UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Reporte Final

Inteligencia Artificial

24/05/2024

Clasificando Radiografías de Alzheimer usando SVM

Equipo Séú Séú Pu+:

Ana Cristina Cuevas García

Morán Pérez Derek Saúl

Emiliano Villicaña García

PROFESOR: Cecilia Reyes Peña

AYUDANTE: Tania Michelle Rubí Rojas

Índice

Motivación:

Los miembros del equipo Séú Séú Pu+ nos planteamos crear un clasificador de imágenes usando los conocimientos adquiridos a lo largo del curso de Inteligencia Artificial, la manera en que lo haremos es tomando los pixeles de las imágenes para poder identificarlos y clasificar en alguna de las categorías que se tengan, el tipo de aprendizaje a usar será supervisado, debido a que contamos con las categorías definidas para cada imagen.

Pero, ¿Qué imágenes usaremos?, en este caso haremos un clasificador de radiografías sobre alzheimer, una enfermedad que actualmente se ha hecho más presente en la vida de las personas, claramente existen médicos dedicados a esta área que pueden identificar las radiografías de manera consistente, este proyecto no tiene la intención de suplantar a un médico ni nada por el estilo, únicamente podría llegar a ser un apoyo al médico para poder dar un análisis medianamente rápido. Es por esto que nos planteamos poder tener cerca de un 90% de efectividad del modelo al momento de clasificar, consideramos que es una buena meta, que tendrá resultados considerablemente buenos y creemos que podemos conseguirlo sin problemas.

Con esto en mente y ya aclarado, comencemos con el proyecto.

Introducción

Primero que nada, hablemos sobre: ¿Qué es el alzheimer?

El Alzheimer es una enfermedad (demencia) progresiva que afecta la memoria y otras funciones mentales, donde las conexiones de las células cerebrales mueren y se degenera, esta enfermedad no tiene cura, y aún no se conoce bien el qué causa esta enfermedad, pero hay un pequeño lado bueno, si bien la enfermedad no tiene cura, si es detectada a tiempo pueden haber medicamentos o técnicas de ejercicios que evitan el desarrollo de la enfermedad, lo que puede hacer que una persona pueda vivir más tiempo sin que la enfermedad se desarrollé, y todavía mejor, puede ser que nunca se desarrolle.

Esta enfermedad fue la quinta principal causa de muerte en personas mayores a 65 años en el año de 2021 en los Estados Unidos.

Además, estadísticamente se ha visto un aumento en los casos de esta enfermedad alrededor del mundo, no es porque sea contagiosa, o porque está mutando de manera que afecta a más personas, la razón es simple, el alzheimer comienza a aparecer en las personas usualmente mayores a 65 años, la cuestión es que con el tiempo, la esperanza de vida de las personas ha aumentado hasta los 70 - 80 años, por lo que la enfermedad tiende a ser más presentes, de igual manera, se presenta más en las mujeres ya que su esperanza de vida es mayo a la de los hombres, de nuevo, esto no indica que por ser mujer tengas riesgo de contraer alzheimer, pero sí a desarrollarlo ya que vives más.

Objetivo del proyecto, ¿Por qué clasificar imágenes de Alzheimer?

El Alzheimer tiene 3 etapas principales: demencia leve, demencia moderada, demencia avanzada, realmente no hay una diferencia clave entre ellas, o un límite exacto.

Como se mencionó anteriormente, existen ejercicios y medicamentos que retrasan el avance progresivo de la enfermedad, estos métodos son más eficaces en etapas tempranas de la enfermedad.

El propósito del proyecto es poder identificar si una persona no tiene presencia de esta demencia, leve, moderada o avanzada. El objetivo principal es poder tener una buena clasificación para las etapas tempranas, puesto que es aquí donde el paciente debe comenzar a tomar acciones para evitar un avance en la enfermedad, antes de que empeore y los costos de tratamiento aumentan de manera exponencial, lo que puede ayudar al paciente a reaccionar a tiempo.

