Tecnológico de Monterrey

Módulo 2 – Reporte de desempeño de modelo KNN

A01745041 - Ainhara Itzae Mejía Rivera

11 de septiembre del 2023

Inteligencia artificial avanzada para la ciencia de datos IMA
TC3006C.101

Data Set Análisis

El data set utilizado para ambos proyectos es el de *Breast Cancer Wisconsin.* Las características se calculan a partir de una imagen digitalizada de una aspiración con aguja fina (PAAF) de una masa mamaria y describen características de los núcleos celulares presentes en la imagen. Este data set fue elegido porque el cáncer de mama es una enfermedad que se mantiene presente en las mujeres de todo el mundo, y la oportunidad de realizar modelos de aprendizajes que nos permitan predecir su existencia a partir de síntomas, ayudarían a estas mujeres a buscar tratamiento en las etapas tempranas del cáncer, mejorando así sus oportunidades de sobrevivirlo.

Los atributos son los siguientes: ID number, y Diagnosis (M = malignos, B = benignos). Después de estos, se incluyen 10 variables que describen los núcleos celulares; radius (mean of distances from center to points on the perimeter), texture (standard deviation of gray-scale values), perimeter, área, smoothness (local variation in radius lengths), compactness (perimeter^2 / area - 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, y fractal dimension ("coastline approximation" - 1). La distribución de las clases es la siguiente: 357 benignos, 212 malignanos.

A partir de esto, se realizó un estudio general de los datos y se graficaron las variables para poder analizar a profundidad el comportamiento de los datos. Primero que nada, se realizó un df.count() para poder identificar si habían valores faltantes, y de esta manera pudimos verificar que todas las variables contaban con todos los datos, es decir, no habían datos faltantes.

Después, se realizaron gráficas de distribución para lograr apreciar los valores dentro de estas variables.

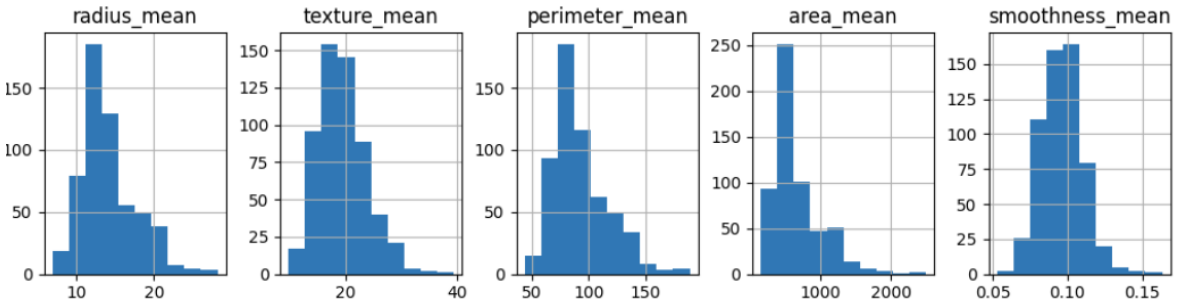

Figura I: Gráfica de distribución de las variables radius_mean, textura_mean, perimeter_mean, área_mean y smoothness_mean.

