# Tópicos Avanzados de IA: Redes complejas y aprendizaje computacional.

Alumno: Ohtli Gerardo Quiroz Sánchez
Prof.: Dr. Matías Alvarado Mentado.

## 1 Objetivo

En este trabajo se plantea una red compleja de las estadísticas operativas selectas de la base de datos de Petróleos Mexicanos (Pemex). Las estadísticas operativas van desde la producción hasta la comercialización interior y exterior de productos como petrolíferos, petroquímicos, gas natural, etc. A partir de la cual se busca encontrar las redes de mundo pequeño (small-world en inglés). Por lo tanto, se hará uso de un método ingenioso como GENIE3 usada en la inferencia de redes reguladoras desde los datos de expresión genética [2]. Este método usa bosques aleatorios como aprendizaje automático para calcular la importancia de una instancia en la predicción de uno de los objetivos [2], por lo que esta importancia se usará como indicador para la construcción de un enlace.

## 2 Teoría

Una red de $|N| = n$ nodos y $|M| = m$ aristas, $G(N, M)$, puede representarse de varios modos entre los cuales está la representación gráfica y la matriz de adyacencia $A = (a_{ij})$ donde

$$a_{ij} = \begin{cases} 1, & \text{si hay una arista entre el nodo i y j} \\ 0, & \text{en otro caso} \end{cases}$$

En una red podemos encontrar varias propiedades que caracterízan, uno de ellos son el coeficiente de agrupamiento y la longitud entre dos nodos.

### 2.1 Coeficiente de Agrupamiento

Consideremos las rutas de longitud 2. Supongamos la ruta formada por una triada de nodos uvw donde tenemos que el nodo v es adyacente a los nodos u y w. Si u y w son adyacentes, uvw forman un triángulo. El coeficiente de

agrupamiento está dada por

$$C = \frac{3(\text{número de triángulos})}{(\text{número de triadas})} \tag{1}$$

es la proporción de triángulos en una red. Un valor del coeficiente va de 0 a 1, si el coeficiente es 0 significa que no hay triángulos en la red; si el coeficiente es 1, cada nodo forma un triángulo con los demás.

## 2.2 Modelo de Grafo aleatorio de Erdos-Rényi

Sea $\mathcal{G}_{n,m}$ el conjunto de todos los grafos con $n$ nodos y exactamente $m$ aristas, $0 \leq m \leq \binom{n}{2} \leq$. Para todo $G \in \mathcal{G}_{n,m}$,

$$P(\mathbb{G}_{n,m} = G) = \binom{\binom{n}{2}}{m}^{-1}$$

donde la probabilidad de un grafo cualquiera con n nodos y m aristas es el inverso de todas las posibles combinaciones de m aristas de $\binom{n}{2}$ posibles aristas.

## 2.3 Mundo pequeño

Una red que presentan la propiedad de mundo pequeño tienden a tener un coeficiente de agrupamiento alto y un promedio de longitud mínima L.

**Definición 1** *La red G se dice que es una red de mundo pequeño si* $L_g \geq L_{rand}$ *y* $C_g^{\Delta} >> C_{rand}^{\Delta}$ *[1].*

Sean

$$\gamma_g = \frac{C_g}{C_{rand}} \tag{2}$$

y

$$\lambda_g = \frac{L_g}{L_{rand}} \tag{3}$$

donde $C_g$ y $L_g$ son el coeficiente de agrupamiento y la longitud promedio de nuestra red, respectivamente. Las variables $C_{rand}$ y $L_{rand}$ corresponden al grafo aleatorio. La métrica de mundo pequeño (small-world-ness en inglés) $S$ está dado por

$$S = \frac{\gamma_g}{\lambda_g} \tag{4}$$

**Definición 2** *Una red se dice que es de mundo pequeño si S>1. [1].*

De esta forma podemos saber con mayor precisión si una red presenta la propiedad de mundo pequeño.

