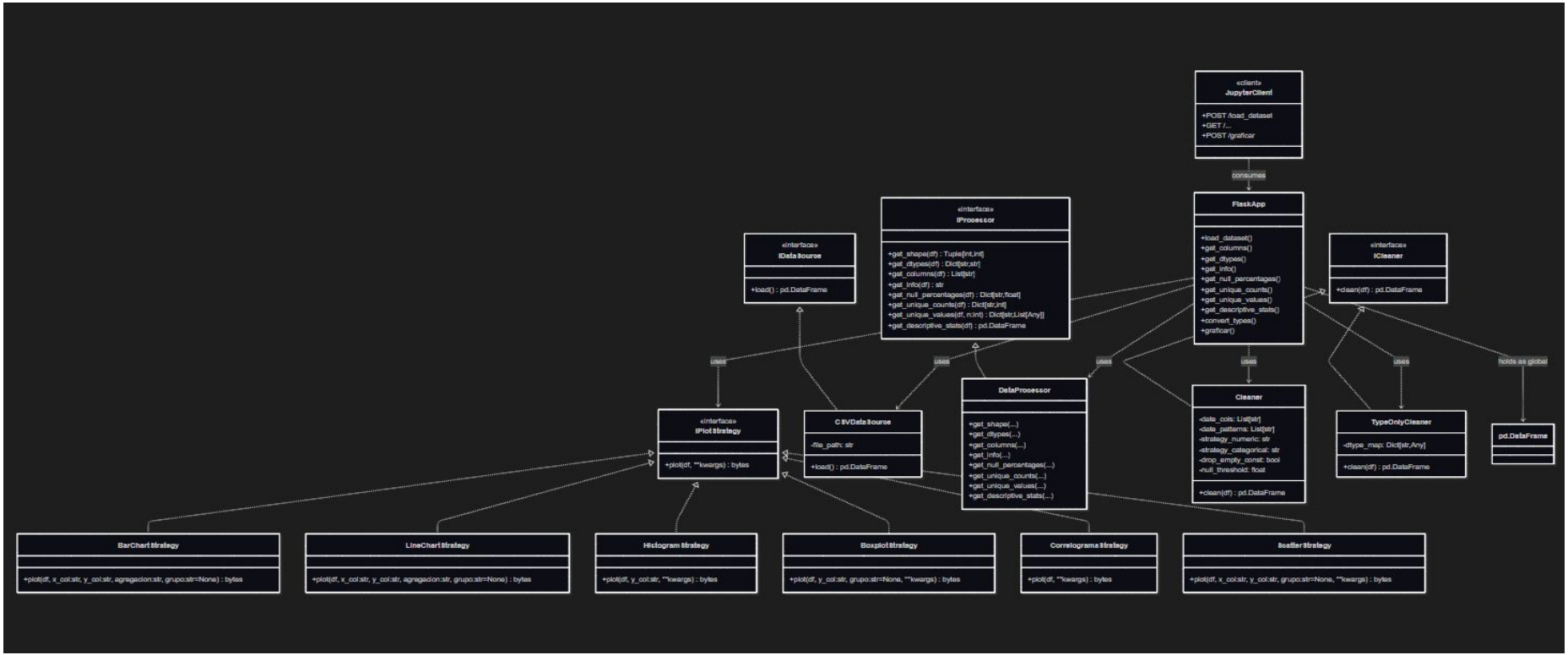




Visualización de datos 03

Ainoa Iglesias Dasilva
alu0101164403@ull.edu.es
Análisis de Datos Masivos, 2025

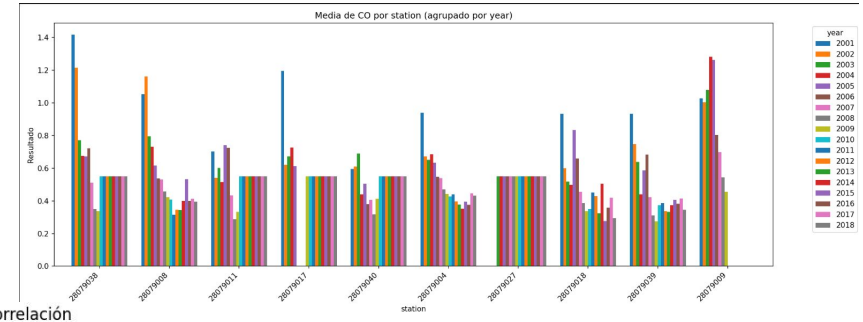
Diagrama

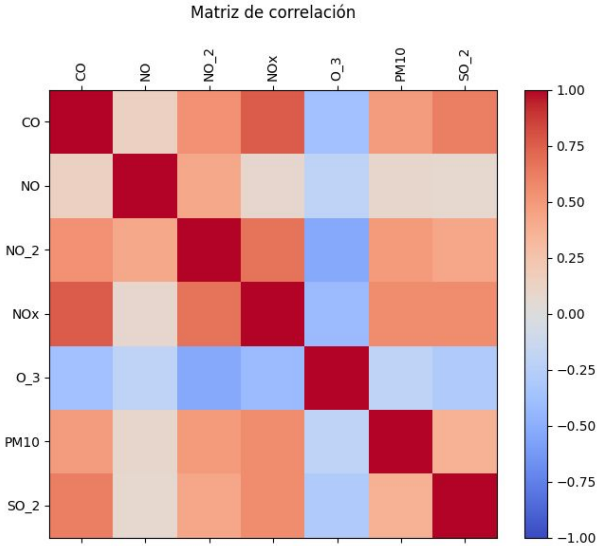

«client»
JupyterClient
+POST /load_dataset
+GET /...
+POST /graficar
consumes
FlaskApp
+load_dataset()
+get_columns()
+get_dtypes()
+get_info()
+get_null_percentages()
+get_unique_counts()
+get_unique_values()
+get_descriptive_stats()
+convert_types()
+graficar()
«interface»
IProcessor
+get_shape(df) : Tuple[int,int]
+get_dtypes(df) : Dict[str,str]
+get_columns(df) : List[str]
+get_info(df) : str
+get_null_percentages(df) : Dict[str,float]
+get_unique_counts(df) : Dict[str,int]
+get_unique_values(df, n:int) : Dict[str,List[Any]]
+get_descriptive_stats(df) : pd.DataFrame
«interface»
IData Source
+load() : pd.DataFrame
«interface»
ICleaner
+clean(df) : pd.DataFrame
uses
holds as global
«interface»
IPlot Strategy
+plot(df, **kwargs) : bytes
CSVData Source
-file_path: str
+load() : pd.DataFrame
DataProcessor