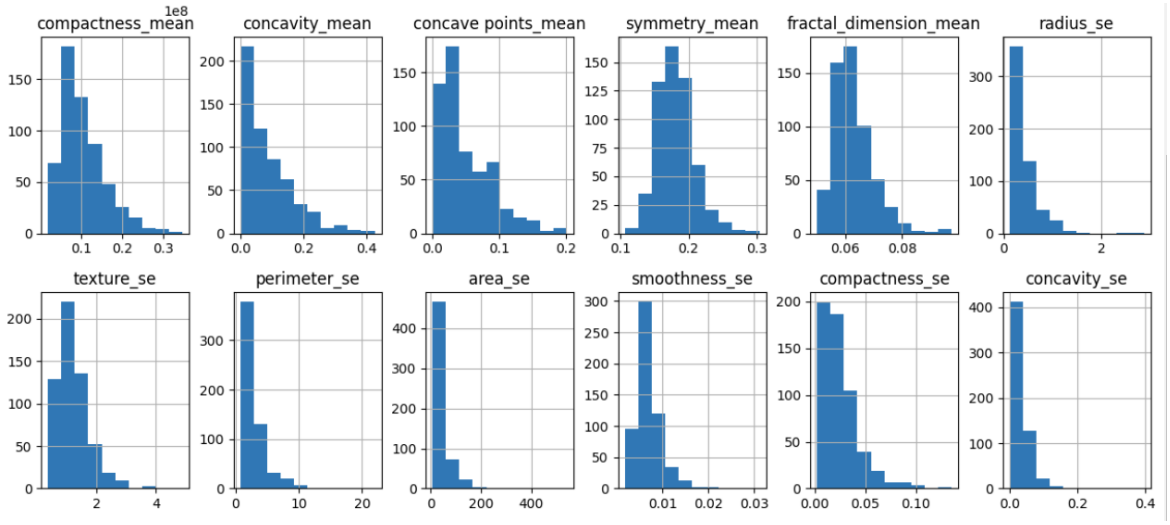

Figura II: Gráficas de distribución de los valores existentes dentro de las variables compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean, radius_se, textura_se, perimeter_se, área_se, smoothness_se, compactness_se y concavity_se.

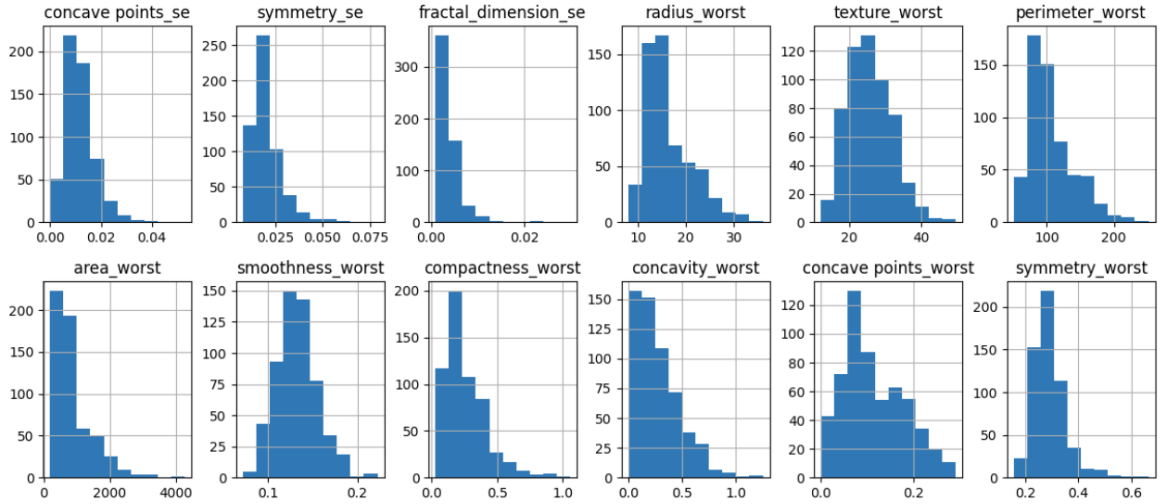

Figura III: Gráficas de distribución de los valores dentro de las variables concave_points_se, symmetry_se, fractal_dimension_se, radius_worst, textura_worst, perimeter_worst, área_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_ worst, symmetry_worst.

Modelo Utilizado: KNN

El modelo de KNN es un algoritmo de clasificación, y utiliza la proximidad para hacer clasificaciones y/o predicciones sobre la agrupación de los datos. Para este modelo, se pueden utilizar diferentes métricas de distancia que podrían impactar su rendimiento, y esto pasa ya que el modelo de KNN busca obtener las distancias y el vecino más cercano de cada consulta. [4]

Distancia Euclidiana:

La distancia euclidiana es una métrica que calcula la longitud de la línea recta que conecta el punto de consulta con el otro punto que se está evaluando. Esto se logra mediante la utilización de la fórmula específica que se describe a continuación:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$ [4]

Distancia Manhattan:

La distancia Manhattan calcula la magnitud del valor absoluto entre dos puntos dados. También es conocida como "distancia de taxi" o "distancia de cuadra de la ciudad", ya que se asemeja a la manera en que uno se desplaza por las calles de la ciudad de Manhattan. [4]

Distancia Minkowski:

La distancia Minkowski engloba tanto las métricas de distancia Euclidiana como la de Manhattan. El parámetro "p" en la fórmula que se muestra a continuación brinda la flexibilidad para generar otras métricas de distancia. Cuando "p" tiene un valor de dos, obtenemos la distancia euclidiana, mientras que con "p" igual a uno, obtenemos la distancia de Manhattan:

$$\left(\sum_{i=1}^{n}|x_i - y_i|\right)^{1/p}$$ [4]

Distancia Haming:

La distancia Haming se emplea generalmente con vectores que contienen valores booleanos o cadenas de caracteres, detectando los puntos en los cuales los vectores difieren entre sí. Por este motivo, se le conoce también como la métrica de superposición. Esta relación se puede expresar utilizando la siguiente fórmula:

$$D_H = \left(\sum_{i=1}^{k}|x_i - y_i|\right)$$ [4]

Resultados del Modelo KNN implementado:

El modelo parece presentar un caso leve de overfitting, a continuación se analizará a detalle este diagnóstico. Además, se realizó una gráfica para demostrar el sesgo y el aprendizaje del modelo. La gráfica demuestra el puntaje de validación dependiendo del tamaño del conjunto de datos utilizados para entrenarlo. Como podemos observar, con una muestra de menos de la mitad parece tener una mejor precisión, y al aumentar la muestra, se estabiliza en un nivel más bajo de precisión.

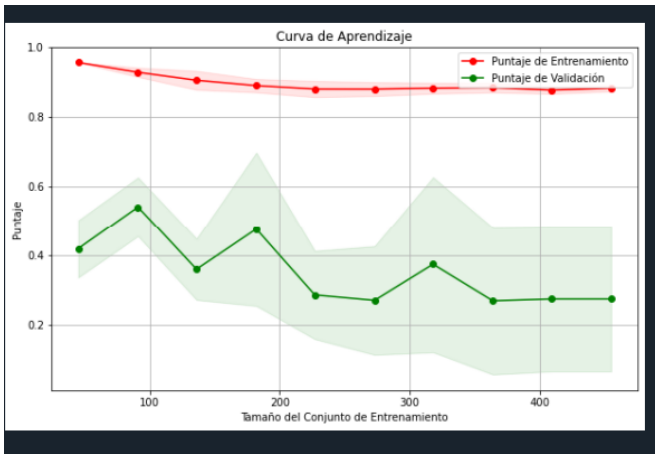

Imagen III: Gráfica que demuestra la curva de aprendizaje, el puntaje de validación, y entrenamiento dependiendo del tamaño de la muestra.

Esta gráfica nos ayuda a visualizar cómo el rendimiento del modelo varía con el tamaño del conjunto de entrenamiento. También sirve para identificar si el modelo se beneficia de más datos o si está alcanzando un límite en su capacidad de aprendizaje (cómo es el caso en este modelo). Si las curvas de entrenamiento y validación están convergiendo hacia un puntaje alto, puede ser señal de que el modelo está aprendiendo correctamente, pero como podemos ver, ese no es el caso en este modelo.

Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation):

Para este modelo, se partieron los datos en el usual 80% dedicados para el entrenamiento del modelo, y el 20 % restante se dividieron a la mitad para la validación y la prueba del modelo. Para visualizar esto, a continuación recopilo las partes del código relevantes, y la gráfica final que demuestra cómo se dividen los datos:

```
# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Imagen IV: Captura del código que demuestra cómo los datos fueron divididos en 80% y 20% para el entrenamiento y prueba, respectivamente.