# 3 Metodología

En este trabajo se intenta inferir una red compleja en base a un método ingenioso conocido como GENIE3. Este método ajusta un modelo de bosques aleatorios para cada variable. Posteriormente se infiere una red en base en la importancia que tiene una variable (nodo) en la predicción de otro tomando esto como indicador para crear un enlace o arista entre estas [2]. La propuesta para el conjunto de datos de las operaciones selectas de PEMEX es el siguiente:

- Cada fila es un nodo (Hidrocarburos líquidos, petroleo crudo, complejos petroquímicos, etc.)
- Para cada nodo $i$, se entrena un **Random Forest** como aprendizaje automático para predecir su serie mensual usando, en el mismo mes, las series del resto de nodos.
- Usa el método **importancia por permutación** (permutation importance) para medir la influencia de un nodo sobre otro:
  Una vez entrenado el modelo *Random Forest* de un nodo $i$ (nodo objetivo):
  - Primero se mide qué tan bien predice el modelo.
  - Luego, se toma la serie mensual de un nodo $j$ (por ejemplo, "volumen de ventas de gasolina magna") y se **desordena al azar**.
  - Si al hacer esto el modelo de $i$ pierde precisión, significa que la información de $j$ era importante para predecir a $i$.
  - Cuanto más empeore el modelo al desordenar a $j$, mayor es la influencia de $j$ sobre $i$.

  Se repite este procedimiento para todos los nodos $j$, y se obtiene incluso una medida de influencia dirigida. El resultado final es una **matriz de influencias W** que nos dice quién influye sobre quién y con qué fuerza
- **Umbralización**. Se aplica un umbral a $W$ en base al percentil 85, y otro para 90 para decidir qué influencias son fuertes para convertirse en aristas dentro de un grafo.
- **Calcula la métrica de mundo pequeño (Ecuación 4)**. Si $S > 1$, entonces la red construida es de mundo pequeño.

**Algorithm 1** Construcción de la red de influencias

**Entrada:** Datos $X$ con $T$ meses y $N$ variables (producción, ventas, volumen, etc.), percentil p
Inicializar matriz $W$ de tamaño $N \times N$ en ceros
**for** cada variable $i = 1..N$ **do**
  Definir objetivo $Y$ = variable $i$
  Definir predictores $X_{\text{otros}}$ = todas las variables $j \neq i$
  Entrenar un modelo **Random Forest** para predecir $Y$ usando $X_{\text{otros}}$
  Calcular precisión base del modelo.
  **for** cada predictor $j \in X_{\text{otros}}$ **do**
    Guardar los valores reales de la variable $j$
    Desordenar aleatoriamente los valores de $j$ (romper relación con $Y$)
    Recalcular la precisión del modelo
    Calcular caída en precisión = influencia de $j$ sobre $i$
    Guardar este valor en $W[j, i]$
  **end for**
**end for**
Repetir los pasos anteriores en varios subconjuntos (bootstraps)
Promediar resultados para estabilizar $W$
Aplicar umbral a $W$ en base al percentil p
Construir red dirigida/no dirigida: Aristas con $W[j, i] > 0$
Calcular métricas de la red: Nodos = variables Coeficiente de agrupamiento $C$ Longitud media de camino $L$ Comparar con red aleatoria Erdos Rényi de igual tamaño Si $S > 1$ (Ecuación 4), la red es de tipo "mundo pequeño"
**Salida:** Matriz de influencias $W$, red compleja y valor de S (métrica de small-worldness (4))

## 3.1 Código

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from sklearn.utils import resample
import networkx as nx

# ================== CONFIGURACIÓN ==================
CSV_PATH = "Data/Estadísticas_operativas_seleccionadas_Mayor.csv"  # <-- tu archivo (filas = productos, columnas = meses)
index_col_name = 0  # si la primera columna contiene el nombre del producto
n_estimators = 500
random_state = 50
n_boot = 30  # número de bootstraps temporales
top_percent = 85  # percentil de corte para aristas (mantener las más fuertes)

```

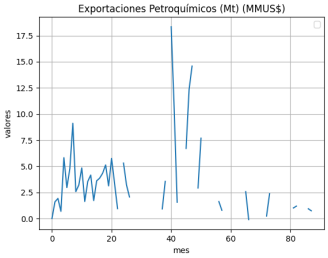
```
# ================== CARGA DE DATOS ==================
df = pd.read_csv(CSV_PATH,encoding="latin-1", index_col=
    index_col_name)

# Normaliza cada fila (producto) con z-score
df_z = df.T.apply(lambda s: (s - s.mean()) / (s.std(ddof=1) + 1e-9)
    ).T

series_names = df_z.index.tolist()
n = df_z.shape[0]