+get_shape(...)
+get_dtypes(...)
+get_columns(...)
+get_info(...)
+get_null_percentages(...)
+get_unique_counts(...)
+get_unique_values(...)
+get_descriptive_stats(...)
Cleaner
-date_cols: List[str]
-date_patterns: List[str]
-strategy_numeric: str
-strategy_categorical: str
-drop_empty_const: bool
-null_threshold: float
+clean(df) : pd.DataFrame
TypeOnlyCleaner
-dtype_map: Dict[str,Any]
+clean(df) : pd.DataFrame
pd.DataFrame
BarChart Strategy
+plot(df, x_col:str, y_col:str, agregacion:str, grupo:str=None) : bytes
LineChart Strategy
+plot(df, x_col:str, y_col:str, agregacion:str, grupo:str=None) : bytes
Histogram Strategy
+plot(df, y_col:str, **kwargs) : bytes
Boxplot Strategy
+plot(df, y_col:str, grupo:str=None, **kwargs) : bytes
Correlograma Strategy
+plot(df, **kwargs) : bytes
Scatter Strategy
+plot(df, x_col:str, y_col:str, grupo:str=None, **kwargs) : bytes

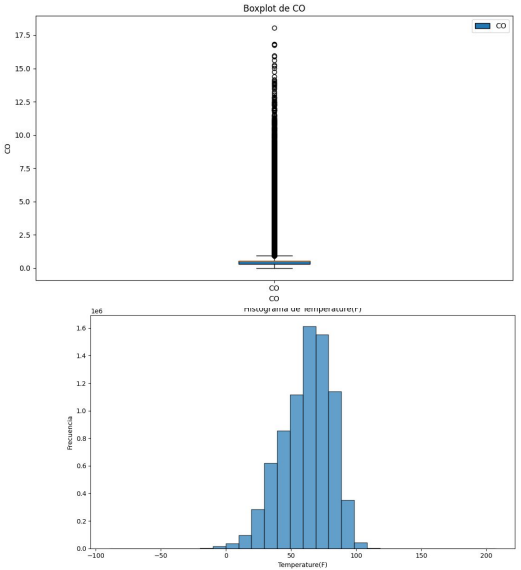
Nuevas visualizaciones

- Histograma
- Correlograma
- Boxplot

- Python 3
- Pandas
- Matplotlib / Seaborn / Plotly
- Jupyter Notebook
- Mermaid
- Flask







