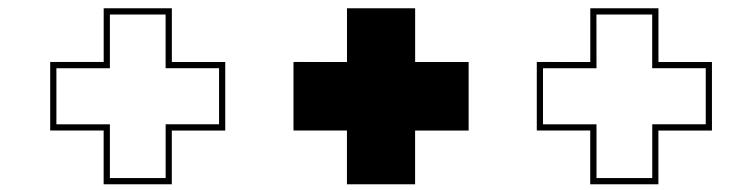


PRESENTACIÓN DE PROYECTO

DETECCIÓN DE MEDICAMENTOS CON MAYOR RIESGO DE DESABASTECIMIENTO

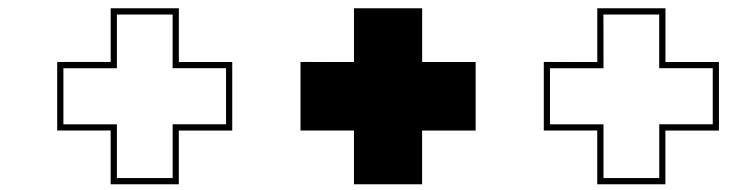
Presentada por: Oscar Silva - 2220087
Edgar Ramírez - 2220101





MOTIVACIÓN

El desabastecimiento de medicamentos vitales en Colombia afecta la calidad de vida de muchas personas y genera costos adicionales en el sistema de salud. Este proyecto busca prevenirlo mediante inteligencia artificial, analizando datos de consumo, precios y logística para predecir escasez y alertar a farmacias y distribuidores. Con ello, se optimiza la disponibilidad de medicamentos y se mejora el acceso oportuno para la población.



OBJETIVO

Desarrollar un modelo de inteligencia artificial que analice datos de consumo, y logística para predecir el riesgo de desabastecimiento de medicamentos no vitales en Colombia, permitiendo a farmacias, distribuidores y autoridades de salud tomar decisiones preventivas y optimizar su disponibilidad.

INFORMACIÓN DEL DATASET

Dentro del dataset se puede presenciar una lista de diagnósticos para diferentes enfermedades con fecha y código, junto a indicación del nombre del medicamento necesario para el tratamiento junto con su concentración, forma, medida, principio activo, y cantidad solicitada por cada entidad de salud.

Actualizado
21 de enero de 2025

Última actualización de los datos
21 de enero de 2025

Última actualización de metadatos
21 de enero de 2025

Fecha de creación
10 de octubre de 2018

Vistas
135K

Descargas
17K

Suministró los datos
Instituto Nacional de Vigilancia de Medicamentos y Alimentos

Propietario de conjunto de datos
Invima

Información de la Entidad

| | |
|----------------------|----------------------------|
| Departamento | Bogotá D.C. |
| Municipio | Bogotá D.C. |
| Nombre de la Entidad | Instituto Nacional de Vigi |
| Orden | Nacional |
| Sector | Salud y Protección Soc |
| Área o dependencia | Operaciones Sanitarias |

Información de Datos

| | |
|-----------------------------|------------|
| Idioma | Español |
| Cobertura Geográfica | Nacional |
| Frecuencia de Actualización | Mensual |
| Fecha Emisión (aaaa-mm-dd) | 2015-02-20 |

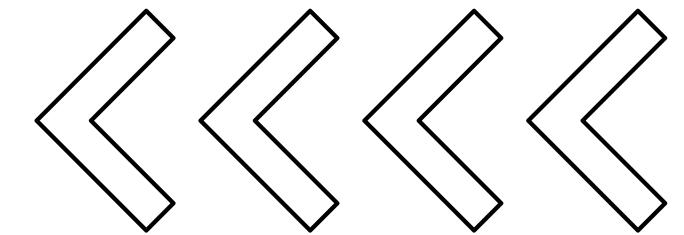
Temas

| | |
|-----------|------------------------|
| Categoría | Salud y Protección Soc |
| Etiquetas | vitales |

Licencia y atribución

| | |
|---------------------|---|
| Licencia |  |
| Enlace de la fuente | https://www.invima.gov |

INFORMACION DEL DATASET



| | | | |
|------------|---|------------|-------------------------------|
| [REDACTED] | Fecha de autorizacion | [REDACTED] | Principios activos |
| [REDACTED] | Solicitante | [REDACTED] | Concentracion del medicamento |
| [REDACTED] | Identificador unico de medicamento (IUM) | [REDACTED] | Cantidad solicitada |
| [REDACTED] | Unidades de medida | [REDACTED] | Diagnostico |

PROCESAMIENTO DEL DATASET



Dropear las columnas Concentración y Unidad de medida.

```
df = df.drop(['UNIDAD_MEDIDA1', 'UNIDAD_MEDIDA2', 'CONCENTRACIÓN_DELMEDICAMENTO1', 'CONCENTRACIÓN_DEL_MEDICAMENTO2'], axis=1)
```



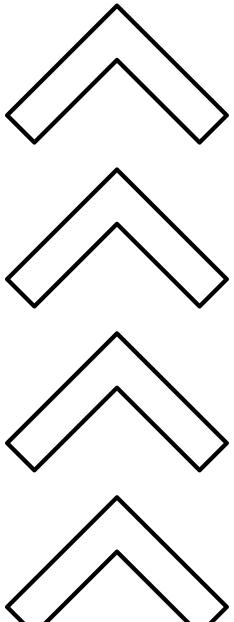
Convertir FECHA_DE_AUTORIZACIÓN a tipo "datetime".

```
df['FECHA_DE_AUTORIZACIÓN'] = pd.to_datetime(df['FECHA_DE_AUTORIZACIÓN'], errors='coerce')
```



Reemplazar valores nulos de CANTIDAD_SOLICITADA por "0".

```
df['CANTIDAD_SOLICITADA'] = df['CANTIDAD_SOLICITADA'].fillna(0)
```



PROCESAMIENTO DEL DATASET



Estandarizar NOMBRE_COMERCIAL_ según IUM.

```
diferentes_nombres = df.groupby('IUM')['NOMBRE_COMERCIAL_'].unique()
diferentes_nombres_df = diferentes_nombres.reset_index()
print(diferentes_nombres_df)

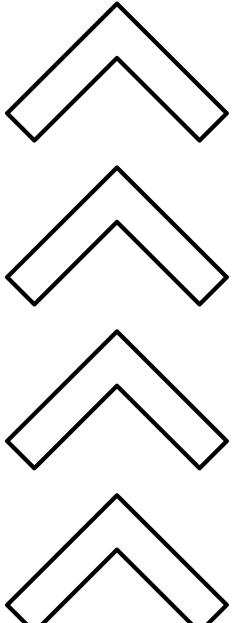
mapeo = {ium: nombres[0] for ium, nombres in diferentes_nombres.items()}
print(mapeo)
df['NOMBRE_COMERCIAL_'] = df['IUM'].map(mapeo)
```