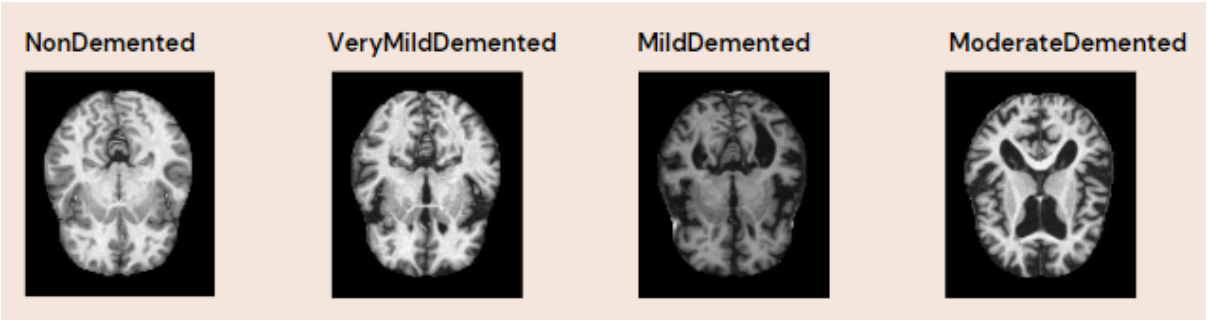
¿Cómo lo haremos?

Encontramos una base de datos en Kaggle, que recopila imágenes de radiografías cerebrales etiquetadas en 4 categorías, las cuales son (yendo de no presencia a

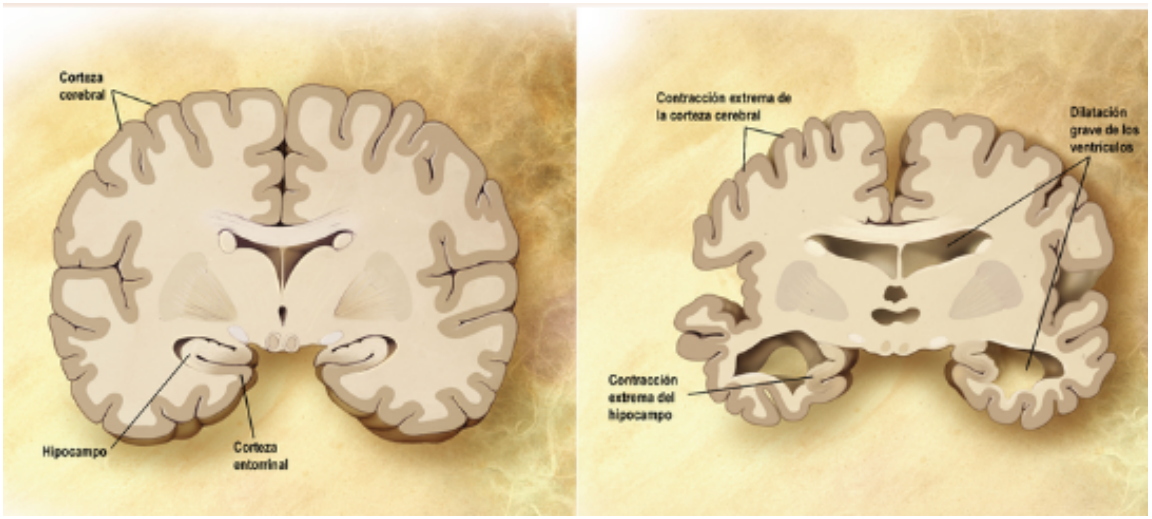
cuando la enfermedad ya tiene mayor presencia) no demencia, demencia muy baja, demencia media y demencia moderada. Esta base de datos cuenta con dos archivos, uno para la clase train y otro para test, debido al costo computacional que lleva crear un proyecto de este tipo, nosotros sólo tomaremos la base train con más de 5,000 datos repartidos en cada una de las categorías para poder resolver este problema, es decir, en esta base es donde haremos el split para dividir el conjunto de entrenamiento y prueba.

La base se encuentra en el siguiente link:
https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

A continuación un ejemplo de cómo es que se ven estas radiografías:



Podemos notar que los "bordes" y el "centro" del cerebro se ven más gastados conforme la enfermedad tiene un avance, estas son las partes del cerebro que parecen mostrar donde afecta Alzheimer, por lo que nuestro modelo debe de notar bien estas partes para clasificar correctamente, la siguiente imagen muestra los nombres de las zonas del cerebro afectadas y que es lo que les ocurre exactamente.