Servidor y cliente

- Servidor: Flask
- Cliente 1: Jupyter
- Cliente 2: Streamlit

```python
# IPlotStrategy
strategies = {
    "Barra":         BarChartStrategy(),
    "Línea":         LineChartStrategy(),
    "Histograma":    HistogramStrategy(),
    "Boxplot":       BoxplotStrategy(),
    "Correlograma":  CorrelogramaStrategy(),
    "Scatter":       ScatterStrategy(),
}

@app.route("/graficar", methods=["POST"])
def graficar():
    data = request.get_json() or {}
    tipo = data.get("tipo")
    strat = strategies.get(tipo)
    if strat is None:
        return jsonify(error=f"Tipo '{tipo}' no soportado"), 400

    # Cogemos copia para no modificar el global
    d = df.copy()

    # Parámetros comunes
    grupo    = data.get("grupo")
```

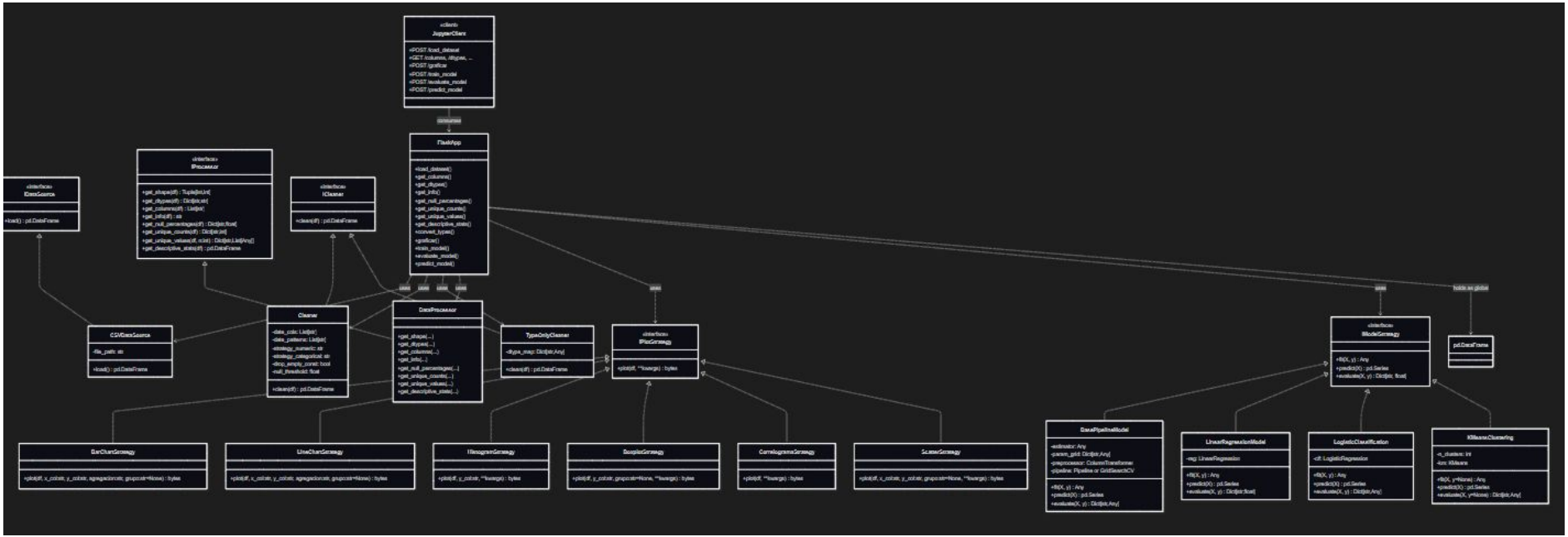
```python
# ------------------- Endpoints ---------------- #

# CARGA DATASET Y LIMPIEZA
@app.route("/load_dataset", methods=["POST"])
def load_dataset():
    """
```

```python
# IProcessor
@app.route("/columns", methods=["GET"])
def get_columns():
    return jsonify(columns=processor.get_columns(df))

@app.route("/dtypes", methods=["GET"])
def get_dtypes():
    return jsonify(dtypes=processor.get_dtypes(df))

@app.route("/info", methods=["GET"])
def get_info():
    return jsonify(info=processor.get_info(df))
```