Estandarizar DIAGNOSTICO_CIE-1NO REPORTA según CÓDIGO_DIAGNOSTICO_CIE-10.

```
diferentes_diagnosticos = df.groupby('CÓDIGO_DIAGNOSTICO_CIE-10')['DIAGNOSTICO_CIE-1NO REPORTA'].unique()
diferentes_diagnosticos_df = diferentes_diagnosticos.reset_index()
print(diferentes_diagnosticos_df)

mapeo_diagnosticos = {codigo: diagn[0] for codigo, diagn in diferentes_diagnosticos.items()}
print(mapeo_diagnosticos)
df['DIAGNOSTICO_CIE-1NO REPORTA'] = df['CÓDIGO_DIAGNOSTICO_CIE-10'].map(mapeo_diagnosticos)
```



CARACTERÍSTICAS

1 TIPO_DE_SOLICITUD:

Permite identificar si se trata de una solicitud de un paciente o varios

2 PRINCIPIO_ACTIVO:

Identifica el medicamento solicitado.

3 FORMA_FARMACÉUTICA:

Diferentes formas pueden tener distintos niveles de disponibilidad.

4 CÓDIGO_DIAGNOSTICO_CIE-10:

Permite relacionar diagnósticos específicos con la demanda de medicamentos.

5 CANTIDAD_SOLICITADA:

Cantidades muy altas o repetitivas pueden indicar falta de suministro.

6 FECHA:

Para analizar tendencias temporales en solicitudes.

MODELO DE CLASIFICACIÓN

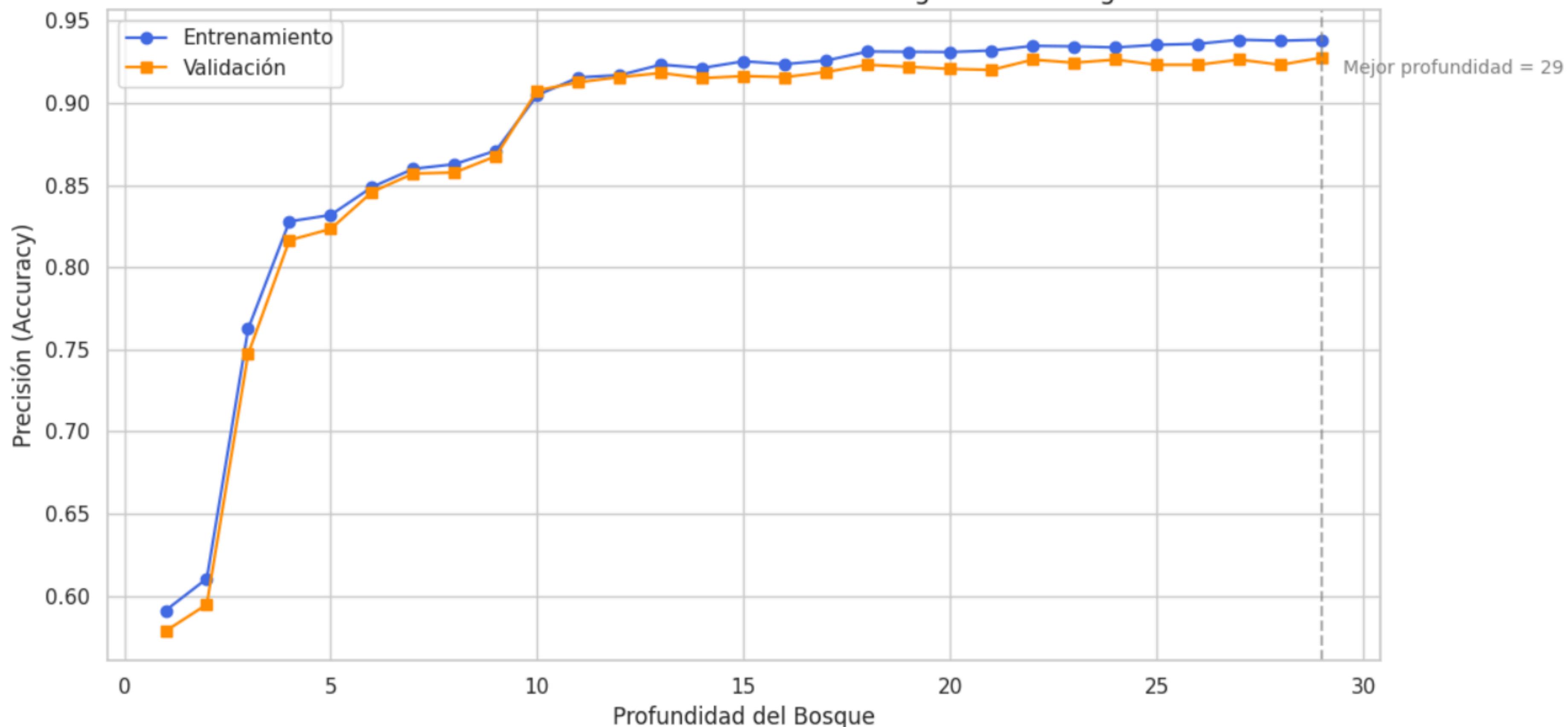
El modelo de clasificación tiene como objetivo predecir si un medicamento estará disponible o no disponible en función de las características. A través de un enfoque de clasificación, el modelo asigna una etiqueta binaria (0 o 1) a cada caso:

- 0 si el medicamento está disponible
- 1 si el medicamento no está disponible.