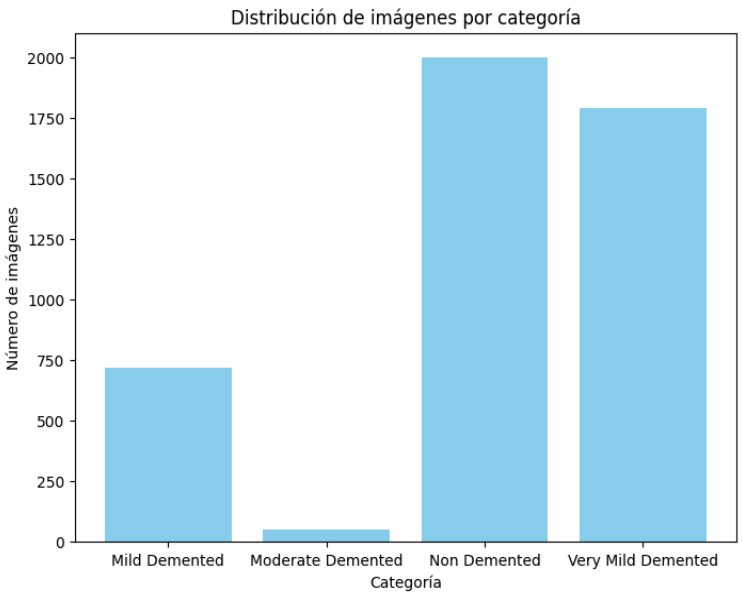


Enfermedad de Alzheimer. (2023). Wikipedia. https://acortar.link/gQkZI

Pre-procesamiento de Datos

Como en todo proyecto de este tipo, primero debemos analizar los datos y quizás hacer transformaciones, para que este proyecto resulte exitoso, primero debemos de tener los datos cuadrados, verificar que el brillo sea igual y ver que estén en el mismo tono de color, puesto que si alguna de las imágenes no cumple esto, se van a presentar complicaciones.

Gráfica de distribución.



Primero veamos la distribución de los datos, para observar la cantidad que tenemos por cada categoría.

Podemos notar un gran desbalance de los datos, lo que podría causar problemas de clasificación, nosotros vamos a resolver esto usando una división estratificada de datos, pero también existen técnicas para aumentar la cantidad de datos usando datos sintéticos, debido al poco o nulo conocimiento que

tenemos de este método, vamos a obviar esta opción e iremos directamente con el método de división estratificada para entrenar el modelo.

Como mencionamos, se tienen 4 categorías, en relación a las 3 categorías de Alzheimer mencionadas anteriormente estas se van a reflejar de la siguiente manera:

Non Demented -> No hay Alzheimer
Very Mild Demented -> Primera Etapa de Alzheimer (la de mayor interés en este proyecto)
Mild Demented -> Etapa media de Alzheimer
Moderate Demented -> La etapa más avanzada que tenemos en los datos