Visualización de datos 04

Ainoa Iglesias Dasilva
alu0101164403@ull.edu.es
Análisis de Datos Masivos, 2025

Diagrama actualizado



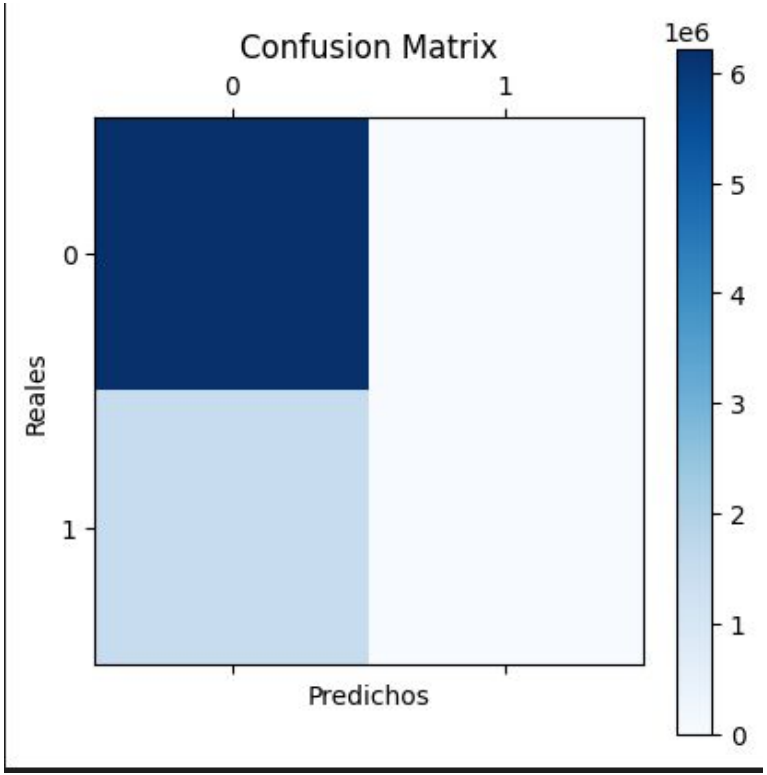
Servidor Actualizado

```python
@app.route("/train_model", methods=["POST"])
def train_model():
    data = request.get_json() or {}
    name = data.get("model_name")
    features = data.get("features", [])
    target = data.get("target")
```

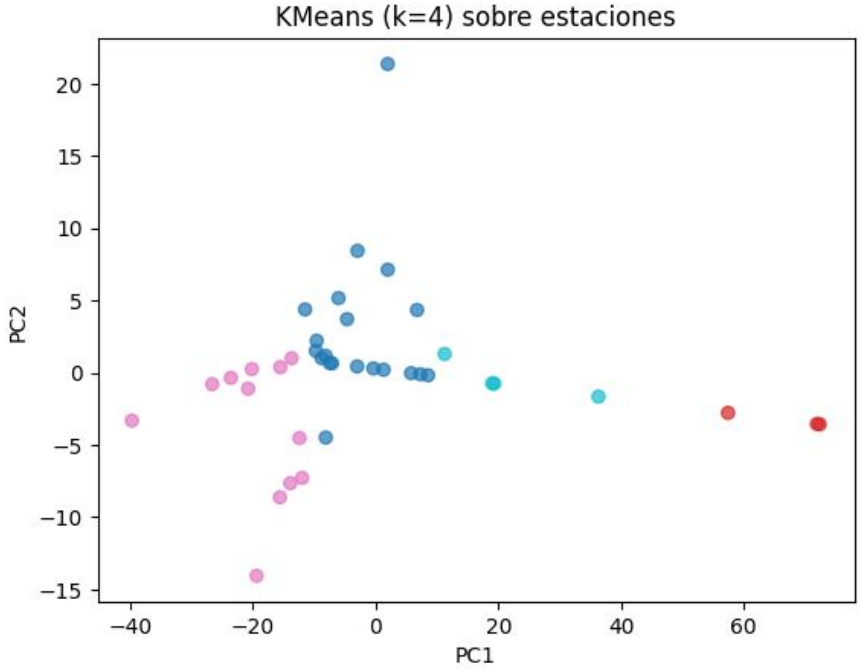
```python
@app.route("/evaluate_model", methods=["POST"])
def evaluate_model():
    data = request.get_json() or {}
    name = data.get("model_name")
```

```python
@app.route("/predict_model", methods=["POST"])
def predict_model():
    data = request.get_json() or {}
    name = data.get("model_name")
```