Este modelo nos permite identificar qué medicamentos tienen una mayor probabilidad de enfrentar desabastecimiento.

| Decision Tree Accuracy: 0.9650436953807741 | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.99 | 0.95 | 0.97 | 927 |
| 1 | 0.94 | 0.98 | 0.96 | 675 |
| accuracy | | | 0.97 | 1602 |
| macro avg | 0.96 | 0.97 | 0.96 | 1602 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1602 |
| Random Forest Accuracy: 0.9650436953807741 | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.99 | 0.94 | 0.97 | 927 |
| 1 | 0.93 | 0.99 | 0.96 | 675 |
| accuracy | | | 0.97 | 1602 |
| macro avg | 0.96 | 0.97 | 0.96 | 1602 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1602 |
| SVC Accuracy: 0.8732833957553059 | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.88 | 0.90 | 0.89 | 927 |
| 1 | 0.86 | 0.83 | 0.85 | 675 |
| accuracy | | | 0.87 | 1602 |
| macro avg | 0.87 | 0.87 | 0.87 | 1602 |
| weighted avg | 0.87 | 0.87 | 0.87 | 1602 |

Random Forest - Clasificación: Underfitting vs Overfitting



COMPARACIÓN RESULTADOS (CLASIFICACION)

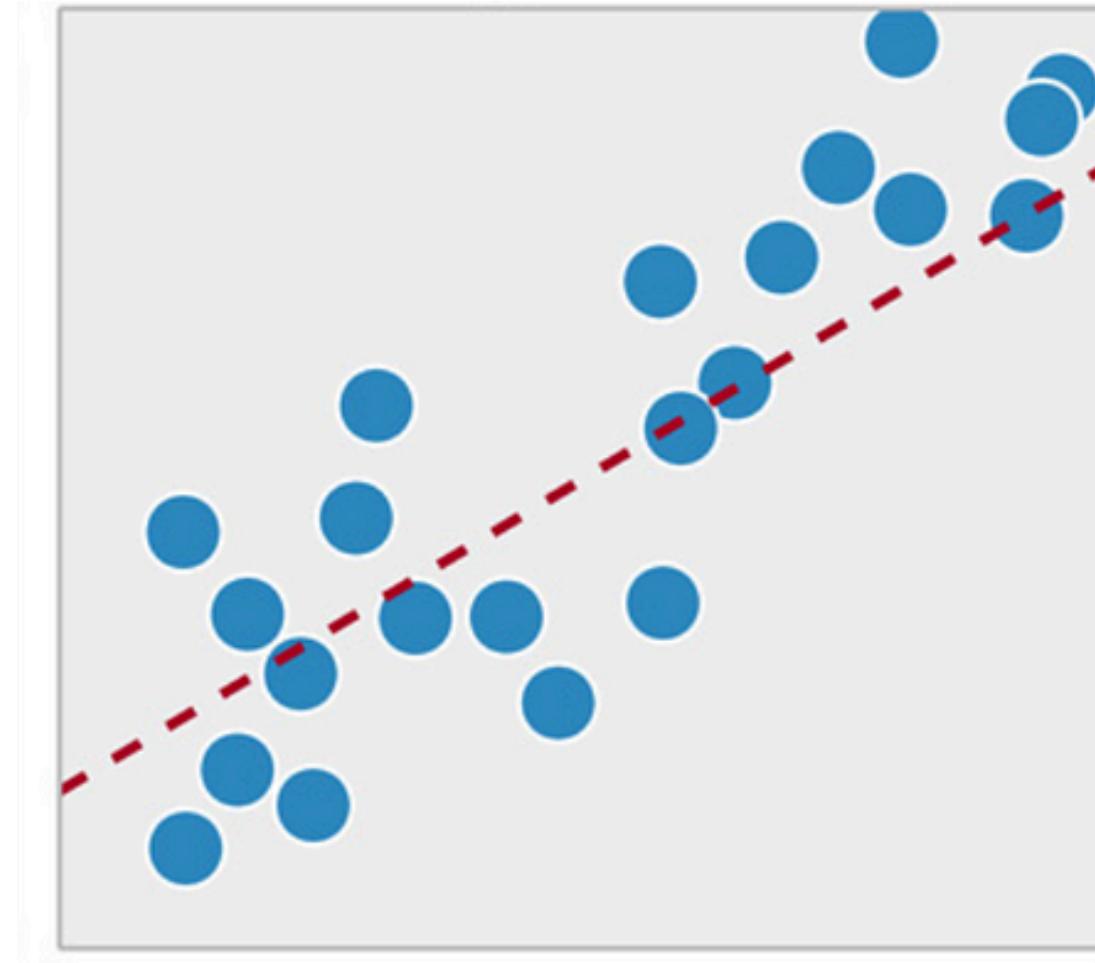
| | Accuracy | | Precision | Recall | F1-score |
|-----|----------|---|-----------|--------|----------|
| DTC | 0.97 | 0 | 0.99 | 0.95 | 0.97 |
| | | 1 | 0.94 | 0.98 | 0.96 |
| RFC | 0.97 | 0 | 0.99 | 0.94 | 0.97 |
| | | 1 | 0.93 | 0.99 | 0.96 |
| SVC | 0.87 | 0 | 0.88 | 0.90 | 0.89 |
| | | 1 | 0.86 | 0.83 | 0.85 |