```
# Dividir el conjunto de prueba en prueba y validación (50% prueba, 50% validación)
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.5, random_state=42)

# Evaluar el modelo en el conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, clf.predict(X_train))

# Evaluar el modelo en el conjunto de prueba
test_accuracy = accuracy_score(y_test, clf.predict(X_test))

# Evaluar el modelo en el conjunto de validación
val_accuracy = accuracy_score(y_val, clf.predict(X_val))
```

Imagen V: Captura del código que divide el 20% de datos destinados para la prueba en prueba y validación. También demuestra cómo se evaluó el modelo con el conjunto de entrenamiento, prueba y validación.

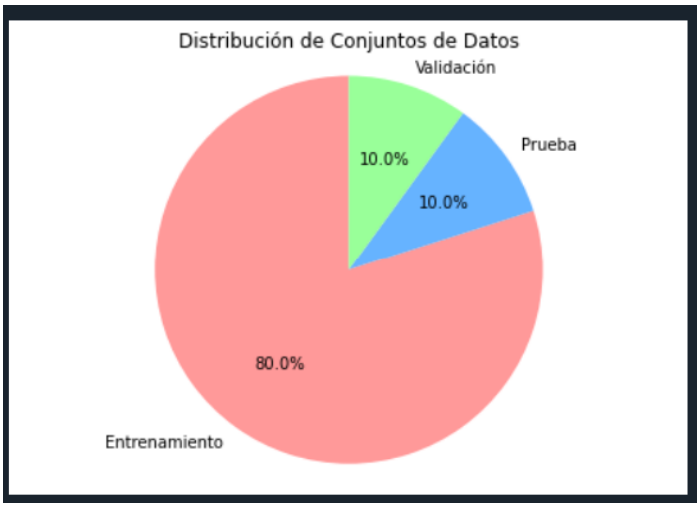

Imagen VI: Gráfica de pie que demuestra el porcentaje en el que fueron divididos los datos.

Los datos son divididos de esta manera ya que así nos aseguramos de que aprenda del mayor número de datos disponibles. Sin embargo, esto podría causar un grado de sesgo hacia cierta clase, y cómo se mostró al inicio del documento, el número de datos de células benignas son mayor al número de células malignas.

Diagnóstico y explicación el grado de bias o sesgo (bajo medio alto):

Para empezar la investigación del sesgo del modelo, debemos de empezar analizando la base de datos que le dimos. Cómo acabamos de mencionar, la base de datos ya contiene un sesgo pequeño hacia las células benignas. Y de igual manera, al darle un porcentaje alto de datos para entrenar al modelo, se puede crear un overfitt ya que es posible que el modelo aprenda detalles específicos de los datos de entrenamiento que no son generalizables a nuevos datos.

Las métricas obtenidas fueron las siguientes:

```
Confusion Matrix:
          Predicted 0  Predicted 1
Actual 0           63            8
Actual 1           20           23

Precision: 0.7419354838709677
Recall: 0.5348837209302325
Accuracy: 0.7543859649122807
Specificity: 0.8873239436619719
Precisión en el conjunto de entrenamiento: 0.8417582417582418
Precisión en el conjunto de prueba: 0.7192982456140351
Precisión en el conjunto de validación: 0.7894736842105263
```

Imagen VII: Métricas calculadas del modelo (Accuracy, Confusion Matrix, Precision, Recall y Specificity)

A partir de estas métricas, podemos concluir que el modelo presenta un sesgo moderado, acercándose a un sesgo alto, ya que presenta una facilidad en predecir los valores de 0, y una dificultad en predecir los valores de 1.

Esto se puede verificar analizando cada una de las métricas a profundidad:

La precisión en la clase 1 es de 74.1 %, y esto sugiere que el modelo puede tener una tendencia a predecir la clase 1 con un nivel bueno de confianza. Por otro lado, el recall en la clase 1 es del 53%, lo que indica le es más difícil capturar todas las muestras de la clase 1.

El accuracy tiene un porcentaje de 75%, lo que es un buen indicador del rendimiento global del modelo, sin embargo, comparándolo con el desempeño de las demás métricas, no nos da una imagen clara del sesgo en los datos, y esto se puede demostrar con la especificidad , la cual es del 89%, lo que indica que el modelo es bastante bueno para identificar las muestras de la clase 0.

Diagnóstico y explicación el grado de varianza (bajo medio alto):

Para este análisis, debemos de tomar en cuenta las métricas de la Imagen VII. Cómo sabemos, el grado de varianza en un modelo mide qué tan fácil se adapta un modelo a las fluctuaciones de los datos de entrenamiento. Analizando las métricas, podemos concluir que el modelo tiene un grado de varianza medio, ya que la precisión y el recall están relativamente equilibrados, y la especificidad es alta. Sin embargo, el recall que es un poco más bajo, pero no tan bajo como para decir que existe una varianza alta en el modelo.

También debemos de fijarnos en la precisión del conjunto de entrenamiento, ya que es más alto que la precisión en el conjunto de prueba, sin embargo, su valor no es tan alto como para que decir que hay un overfitt.

Diagnóstico y explicación el nivel de ajuste del modelo (underfitt fitt overfitt):

A partir de las métricas obtenidas, podemos concluir que el modelo parece tener un nivel bajo de overfitt, más cercano a un fitt. Esto se debe a que las métricas obtenidas indican que el modelo no tiene una precisión extremadamente alta en los datos de entrenamiento, pero tampoco está perdiendo un número significativo de casos positivos en el conjunto de prueba.

Como se mencionó antes, el único “problema” es el recall, ya que sólo demuestra una precisión por apenas encima de la mitad. Este es un problema que se puede solucionar con tan sólo la implementación de técnicas de regularización, las cuales serán implementadas a continuación.

Ajustes Finales:

Algunos los ajustes que se le pueden implementar son las técnicas de regularización. Ya que el data set tiene un poco de sesgo, la técnica que considero que serviría más es la

técnica del clasificador KNeighborsClassifier con el parámetro weights establecido en 'distance', logrando darle más peso a los vecinos más cercanos.