Funcionalidad




```
Clustering Metrics: {'inertia': 2707.9749586203966, 'silhouette': np.float64(0.41513574968840117)}
```



Proyecto Final - Caso de uso

Dataset - Accidentes de tráfico en EEUU entre 2016 y 2023
Entender los factores de riesgo para mejorar la seguridad vial

Ainoa Iglesias Dasilva
alu0101164403@ull.edu.es
Análisis de Datos Masivos, 2025

1. Objetivos

- Exploración y visualización
  - Analizar tendencias temporales de accidentes (hora, día, mes)
  - Estudiar distribuciones de variables meteorológicas y de visibilidad
- Clasificación de gravedad
  - Predecir si un accidente será grave o leve
  - Comparar Naive Bayes, Árbol de Decisión y k-NN
  - Priorizar recall para no dejar escapar casos graves
- Clustering de condiciones
  - Agrupar accidentes según clima y visibilidad (K-means, jerárquico, DBSCAN)
  - Identificar perfiles de riesgo (días lluviosos, ventosos, etc.)

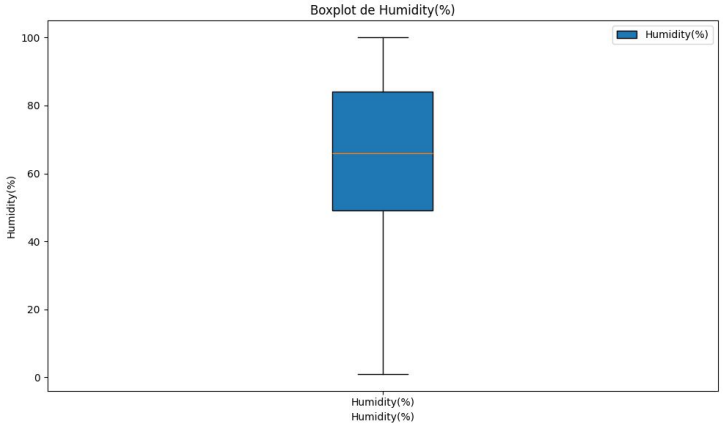
2. Carga y análisis de datos

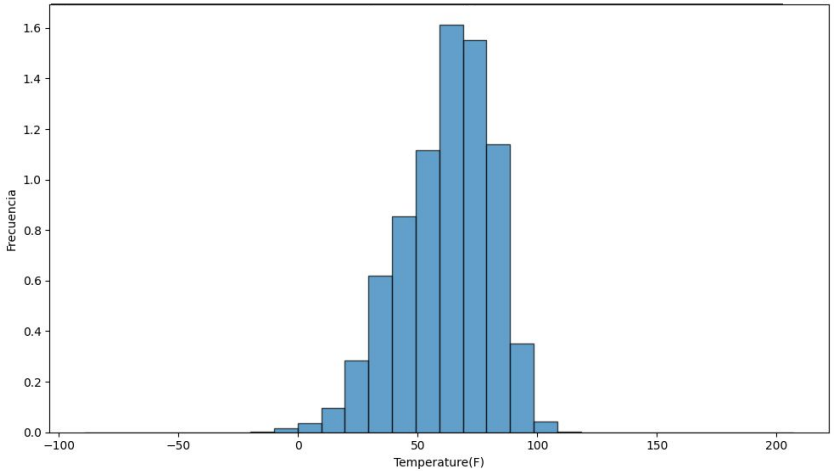
Tipos de dato detectados:

	dtype
Airport_Code	object
Amenity	bool
Astronomical_Twilight	object
Bump	bool
City	object
Civil_Twilight	object
County	object
Crossing	bool
Description	object
Distance(mi)	float64
End_Lat	float64
End_Lng	float64
End_Time	datetime64[ns]
Give_Way	bool
Hour	float64
Humidity(%)	float64
ID	object
Junction	bool
Month	float64
Nautical_Twilight	object
No_Exit	bool
Precipitation(in)	float64
Pressure(in)	float64
Railway	bool