MODELO DE REGRESIÓN

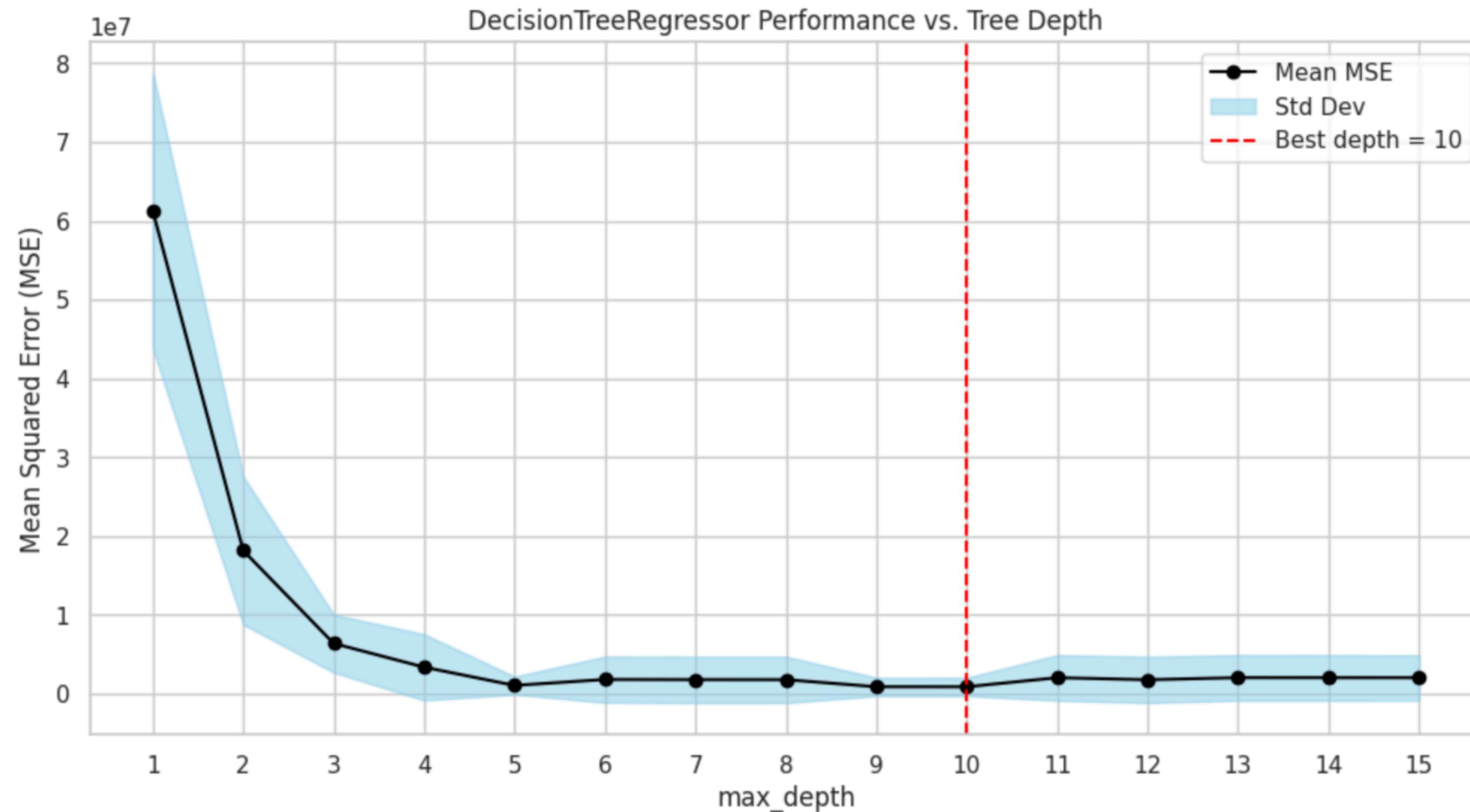
- **Decision Tree Regressor (DTR)**
- **Random Forest Regressor (RFR)**
- **Support Vector Regressor (SVR)**
- **Deep Neural Network (DNN)**

Mientras el modelo de clasificación determina si un medicamento estará disponible o no, el de regresión se aplica para la ‘Cantidad Solicitada’ de medicamento.

Su objetivo es estimar la gravedad del desabastecimiento. Así, se permite priorizar los casos más críticos y tomar decisiones más informadas.



DECISION TREE REGRESSOR



DECISION TREE REGRESSOR

Por Defecto ->

```
dtr = DecisionTreeRegressor(random_state=42)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mae_scores = cross_val_score(dtr, X, y, cv=kf, scoring=mae_scorer)
mse_scores = cross_val_score(dtr, X, y, cv=kf, scoring=mse_scorer)

print(" Decision Tree Regressor (DTR) ")
print(f"MAE promedio: {mae_scores.mean():.2f} ± {mae_scores.std():.2f}")
print(f"MSE promedio: {mse_scores.mean():.2f} ± {mse_scores.std():.2f}")
```

```
Decision Tree Regressor (DTR)
MAE promedio: 50.91 ± 39.27
MSE promedio: 2050283.51 ± 2864385.89
```

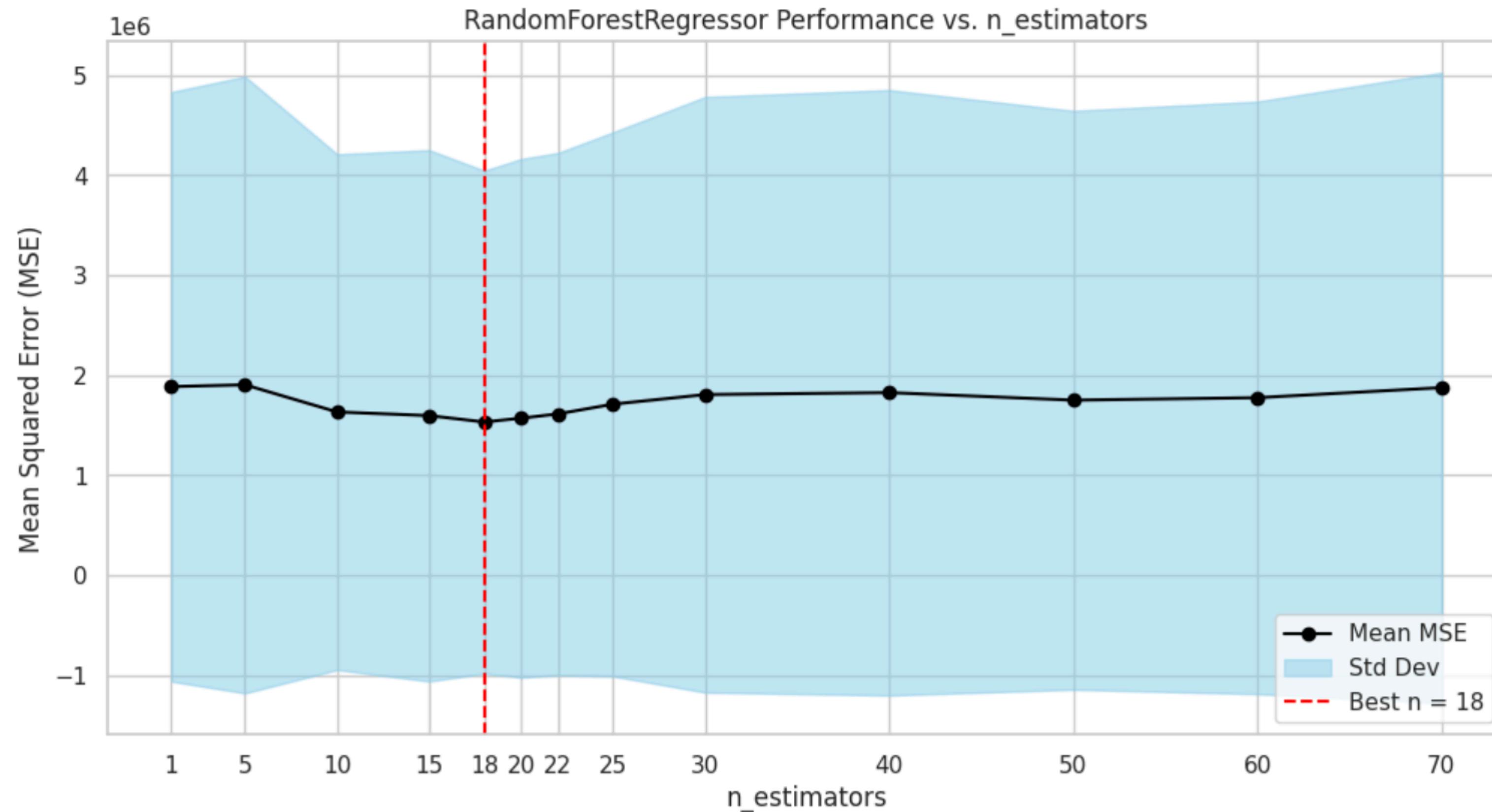
```
reg = DecisionTreeRegressor(max_depth=10, random_state=42)
mae_scores = cross_val_score(reg, X, y, cv=kf, scoring=mae_scorer)
mse_scores = cross_val_score(reg, X, y, cv=kf, scoring=mse_scorer)

print(" Decision Tree Regressor (DTR - max_depth=10) ")
print(f"MAE promedio: {mae_scores.mean():.2f} ± {mae_scores.std():.2f}")
print(f"MSE promedio: {mse_scores.mean():.2f} ± {mse_scores.std():.2f}")
```

```
Decision Tree Regressor (DTR - max_depth=10)
MAE promedio: 42.31 ± 26.17
MSE promedio: 899300.97 ± 1145168.30
```

