

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 [3]. Este método usa bosques aleatorios como aprendizaje automático para calcular la importancia de una instancia en la predicción de uno de los objetivos [3], 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

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

En una red podemos encontrar varias propiedades que caracterizan, 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

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

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, $0 \leq m \leq \binom{n}{2}$. 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 $C_g \gg C_{rand}^\Delta$ [2].

Sean

$$\gamma_g = \frac{C_g}{C_{rand}} \quad (2)$$

y

$$\lambda_g = \frac{L_g}{L_{rand}} \quad (3)$$

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

$$S = \frac{\gamma_g}{\lambda_g} \quad (4)$$

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

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

3 Metodología

En este trabajo se inferirá en base a un método ingenioso como GENIE3 usada en la inferencia de redes reguladoras desde los datos de expresión genética [3]. Este método usa bosques aleatorios como aprendizaje automático para calcular la importancia de una instancia en la predicción de otro [3]. Este indicador de importancia se usará para la construcción de un enlace. El indicador de importancia usado en el modelo de bosques aleatorios es *importancia de permutación* que ha mostrado ser útil para modelos que capturan relaciones complejas y no lineales entre las características y la variable objetivo [1]. El esquema del algoritmo es el siguiente [1]:

- Entradas: modelo predictivo ajustado m , conjunto de datos tabular (entrenamiento o validación) D .
- Calcular la puntuación de referencia s del modelo m en los datos D (por ejemplo, la exactitud en el caso de un clasificador o el R^2 en el caso de un regresor).
- Para cada característica j (columna de D):
 - Para cada repetición k en $1, \dots, K$:
 - * Barajar aleatoriamente la columna j del conjunto de datos D para generar una versión corrompida de los datos llamada $\tilde{D}_{k,j}$.
 - * Calcular la puntuación $s_{k,j}$ del modelo m en los datos corrompidos $\tilde{D}_{k,j}$.
 - Calcular la importancia i_j para la característica f_j , definida como:

$$i_j = s - \frac{1}{K} \sum_{k=1}^K s_{k,j}$$

A partir de la importancia de permutación se construirá una matriz de adyacencia que tenga pesos en lugar de la binaria como se mostró anteriormente.

3.1 Código

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.ensemble import RandomForestRegressor
4 from sklearn.inspection import permutation_importance
5 from sklearn.utils import resample
6 import networkx as nx
7
8 # ===== CONFIGURACIÓN =====
9 CSV_PATH = "Data/Estadísticas_operativas_seleccionadas_Mayor.csv"
10 # <-- tu archivo (filas = productos, columnas = meses)
```

```

10 index_col_name = 0 # si la primera columna contiene el nombre del
    producto
11 n_estimators = 500
12 random_state = 50
13 n_boot = 30 # número de bootstraps temporales
14 top_percent = 85 # percentil de corte para aristas (mantener las m
    ás fuertes)
15
16 # ===== CARGA DE DATOS =====
17 df = pd.read_csv(CSV_PATH, encoding="latin-1", index_col=
    index_col_name)
18
19 # Normaliza cada fila (producto) con z-score
20 df_z = df.T.apply(lambda s: (s - s.mean()) / (s.std(ddof=1) + 1e-9)
    ).T
21
22 series_names = df_z.index.tolist()
23 n = df_z.shape[0]
24
25 # ===== MATRIZ DE INFLUENCIAS (RF + BOOTSTRAP)
    =====
26 W = pd.DataFrame(0.0, index=series_names, columns=series_names) #
    fuente -> objetivo
27
28 rng = np.random.RandomState(random_state)
29 for target in series_names:
30     y = df_z.loc[target, :].values
31     # predictores: todas las demás series
32     X = df_z.drop(index=target).T.values
33     feat_names = df_z.drop(index=target).index.tolist()
34
35     imp_accum = np.zeros(len(feat_names), dtype=float)
36
37     for b in range(n_boot):
38         # bootstrap por columnas (meses)
39         idx = resample(np.arange(len(y)), replace=True,
            random_state=rng.randint(0, 1_000_000))
40         Xb = X[idx, :]
41         yb = y[idx]
42
43         rf = RandomForestRegressor(
44             n_estimators=n_estimators,
45             random_state=rng.randint(0, 1_000_000),
46             n_jobs=-1,
47             max_features="sqrt"
48         )
49         rf.fit(Xb, yb)
50
51         # importancia por permutación
52         pi = permutation_importance(rf, Xb, yb, n_repeats=5,
            random_state=rng.randint(0, 1
53             _000_000), n_jobs=-1)
54         imp_accum += np.maximum(0, pi.importances_mean)
55
56     imp_mean = imp_accum / n_boot
57     if imp_mean.max() > 0:
58         imp_mean = imp_mean / imp_mean.max()

```