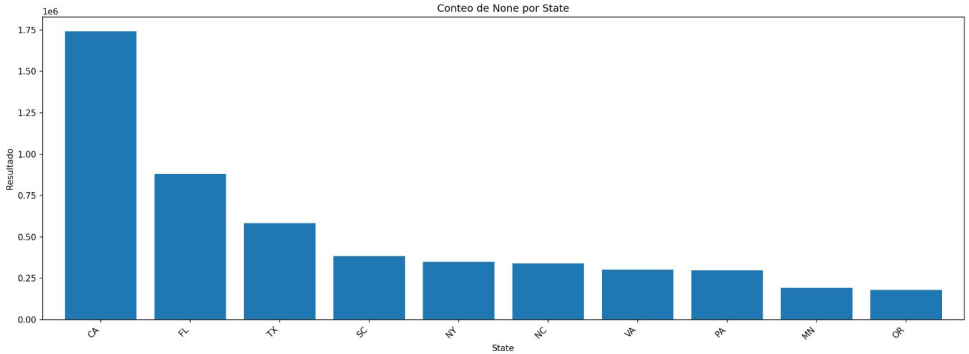
Log de limpieza aplicado:

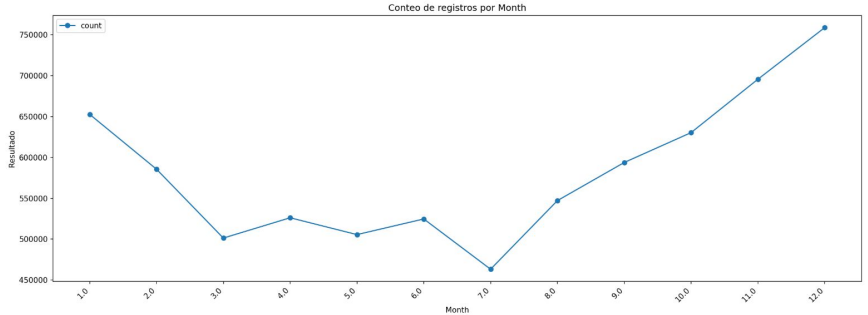
- Eliminadas 2 columnas vacías o constantes.
- 'End_Lat' imputada con media.
- 'End_Lng' imputada con media.
- 'Description' imputada con 'Desconocido'.
- 'Street' imputada con 'Desconocido'.
- 'City' imputada con 'Desconocido'.
- 'Zipcode' imputada con 'Desconocido'.
- 'Timezone' imputada con 'Desconocido'.
- 'Airport_Code' imputada con 'Desconocido'.
- 'Weather_Timestamp' imputada con 'Desconocido'.
- 'Temperature(F)' imputada con media.
- 'Wind_Chill(F)' imputada con media.
- 'Humidity(%)' imputada con media.
- 'Pressure(in)' imputada con media.
- 'Visibility(mi)' imputada con media.
- 'Wind_Direction' imputada con 'Desconocido'.
- 'Wind_Speed(mph)' imputada con media.
- 'Precipitation(in)' imputada con media.
- 'Weather_Condition' imputada con 'Desconocido'.
- 'Sunrise_Sunset' imputada con 'Desconocido'.