<- Optimizado

RANDOM FOREST REGRESSOR



RANDOM FOREST REGRESSOR

Por Defecto ->

```
rf = RandomForestRegressor(random_state=42)
mae_scores = cross_val_score(rf, X, y, cv=kf, scoring=mae_scorer)
mse_scores = cross_val_score(rf, X, y, cv=kf, scoring=mse_scorer)

print(" Random Forest Regressor (RFR)")
print(f"MAE promedio: {mae_scores.mean():.2f} ± {mae_scores.std():.2f}")
print(f"MSE promedio: {mse_scores.mean():.2f} ± {mse_scores.std():.2f}")
```

```
Random Forest Regressor (RFR)
MAE promedio: 44.84 ± 42.13
MSE promedio: 1841891.65 ± 3164847.05
```

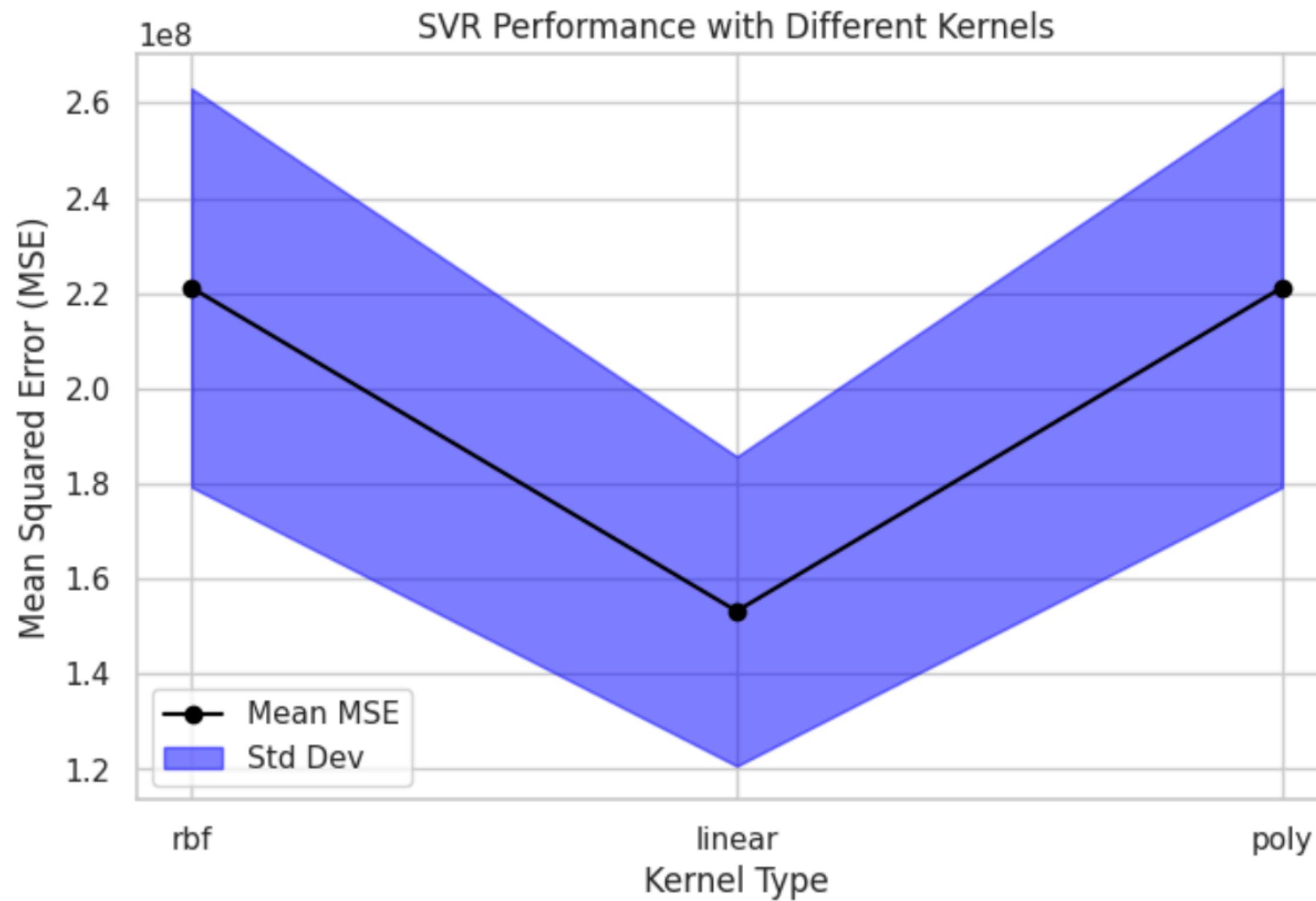
```
reg = RandomForestRegressor(n_estimators=18, random_state=42)
mae_scores = cross_val_score(reg, X, y, cv=kf, scoring=mae_scorer)
mse_scores = cross_val_score(reg, X, y, cv=kf, scoring=mse_scorer)

print(" Random Forest Regressor (RFR - n_estimators=18)")
print(f"MAE promedio: {mae_scores.mean():.2f} ± {mae_scores.std():.2f}")
print(f"MSE promedio: {mse_scores.mean():.2f} ± {mse_scores.std():.2f}")
```

```
Random Forest Regressor (RFR - n_estimators=18)
MAE promedio: 42.97 ± 38.39
MSE promedio: 1531272.40 ± 2512613.54
```