```

59
60     for k, src in enumerate(feats_names):
61         W.loc[src, target] = imp_mean[k]
62
63
64 W.to_csv('matrizInfluencias.csv', index=True) #Guardamos los datos
65
66 # ===== CONSTRUCCIÓN DE LA RED =====
67 '''
68 thr es el umbral de corte que selecciona solo las conexiones más
69 fuertes (por percentil), para que la red no se llene de ruido
70 '''
71 thr = np.percentile(W.values[W.values > 0], top_percent) if (W.
72 values > 0).sum() > 0 else 1.0
73
74 A = (W >= thr).astype(int) # adyacencia binaria por percentil
75 G = nx.from_pandas_adjacency(pd.DataFrame(A, index=series_names,
76 columns=series_names),
77                             create_using=nx.DiGraph)
78
79 # Convertir a no dirigido para small-world clásico
80 G_und = G.to_undirected()
81 G_und.remove_nodes_from(list(nx.isolates(G_und))) # elimina nodos
82 aislados
83
84 # ===== FUNCIÓN SMALL-WORLD =====
85 def small_world_sigma(Gu):
86     if Gu.number_of_nodes() < 3 or Gu.number_of_edges() == 0:
87         return np.nan, np.nan, np.nan, np.nan, np.nan
88     # componente gigante
89     components = list(nx.connected_components(Gu))
90     GC = Gu.subgraph(max(components, key=len)).copy()
91
92     C = nx.transitivity(GC) # clustering global
93     L = nx.average_shortest_path_length(GC)
94
95     # Grafo aleatorio ER con mismo n y p
96     n_nodes = GC.number_of_nodes()
97     m_edges = GC.number_of_edges()
98     p = (2 * m_edges) / (n_nodes * (n_nodes - 1))
99     ER = nx.gnp_random_graph(n_nodes, p, seed=123)
100     Cr = nx.transitivity(ER)
101     Lr = nx.average_shortest_path_length(ER)
102
103     S = (C / Cr) / (L / Lr) if (Cr > 0 and Lr > 0) else np.nan
104     return C, L, Cr, Lr, S
105
106 # ===== RESULTADOS =====
107 C, L, Cr, Lr, S = small_world_sigma(G_und)
108 print(f"Nodos: {G_und.number_of_nodes()}, Aristas: {G_und.
109       number_of_edges()}")
110 print(f"C={C:.3f}, L={L:.3f}, C_rand={Cr:.3f}, L_rand={Lr:.3f}, S={
111       S:.3f}")
112
113 if S > 1:

```

```

110     print("La red muestra propiedades de mundo pequeño.")
111 else:
112     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.

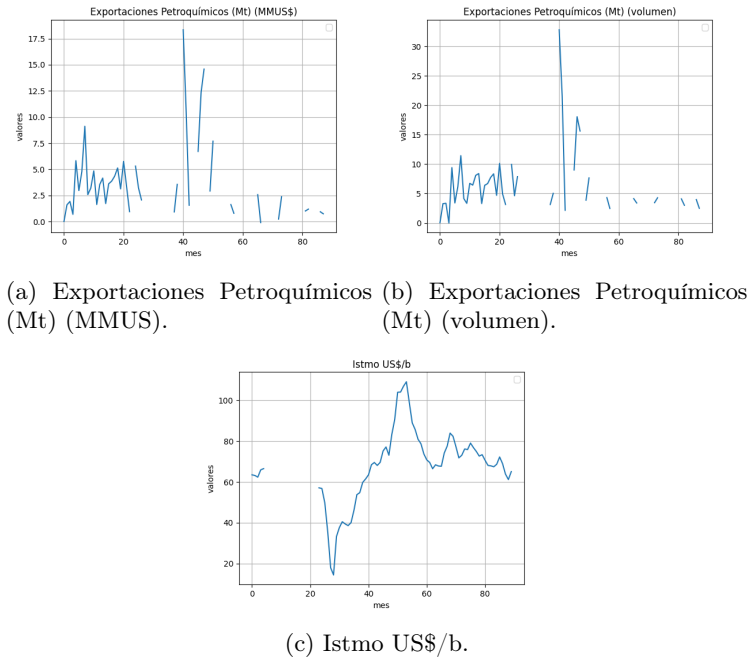


Figure 1: Valores faltantes.

5 resultados

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

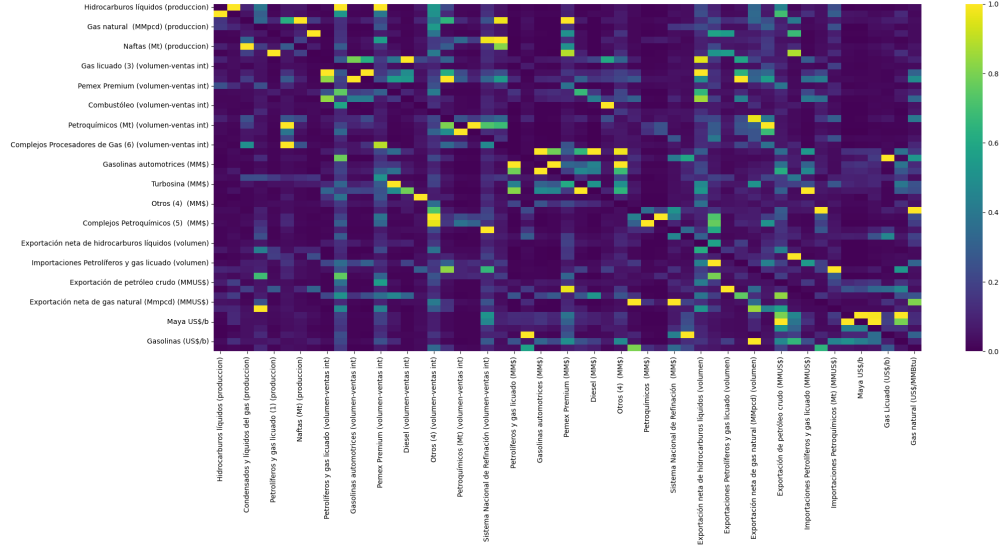


Figure 2: Mapa de calor de la matriz de influencias

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.

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, pero aumentó el valor de S en 1.926. En la Tabla (1) se resumen estos valores.

References

- [1] 5.2. Permutation feature importance — scikit-learn 1.7.0 documentation - sklearn. URL: https://sklearn.org/stable/modules/permutation_importance.html#outline-of-the-permutation-importance-algorithm.



- [2] 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>.
- [3] 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>.