```
# Inicializar y entrenar el modelo KNN con regularización L2
clf = KNeighborsClassifier(n_neighbors=5, weights='distance')  # Añade 'weights='distance''
clf.fit(X_train, y_train)
```

Imagen VIII: Captura del código en dónde se implementa la técnica de regularización

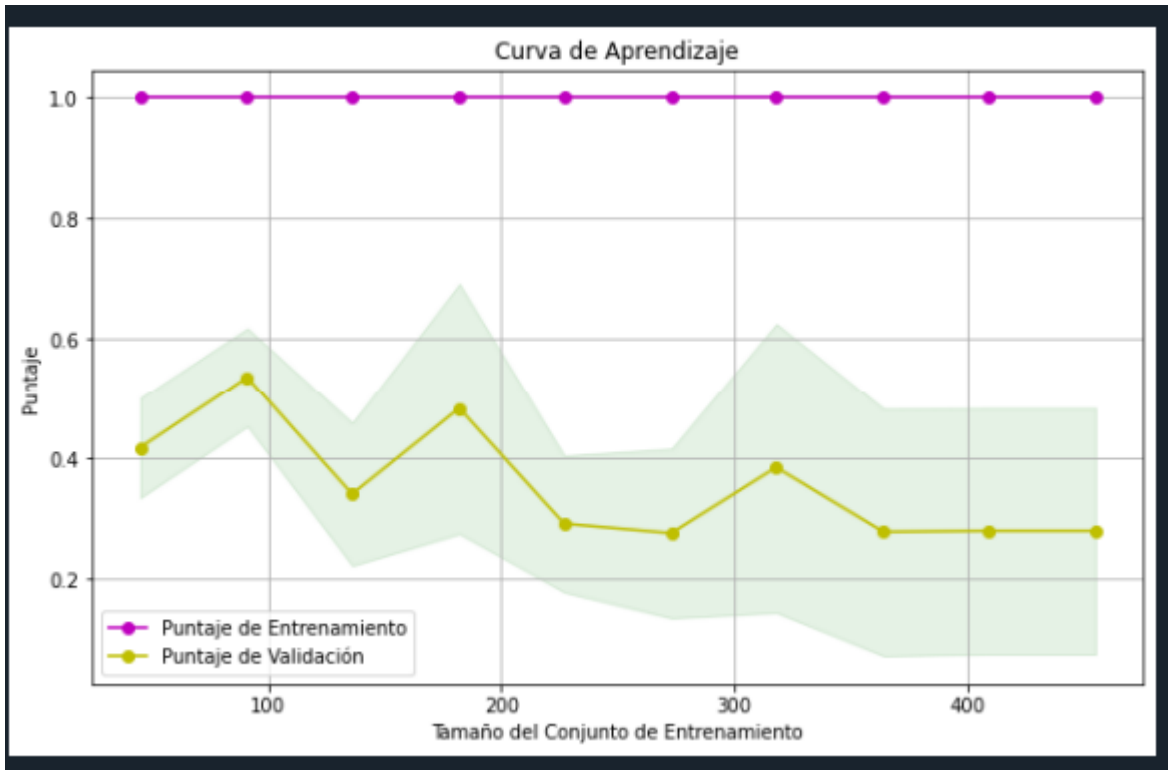

Imagen IX: Gráfica que demuestra la curva de aprendizaje estática y la variación de los puntajes dependiendo del tamaño de la muestra de entrenamiento.

A partir de esta gráfica, podemos comparar con la primera, y obtenemos que llegó a su capacidad de aprendizaje de manera rápida, lo que respalda mi conclusión final de este modelo.

