Informe Técnico: Implementación de K-Means para Jurisdicciones Argentinas

1. Descripción General del Proyecto

El conjunto de scripts implementa un análisis de clustering mediante el algoritmo **K-Means** aplicado a datos de cantidad de viviendas por jurisdicción en Argentina.
El flujo del programa está modularizado en tres archivos:

- **1K-means_jurisdicciones.py**: módulo principal (main).
- **funcion_kmeans.py**: procesamiento del modelo K-Means y cálculo de centroides.
- **funcion_graficos.py**: visualización de los resultados.

El objetivo es agrupar jurisdicciones con volúmenes de viviendas similares y ofrecer una representación visual de dichos clusters.

2. Estructura Técnica y Flujo de Ejecución

2.1. Módulo principal: 1K-means_jurisdicciones.py

- **Responsabilidad:** Cargar los datos, escalar las variables, aplicar el algoritmo y graficar resultados.
- **Aspectos positivos:**
    - Correcto uso de StandardScaler para normalizar los datos.
    - Estructura clara y lineal del flujo.
    - Función cargar_y_escalar_datos() bien delimitada, evita redundancia.
- **Debilidades detectadas:**
    - Los datos están **codificados manualmente** en el script, lo cual limita la escalabilidad.
      -> Se sugiere permitir la **lectura desde un archivo externo (CSV o Excel)**.
    - No existe manejo de **errores o excepciones** (e.g., errores al escalar, valores nulos).

- Falta una **validación del tamaño del dataset** o detección de outliers antes del escalado.
- El valor de k está fijado manualmente (k=3), sin un procedimiento automatizado para determinar el número óptimo de clusters.

2.2. Módulo de procesamiento: funcion_kmeans.py

- **Responsabilidad:** Entrenar el modelo, mostrar el gráfico del método del codo y devolver los resultados.
- **Aspectos positivos:**
  - Correcto uso de sklearn.cluster.KMeans con parámetros reproducibles (random_state, n_init).
  - Incluye el cálculo de centroides en la **escala original**, lo cual mejora la interpretabilidad.
  - Genera etiquetas descriptivas (map_labels) para facilitar la lectura de resultados.
  - El método del codo está implementado correctamente y con visualización clara.
- **Debilidades técnicas:**
  - El gráfico del codo se muestra **cada vez que se ejecuta el script**, lo que podría automatizarse o parametrizarse.
  - El código **imprime resultados directamente en consola**, en lugar de retornarlos para usos posteriores (por ejemplo, visualización o exportación).
  - No hay manejo de posibles **warnings** de sklearn (por ejemplo, si los clusters son vacíos o los datos son escasos).
  - No se valida si el número de clusters (k) es mayor que el número de observaciones, lo que puede generar errores.
  - El **mapa de etiquetas** (map_labels) está rígidamente definido para tres clusters; si se cambia k, el código podría romperse.

2.3. Módulo de gráficos: funcion_graficos.py

- **Responsabilidad:** Visualizar los resultados del clustering mediante dispersión y líneas de centroides.
- **Aspectos positivos:**
    - Uso adecuado de matplotlib con etiquetas claras y leyenda informativa.
    - Buena práctica de ajustar diseño con tight_layout().
    - Representación visual clara para el conjunto de datos reducido.
- **Debilidades o posibles mejoras:**
    - El eje X se construye con índices, pero luego se rotulan con jurisdicciones; podría usarse directamente la columna para evitar desalineaciones.
    - No se utilizan colores diferenciados por cluster, lo que **reduce la legibilidad visual**.
    - No hay control de **orden de las jurisdicciones** (actualmente depende del orden original del DataFrame).
    - No hay opción de **guardar la figura** como archivo (.png, .svg o .pdf).
    - No existe manejo de gráficos en entornos no interactivos (por ejemplo, ejecución en servidores o notebooks sin entorno gráfico).

3. Evaluación Global

Criterio	Evaluación	Comentario
Estructura modular	✅ Buena	Módulos separados y funcionales
Escalabilidad de datos	⚠ Media	Falta lectura externa y validaciones
Manejo de errores	❌ Bajo	No hay control de excepciones
Documentación	⚠ Media	Faltan docstrings detallados y comentarios técnicos

Criterio	Evaluación	Comentario
Visualización	✅ Correcta	Claridad, pero mejorable en estética y legibilidad
Automatización del número de clusters	⚠ Media	Depende del usuario; podría implementarse con criterios estadísticos
Legibilidad del código	✅ Alta	Código limpio y sintaxis clara