# ================== MATRIZ DE INFLUENCIAS (RF + BOOTSTRAP)
    ==================
W = pd.DataFrame(0.0, index=series_names, columns=series_names)  #
    fuente -> objetivo

rng = np.random.RandomState(random_state)
for target in series_names:
    y = df_z.loc[target, :].values
    # predictores: todas las demás series
    X = df_z.drop(index=target).T.values
    feat_names = df_z.drop(index=target).index.tolist()

    imp_accum = np.zeros(len(feat_names), dtype=float)

    for b in range(n_boot):
        # bootstrap por columnas (meses)
        idx = resample(np.arange(len(y)), replace=True,
    random_state=rng.randint(0, 1_000_000))
        Xb = X[idx, :]
        yb = y[idx]

        rf = RandomForestRegressor(
            n_estimators=n_estimators,
            random_state=rng.randint(0, 1_000_000),
            n_jobs=-1,
            max_features="sqrt"
        )
        rf.fit(Xb, yb)

        # importancia por permutación
        pi = permutation_importance(rf, Xb, yb, n_repeats=5,
                                    random_state=rng.randint(0, 1
    _000_000), n_jobs=-1)
        imp_accum += np.maximum(0, pi.importances_mean)

    imp_mean = imp_accum / n_boot
    if imp_mean.max() > 0:
        imp_mean = imp_mean / imp_mean.max()

    for k, src in enumerate(feat_names):
        W.loc[src, target] = imp_mean[k]


W.to_csv('matrizInfluencias.csv', index=True) #Guardamos los datos

# ================== CONSTRUCCIÓN DE LA RED ==================
```

```
'''
thr es el umbral de corte que selecciona solo las conexiones más fuertes (por percentil), para que la red no se llene de ruido
'''
thr = np.percentile(W.values[W.values > 0], top_percent) if (W.values > 0).sum() > 0 else 1.0

A = (W >= thr).astype(int)  # adyacencia binaria por percentil
G = nx.from_pandas_adjacency(pd.DataFrame(A, index=series_names, columns=series_names),
                             create_using=nx.DiGraph)

# Convertir a no dirigido para small-world clásico
G_und = G.to_undirected()
G_und.remove_nodes_from(list(nx.isolates(G_und)))  # elimina nodos aislados


# ================== FUNCIÓN SMALL-WORLD ==================
def small_world(Gu):
    if Gu.number_of_nodes() < 3 or Gu.number_of_edges() == 0:
        return np.nan, np.nan, np.nan, np.nan, np.nan
    # componente gigante
    components = list(nx.connected_components(Gu))
    GC = Gu.subgraph(max(components, key=len)).copy()

    C = nx.transitivity(GC)  # clustering global
    L = nx.average_shortest_path_length(GC)

    # Grafo aleatorio ER con mismo n y p
    n_nodes = GC.number_of_nodes()
    m_edges = GC.number_of_edges()
    p = (2 * m_edges) / (n_nodes * (n_nodes - 1))
    ER = nx.gnp_random_graph(n_nodes, p, seed=123)
    Cr = nx.transitivity(ER)
    Lr = nx.average_shortest_path_length(ER)

    S = (C/Cr)/(L/Lr) if (Cr > 0 and Lr > 0) else np.nan
    return C, L, Cr, Lr, S


# ================== RESULTADOS ==================
C, L, Cr, Lr, S = small_world(G_und)
print(f"Nodos: {G_und.number_of_nodes()}, Aristas: {G_und.number_of_edges()}")
print(f"C={C:.3f}, L={L:.3f}, C_rand={Cr:.3f}, L_rand={Lr:.3f}, S={S:.3f}")