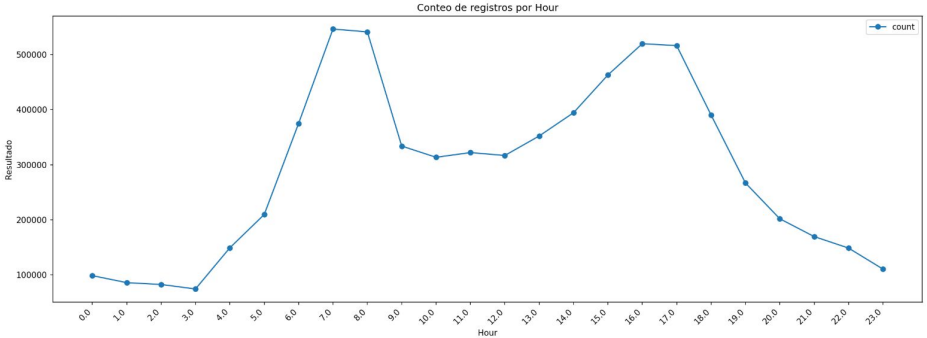


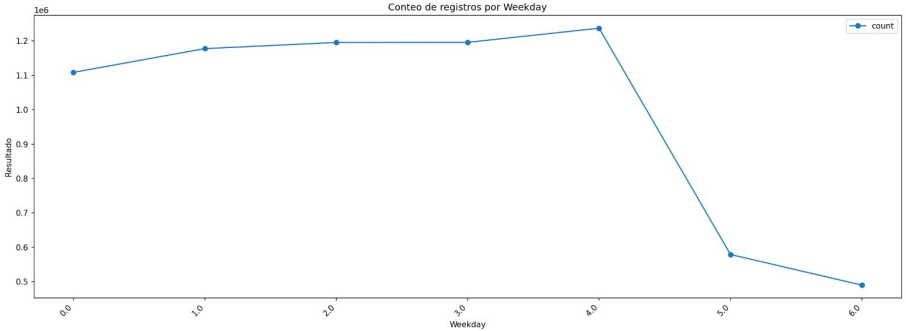


2. Análisis de datos

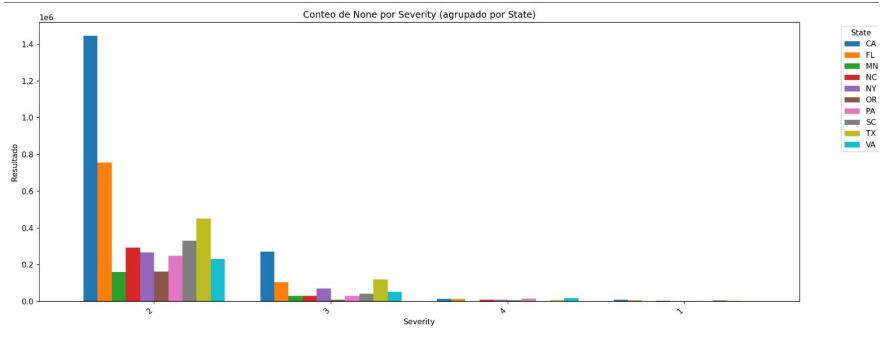


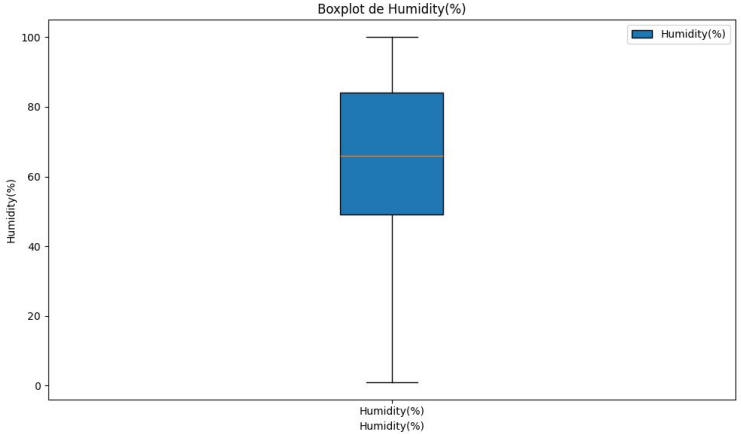


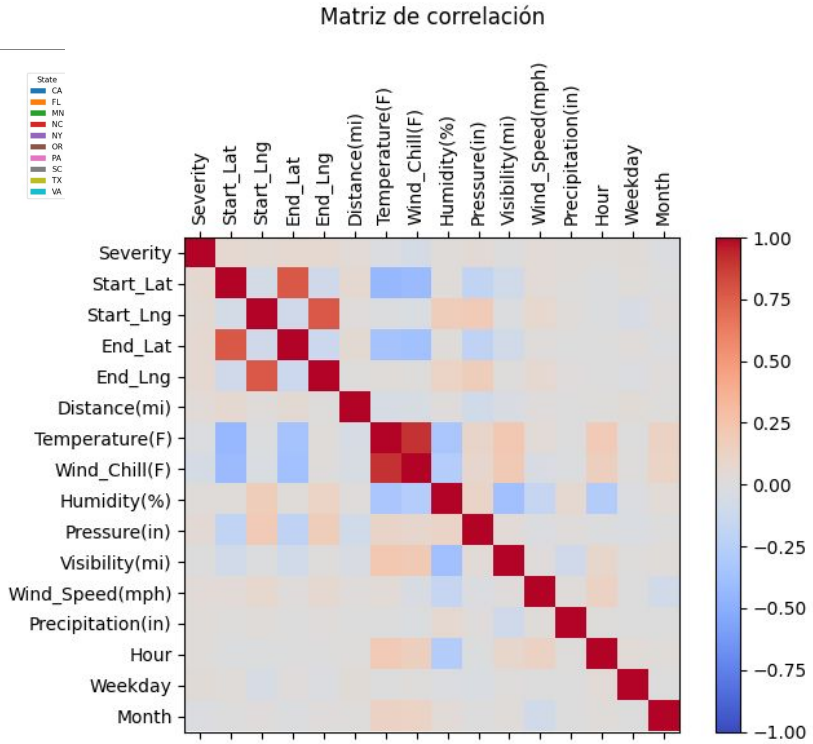




2. Análisis de datos







3. Modelos

- Clasificación
  - Naive Bayes
  - Árbol de Decisión
  - k-NN
- Prioriza recall

- Agrupación
  - KMeans
  - Jerárquico
  - DBSCAN
  - Silhoutte

	precision	recall	accuracy	recall_std
Naive Bayes	0.232707	0.056474	0.818622	0.012421
Decision Tree	0.452394	0.032272	0.838393	0.024211
k-NN	0.259199	0.069534	0.818436	0.005036

✓ 6m 17.7s

	Algoritmo	Clusters	Silhouette
0	KMeans (k=4)	4	0.145529
1	Agglomerative (k=4)	4	0.205806
2	DBSCAN (ε=0.5, min=10)	103	-0.132325

	Distance(mi)	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)
-1	0.026646	-0.077320	-0.005435	-0.033842	-0.036231
0	-0.273165	1.173575	-0.365455	0.358321	0.396421
1	-0.286960	1.159135	-0.079422	0.328900	0.393021
2	-0.274831	1.227832	-0.350862	0.330046	0.395140
3	-0.286706	1.062859	-0.011171	0.369558	0.393425
...	...	...	...	...	...
98	-0.327822	1.144226	-1.058200	0.424945	0.396421
99	-0.278181	0.046370	-0.231548	0.641434	0.396421
100	-0.265590	1.313271	0.047653	0.608204	0.396421
101	-0.271474	0.793613	-0.550895	-0.193046	0.396421
102	-0.325225	0.769509	0.441818	0.511310	0.396421