```
Confusion Matrix:
          Predicted 0  Predicted 1
Actual 0           64            7
Actual 1           17           26

Precision: 0.7878787878787878
Recall: 0.6046511627906976
Accuracy: 0.7894736842105263
Specificity: 0.9014084507042254
Precisión en el conjunto de entrenamiento: 1.0
Precisión en el conjunto de prueba: 0.7368421052631579
Precisión en el conjunto de validación: 0.8421052631578947
```

Imagen X: Resultados de las métricas después de aplicarle la técnica de regularización.

Comparando las métricas nuevas, podemos concluir que hubo un mejoramiento notorio en el recall, y podemos verificarlo en la clase 1, la cual tuvo mejores predicciones en valores que sí eran clase 1. Sin embargo, analizando los resultados de las demás métricas, puedo llegar a la conclusión de que existe un overfitt en este modelo, ya que la precisión en el conjunto de entrenamiento es del 100%, lo que significa que se aprendió las características de estos datos y probablemente no sería un buen modelo cuando se le introduzca más data nueva.

Implementación 2: StandardScaler

```
scaler = StandardScaler()
X = scaler.fit transform(X)
```

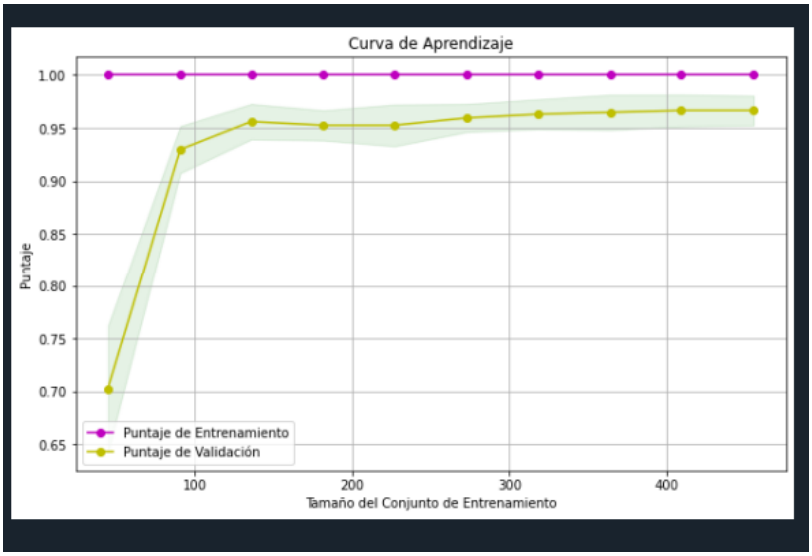

Imagen XI: Gráfica de la curva de aprendizaje después de aplicarle la técnica de regularización.