if S > 1:
    print("La red muestra propiedades de mundo pequeño.")
else:
    print("No se detectan propiedades claras de mundo pequeño.")
```

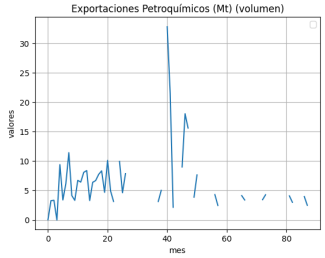
Listing 1: Función Reference Set Update

# 4 Datos

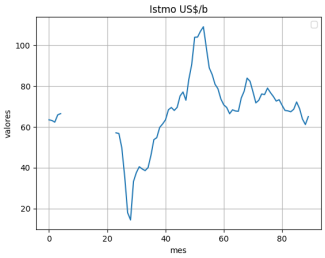
Los datos se obtuvieron de las estadísticas operativas selectas de la base de datos de Petróleos Mexicanos (Pemex) (página https://ebdi.pemex.com/bdi/). Algunos de los datos que fueron removidos fueron porque tenían demasiados datos faltantes (valores N/D equivalentemente a valores nan), estos fueron exportaciones Petroquímicos (Mt) (MMUS), Exportaciones Petroquímicos (Mt) (volumen) y Istmo US$/b. En la Figura (1) se puede apreciar las graficas de estas instancias que no tienen valores faltantes.




(a) Exportaciones Petroquímicos (Mt) (MMUS).




(b) Exportaciones Petroquímicos (Mt) (volumen).




(c) Istmo US$/b.

Figure 1: Valores faltantes.

# 5 resultados

En la ejecución del programa se estableció la siguiente configuración:

1. n_estimators = 500 (número de árboles)
2. random_state = 50 (semilla)
3. n_boot = 30 número de bootstraps (re-muestreo con reemplazo)
4. top_percent = percentil 85 y 90 de corte para aristas (mantener las más fuertes)

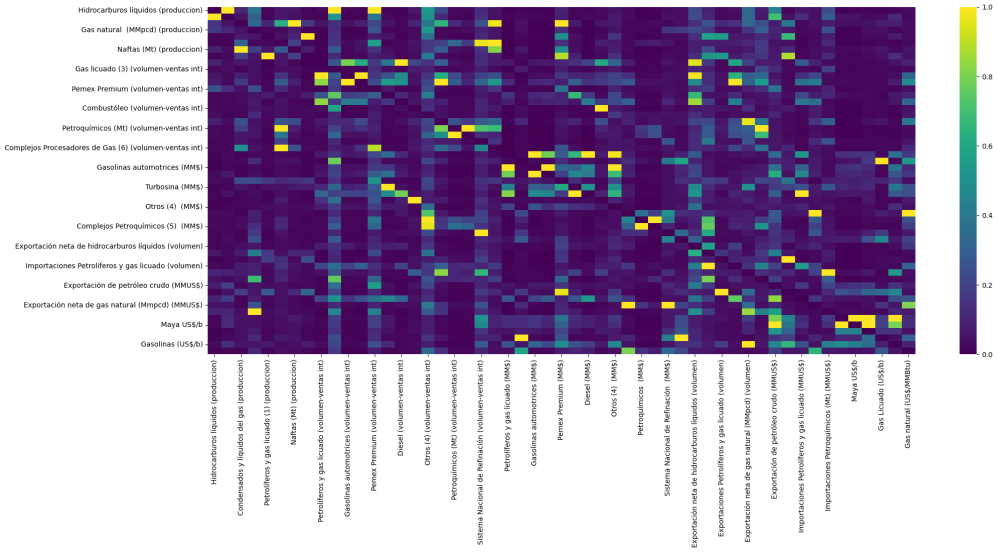



Figure 2: Mapa de calor de la matriz de influencias

| Percentil | Nodos | Aristas | C | L | $C_{rand}$ | $L_{rand}$ | S |
|---|---|---|---|---|---|---|---|
| 85 | 53 | 319 | 0.422 | 1.910 | 0.229 | 1.787 | 1.728 |
| 90 | 53 | 221 | 0.362 | 2.231 | 0.172 | 2.037 | 1.926 |

Table 1: Resultados de ambas redes complejas. Ambas presentan la propiedad de mundo pequeño.