<- Optimizado

SUPPORT VECTOR REGRESSOR



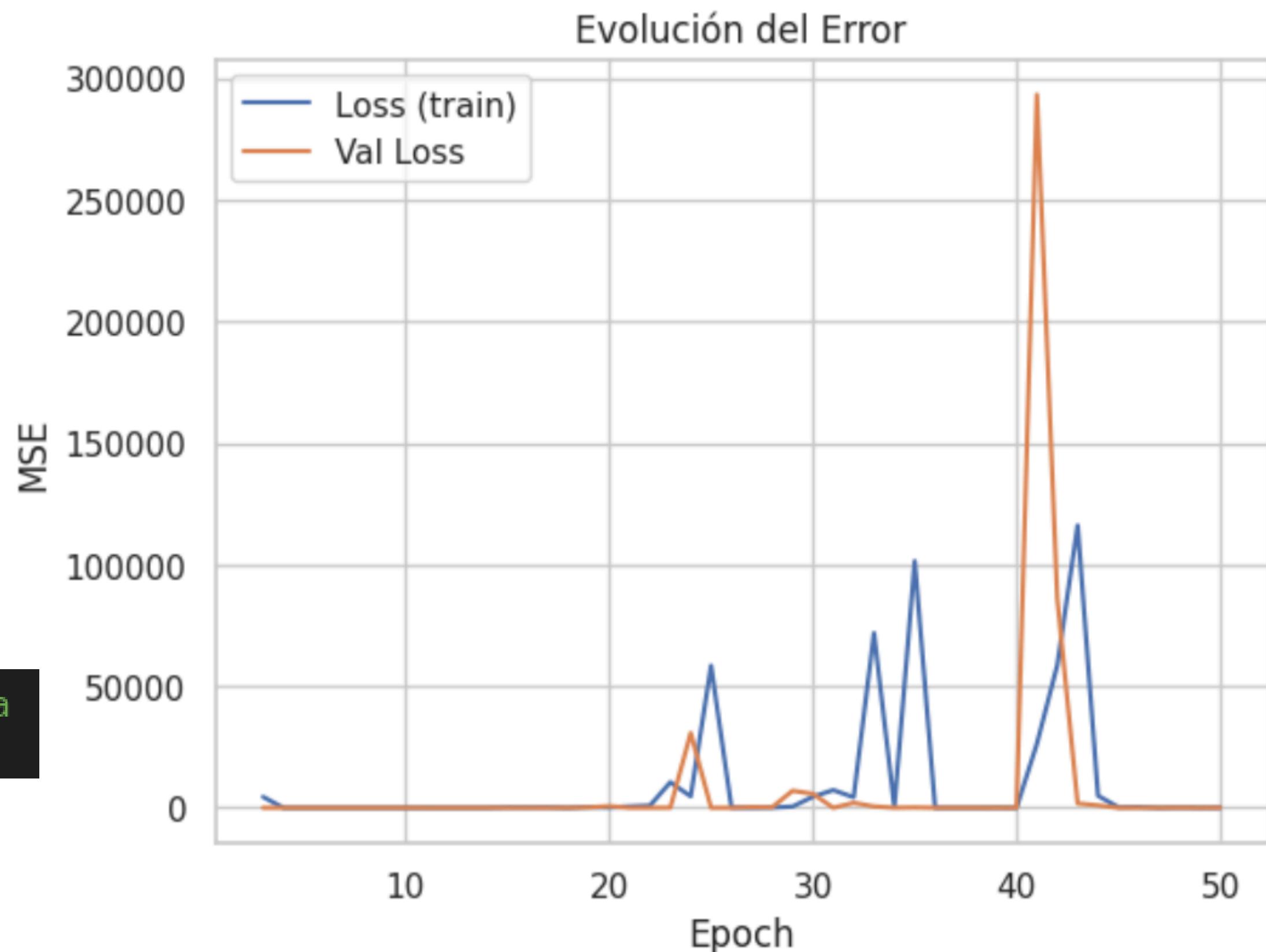
DEEP NEURAL NETWORK MODEL

| Layer (type) |
|-----------------|
| dense (Dense) |
| dense_1 (Dense) |
| dense_2 (Dense) |

| Output Shape | Param # |
|--------------|---------|
| (None, 64) | 73,216 |
| (None, 32) | 2,080 |
| (None, 1) | 33 |

```
# Compilar con tasa de aprendizaje reducida  
optimizer = Adam(learning_rate=0.0005)
```

Deep Neural Network (DNN)
MAE promedio: 3.22 ± 5.12
MSE promedio: 886.40 ± 1763.92



COMPARACIÓN RESULTADOS (REGRESIÓN)