	Wind_Speed(mph)	Precipitation(in)	Hour	Weekday	Month
-1	0.014400	0.008651	0.004093	0.026268	-0.021278
0	0.174133	-0.093864	0.478407	0.732645	0.264436
1	0.102888	-0.095066	0.070950	0.186079	0.179185
2	0.253641	-0.094383	0.623941	-1.453620	0.403866
3	0.048925	-0.092903	0.104002	-0.360487	0.237439
...	...	...	...	...	...
98	-0.057785	-0.095607	-0.218821	0.732645	-0.157014
99	-0.395429	-0.095607	0.902590	-0.360487	1.129609
100	0.296521	-0.087095	0.084607	1.279211	0.090226
101	0.325901	-0.095607	0.984282	0.732645	-0.430279
102	0.985840	-0.080712	-0.389632	0.732645	-0.544139

7. Conclusiones

- Framework visualizaciones
- Servidor
- Clientes
- Carga y análisis
- Visualizaciones básicas
- Visualizaciones temporales
- Modelos comparador: naive bayes vs. knn vs árbol
- Clustering

- Mayor concentración de accidentes
  - Meses invierno
  - Entre semana
  - 7-8 de la mañana y 5-6 de tarde
- No hay un claro
- Clasificación: Knn mejor predictor accidentes graves
- Agrupación: Jerárquico tiene mejor sithoutte
- 4 clusters
  - alta temperatura + buena visibilidad + poca distancia + horas puntas y entre semana
  - Tº muy muy alta, baja humedad, horas muy concretas + lunes/martes
  - distancia media + humedad neutra + alta presión + ligera precipitación y viento + primeras horas
  - durante el dia + fin de semana + verano + alta presión + ligera lluvia