4. Recomendaciones Técnicas para Mejoras Futuras

1. **Lectura de datos externa:**
    - Permitir carga desde CSV o Excel con pd.read_csv() o pd.read_excel().
    - Agregar validaciones de columnas esperadas y control de valores faltantes.
2. **Automatización del número óptimo de clusters:**
    - Implementar el método del **coeficiente de silueta** o el **Gap Statistic** para estimar k automáticamente.
    - Permitir al usuario elegir si desea cálculo automático o manual.
3. **Estandarización de visualización:**
    - Añadir colores por cluster (plt.scatter(..., c=df['cluster'], cmap='tab10')).
    - Agregar opción de **exportar gráfico** (plt.savefig()).
    - Incluir una **barra de leyenda** o etiquetas dinámicas sobre puntos.
4. **Manejo de errores y robustez:**
    - Envolver los bloques principales con try/except para capturar errores en escalado o entrenamiento.
    - Validar la dimensión de X_scaled y el rango de k.
5. **Documentación técnica:**

- Agregar **docstrings estilo NumPy o Google** en todas las funciones.
- Incluir comentarios explicativos sobre los pasos del escalado y clustering.

6. **Optimización para datasets mayores:**
   - Reemplazar bucles de visualización por operaciones vectorizadas o usar seaborn para escalabilidad.
   - Ofrecer opción de desactivar gráficos interactivos al correr en modo batch.
7. **Estructura modular avanzada:**
   - Incluir un archivo config.json para definir parámetros como k, colores, rutas de salida, etc.
   - Implementar un main() con argparse para ejecutar desde consola con parámetros dinámicos.

5. Conclusión

El proyecto demuestra una implementación sólida del algoritmo K-Means aplicado al análisis de viviendas por jurisdicción, utilizando herramientas estadísticas y visuales de manera coherente. La estructura modular permite una comprensión rápida de cada etapa del proceso: desde la carga y normalización de los datos hasta la segmentación y representación gráfica de los resultados. El código está escrito con librerías del ecosistema de Python (pandas, numpy, matplotlib, sklearn) y respeta el flujo lógico de una rutina de clustering. En este sentido, se cumple el objetivo técnico de forma efectiva, ofreciendo una clasificación y una visualización clara del comportamiento de las jurisdicciones argentinas según la cantidad de viviendas habitadas.

En cuanto a la salida en consola, el script ofrece una salida de fácil interpretación del modelo. La impresión de los centroides en escala original permite comprender con claridad las magnitudes reales de cada grupo sin depender del conocimiento del proceso de estandarización. Los resultados suelen mostrar tres niveles de agrupamiento: un grupo de jurisdicciones con “pocas viviendas”, otro con “cantidad normal” y un tercero con “muchas viviendas”. Este tipo de segmentación es valioso para

la toma de decisiones en análisis territorial o demográfico, permitiendo identificar contrastes marcados entre grandes centros urbanos (como Buenos Aires o CABA) y provincias de baja densidad. La inclusión de etiquetas descriptivas en texto (“Pocas viviendas”, “Cantidad normal de viviendas”, etc.) haciéndolo más comprensible para usuarios no técnicos.

Sin embargo, el formato de salida presenta oportunidades claras de mejora. Actualmente, los resultados se muestran únicamente por consola, lo que limita su uso posterior para informes, dashboards o integraciones con otras aplicaciones. Una mejora posible sería generar tablas dinámicas o exportar los resultados a CSV/Excel, manteniendo las asignaciones de cada jurisdicción junto con su cluster correspondiente. Asimismo, el script podría incluir un resumen estadístico por grupo (media, desviación estándar, número de elementos), lo que ampliaría el valor analítico de la ejecución. Si bien el gráfico del codo se presenta correctamente, su aparición automática en cada ejecución puede ralentizar los flujos de trabajo; una opción más flexible sería mostrarlo sólo si se activa mediante un parámetro o argumento de línea de comandos.

Desde un punto de vista metodológico, la elección de tres clusters (k=3) ofrece una segmentación razonable para un conjunto de datos reducido como el presente, pero podría beneficiarse de una justificación más formal. Métodos como el coeficiente de silueta, el Gap Statistic o incluso un breve análisis exploratorio previo de dispersión podrían aportar una base estadística más robusta para validar ese número de grupos. Por otro lado, si se ampliara la base de datos para incluir más variables (por ejemplo, superficie territorial, densidad poblacional, o PBI por provincia), el modelo podría evolucionar hacia un análisis más representativo, revelando patrones regionales más complejos y permitiendo entre múltiples factores.

En conclusión, el trabajo desarrollado constituye una implementación técnica eficaz y didáctica del algoritmo K-Means, con un enfoque claro, buena organización modular y resultados interpretables tanto visual como numéricamente. Su principal valor reside en la simplicidad y transparencia del flujo de trabajo, lo cual lo hace ideal para proyectos educativos o exploratorios de segmentación territorial. No obstante, para alcanzar un nivel profesional o de aplicación institucional, se recomienda reforzar la robustez del código con control de errores, automatización de parámetros, capacidad de exportación y análisis complementarios que validen la calidad de los clusters. Con estas

mejoras, el proyecto podría evolucionar de un experimento técnico funcional a una herramienta analítica completa y adaptable a distintos escenarios de datos reales.