Al terminar la ejecución se guardará la matriz de influencias W para que no halla necesidad de tener que ejecutar el programa de nuevo debido a su costo computacional.

En la Figura (2) muestro el mapa de calor de la matriz de influencias, es decir, qué tanto cada variable (ej. exportación, ventas internas, etc.) contribuye a explicar a las demás en el modelo Random Forest. A partir de esta matriz se decide crear dos umbrales a partir del cual vamos a quitar las aristas que tengan un valor menor al percentil 85 y al percentil 90. Posteriormente procedemos a crear la red compleja que se muestra en las Figuras (3) y (4).

Los resultados de la red compleja creada a partir del umbral con percentil 85 en la matriz de influencias son 53 nodos y 319 aristas donde presentó un valor en la métrica de mundo pequeño S igual a 1.728, por lo que se considera de mundo pequeño (véase la Ecuación 4). En el percentil 90, disminuyeron aún más el número de aristas hasta 221 con un aumento en S de 1.926 por lo que también es de mundo pequeño. En la Tabla (1) se resumen estos valores.

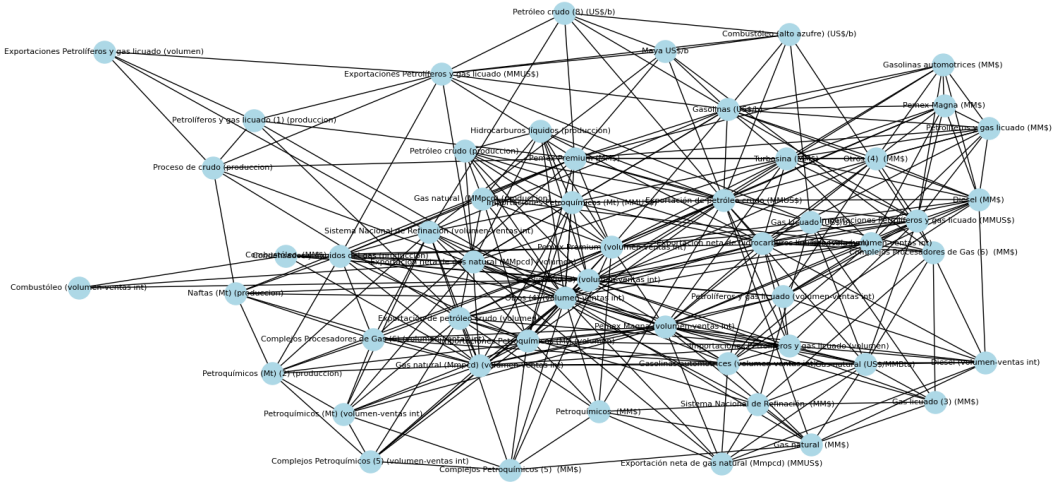



Figure 3: Red compleja con el percentil 85

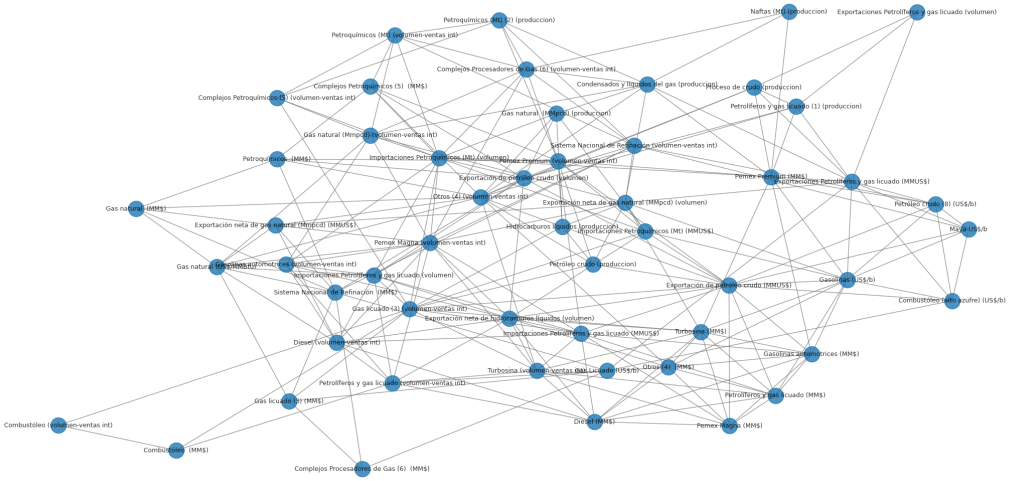



Figure 4: Red compleja con el percentil 90

# References


[1] Mark D. Humphries and Kevin Gurney. "Network 'Small-World-Ness': A Quantitative Method for Determining Canonical Network Equivalence". In: *PLOS ONE* 3.4 (Apr. 2008), pp. 1–10. DOI: 10.1371/journal.pone.0002051. URL: https://doi.org/10.1371/journal.pone.0002051.

[2] Vân Anh Huynh-Thu et al. "Inferring Regulatory Networks from Expression Data Using Tree-Based Methods". In: *PLOS ONE* 5.9 (Sept. 2010), pp. 1–10. DOI: 10.1371/journal.pone.0012776. URL: https://doi.org/10.1371/journal.pone.0012776.