```
Confusion Matrix:
          Predicted 0  Predicted 1
Actual 0           68            3
Actual 1            3           40

Precision: 0.9302325581395349
Recall: 0.9302325581395349
Accuracy: 0.9473684210526315
Specificity: 0.9577464788732394
Precisión en el conjunto de entrenamiento: 1.0
Precisión en el conjunto de prueba: 0.9122807017543859
Precisión en el conjunto de validación: 0.9824561403508771
```

Imagen XII: Resultados de las métricas después de aplicarle la técnica de regularización.

Comparando las métricas nuevas, podemos concluir que hubo un mejoramiento notorio en el recall, y podemos verificarlo en la clase 1, la cual tuvo mejores predicciones en valores que sí eran clase 1. De nuevo, analizando los resultados de las demás métricas, puedo llegar a la conclusión de que existe un overfitt en este modelo, ya que la precisión en el conjunto de entrenamiento es del 100%, lo que significa que se aprendió las características de estos datos y probablemente no sería un buen modelo cuando se le introduzca más data nueva.

Aunque, en la gráfica de puntos podemos ver que finalmente obtenemos los resultados deseados, demostrando que el modelo aprende de manera correcta.

Implementación 3: SelectKBest

```
Confusion Matrix:
          Predicted 0  Predicted 1
Actual 0           69            2
Actual 1            1           42

Precision: 0.9545454545454546
Recall: 0.9767441860465116
Accuracy: 0.9736842105263158
Specificity: 0.971830985915493
Precisión en el conjunto de entrenamiento: 1.0
Precisión en el conjunto de prueba: 0.9649122807017544
Precisión en el conjunto de validación: 0.9824561403508771
```

Imagen XIV: Resultados de las métricas después de aplicarle la técnica de regularización.

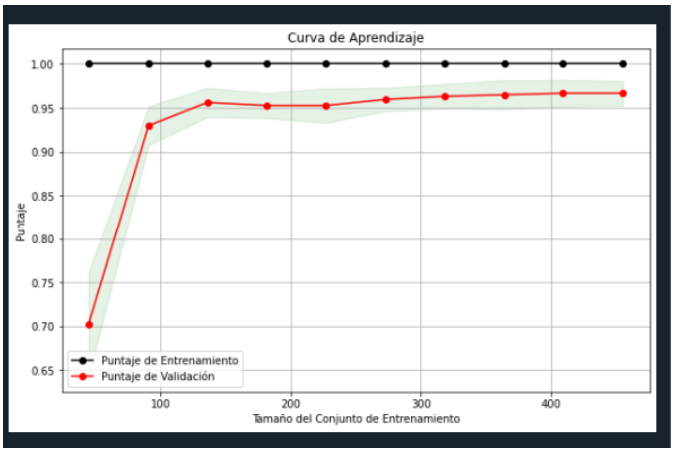

Imagen XV: Gráficas después de aplicarle la técnica de regularización.

Comparando las métricas nuevas, podemos concluir que hubo un mejoramiento notorio en todos los rubros, pienso que es el que mejor resultado que tuvo. De nuevo, analizando los resultados de las demás métricas, puedo llegar a la conclusión de que existe un overfitt en este modelo, ya que la precisión en el conjunto de entrenamiento es del 100%, lo que significa que se aprendió las características de estos datos y probablemente no sería un buen modelo cuando se le introduzca más data nueva.

De nuevo, la gráfica final demuestra lo dicho, y también visibiliza que no existen más cambios que se le puedan implementar ya que ya llegó al mejor valor de aprendizaje.

Comparación Final:

El mejor modelo fue claramente el último, aunque por los valores de las métricas, se podría alegar que el modelo tiene overfitt, y que en realidad aprendió los valores del data set de entrenamiento, dado el 100% de precisión obtenido. El penúltimo modelo también parece bastante bueno, pero tiene el mismo error, por lo que puedo concluir que a pesar de tener un nivel bajo de recall, el primer modelo es el mejor de estos.

Referencias:

[1] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. In On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings (pp. 986-996). Springer Berlin Heidelberg.

[2] Mani, I., & Zhang, I. (2003, August). kNN approach to unbalanced data distributions: a case study involving information extraction. In Proceedings of workshop on learning from imbalanced datasets (Vol. 126, No. 1, pp. 1-7). ICML.

[3] Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. Neurocomputing, 195, 143-148.

[4] ¿Qué es KNN? | IBM. (2023). Ibm.com. https://www.ibm.com/mx-es/topics/knn