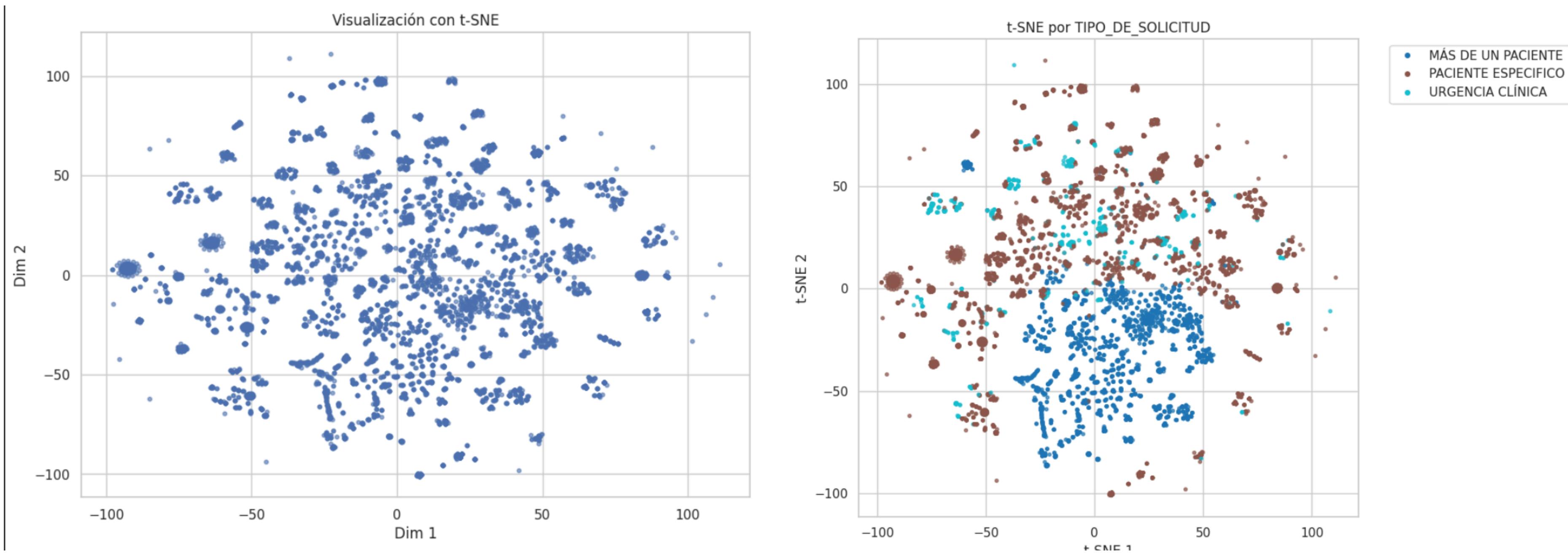
| | Decision Tree | Random Forest | SVR (linear) | DNN |
|-----------|---------------|---------------|----------------|----------|
| MAE | 42,31 | 42,97 | 2.278,19 | 3,22 |
| MAE (std) | 26,17 | 38,39 | 204,36 | 5,12 |
| MSE | 899.300,97 | 1.531.272,40 | 153.201.697,35 | 886,40 |
| MSE (std) | 1.145.168,30 | 2.512.613,54 | 32.513.735,89 | 1.763,92 |

ANÁLISIS DE RESULTADOS

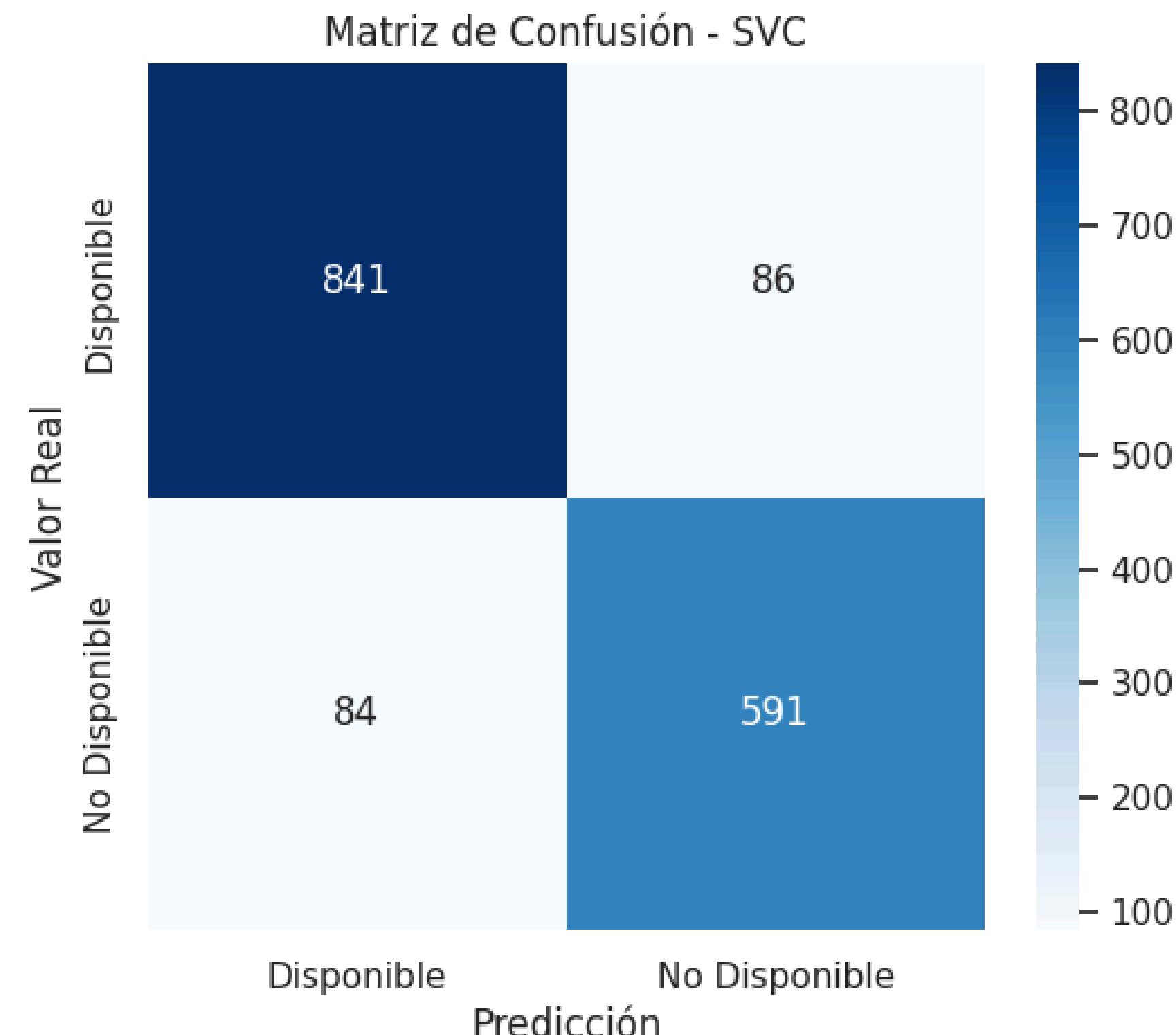
- **Clasificación:** Identifica qué medicamentos podrían faltar.
- **Regresión:** Indica cuán grave sería esa falta.
- **RandomForestClassifier** mostró mejor desempeño para detectar desabastecimiento, gracias a su capacidad para manejar datos con relaciones complejas y variables categóricas.
- **DeepNeuralNetwork** mostró mejor desempeño para estimar la cantidad solicitada gracias a su capacidad de modelar relaciones no lineales y patrones complejos.



T-SNE

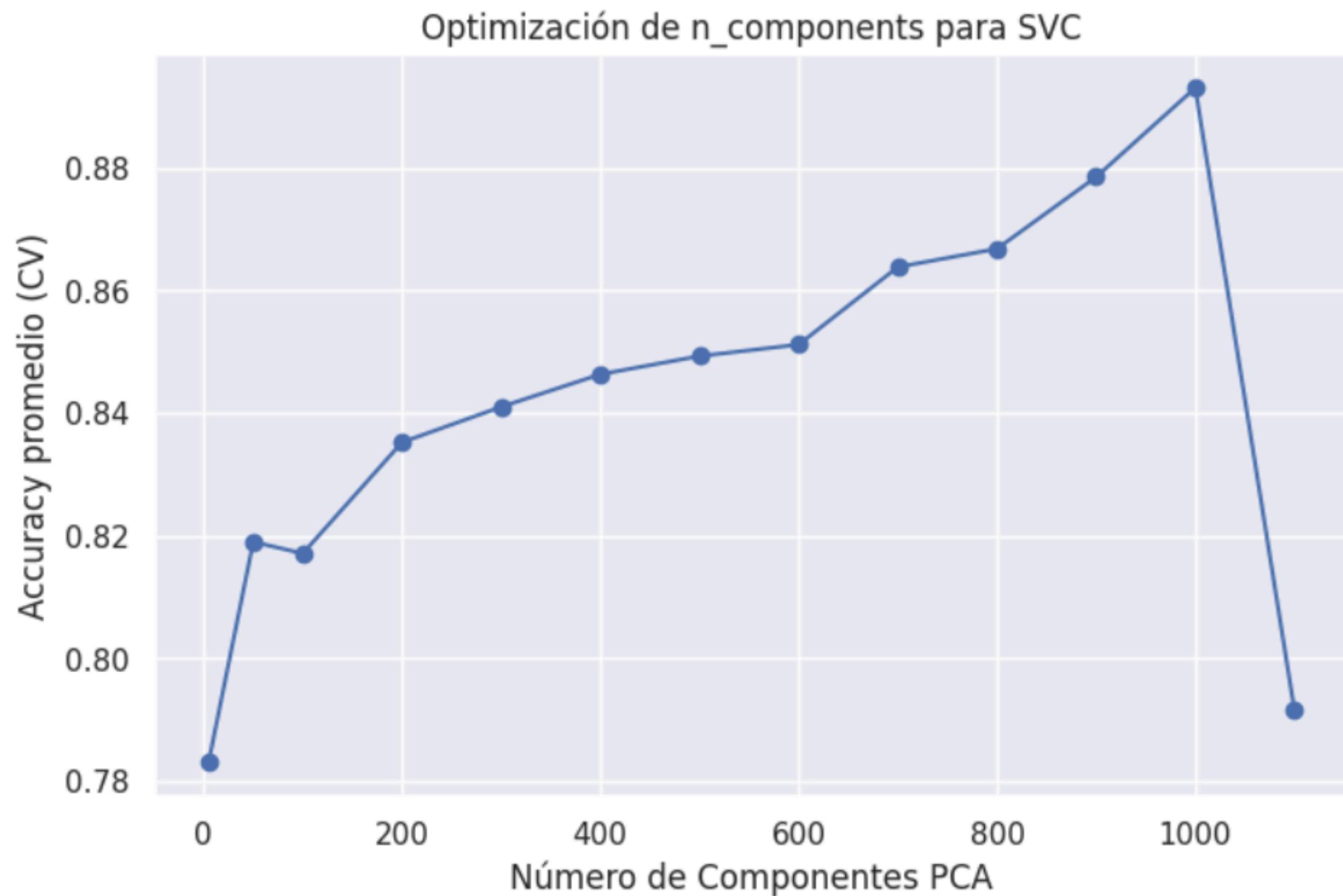


SVC CON T-SNE



Nº COMPONENTES VS ACCURACY (PCA + SVC)

```
X_train shape (6404, 1143)  
X_test shape (1602, )
```



PCA + SUPPORT VECTOR CLASSIFIER (SVC)

SVC sin PCA

| SVC Accuracy: 0.8732833957553059 | | | | |
|----------------------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.88 | 0.90 | 0.89 | 927 |
| 1 | 0.86 | 0.83 | 0.85 | 675 |
| accuracy | | | 0.87 | 1602 |
| macro avg | 0.87 | 0.87 | 0.87 | 1602 |
| weighted avg | 0.87 | 0.87 | 0.87 | 1602 |

Matriz de Confusión - SVC

| Predicción | Disponible | No Disponible |
|---------------|------------|---------------|
| Valor Real | 838 | 89 |
| No Disponible | 114 | 561 |

SVC con PCA

| X_test shape antes de PCA: (1602, 1143) | | | | |
|---|-----------|--------|----------|---------|
| X_test shape despues de PCA: (1602, 1000) | | | | |
| Varianza total de los componentes: 1.0000 | | | | |
| SVC Accuracy: 0.9013732833957553 | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.90 | 0.94 | 0.92 | 927 |
| 1 | 0.91 | 0.85 | 0.88 | 675 |
| accuracy | | | | 1602 |
| macro avg | 0.90 | 0.89 | 0.90 | 1602 |
| weighted avg | 0.90 | 0.90 | 0.90 | 1602 |

Matriz de Confusión - SVC

| Predicción | Disponible | No Disponible |
|---------------|------------|---------------|
| Valor Real | 870 | 57 |
| No Disponible | 100 | 575 |

COMPARACIÓN RESULTADOS (CLASIFICACIÓN)

| | Accuracy | | Precision | Recall | F1-score |
|-------|----------|---|-----------|--------|----------|
| T-SNE | 0.89 | 0 | 0.91 | 0.91 | 0.91 |
| | | 1 | 0.87 | 0.88 | 0.87 |
| PCA | 0.90 | 0 | 0.90 | 0.94 | 0.92 |
| | | 1 | 0.91 | 0.85 | 0.88 |

GRACIAS POR LA ATENCIÓN