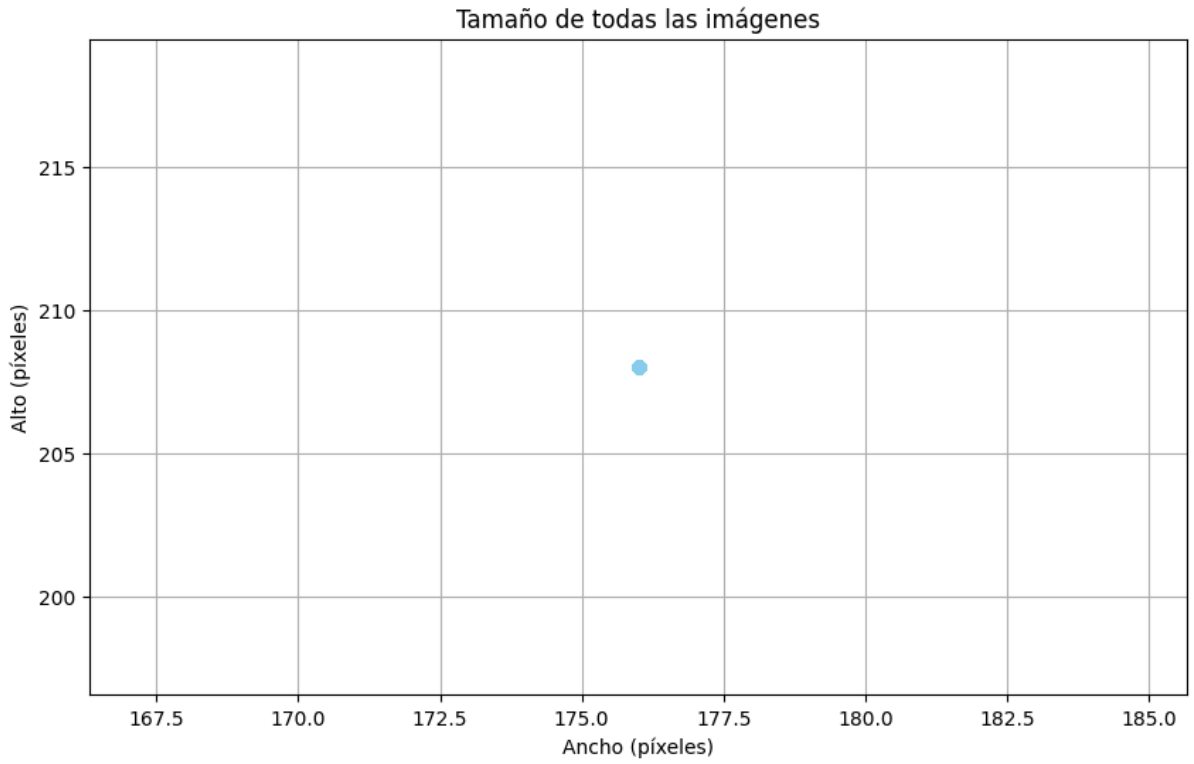
El desbalance que se genera puede ser inducido de manera lógica: para cada etapa mayor de Alzheimer menos datos se tienen ya que un paciente con Alzheimer avanzado no suele sacar muchas radiografías o las personas llegan a perecer en estas etapas.

Tamaño de las imágenes

Una de las transformaciones esenciales para los datos es observar qué tamaño tienen y de ser posible, re-escalarlo para tener imágenes cuadradas, ya que la mayoría de veces es preferible hacerlo así para que el modelo identifique mejor las imágenes.

En esta sección vamos a ver el tamaño en pixeles de cada una de las radiografías, recordemos que necesitamos que sean cuadradas, pero a su vez, hacerlo sin que se recorte una parte del cerebro en la radiografía, lo que haría que se pierda información posiblemente relevante.

Nosotros usamos un gráfico para poder visualizar cual es el tamaño de todas las imágenes en lo que respecta a la cantidad de píxeles de largo y ancho, obteniendo lo siguiente:



Afortunadamente, todas nuestras imágenes tienen las mismas dimensiones, teniendo 176 píxeles de largo por 208 de largo, es decir se tienen unas imágenes rectangulares, para facilitar el proceso, haremos un recorte de 16 pixeles de arriba a abajo y de 16 de abajo arriba, esta cantidad es elegida de esta manera ya que nos permite recortar la imagen y tenerlo de forma cuadrada, pero mejor aún, al hacerlo así, no estamos recortando la imagen del cerebro, si no que se mantiene intacto, veamos un ejemplo:

Imagen antes del recorte:

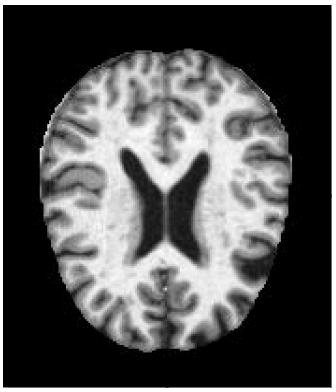
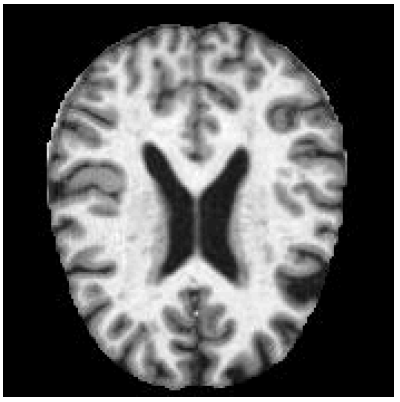


Imagen después del recorte:

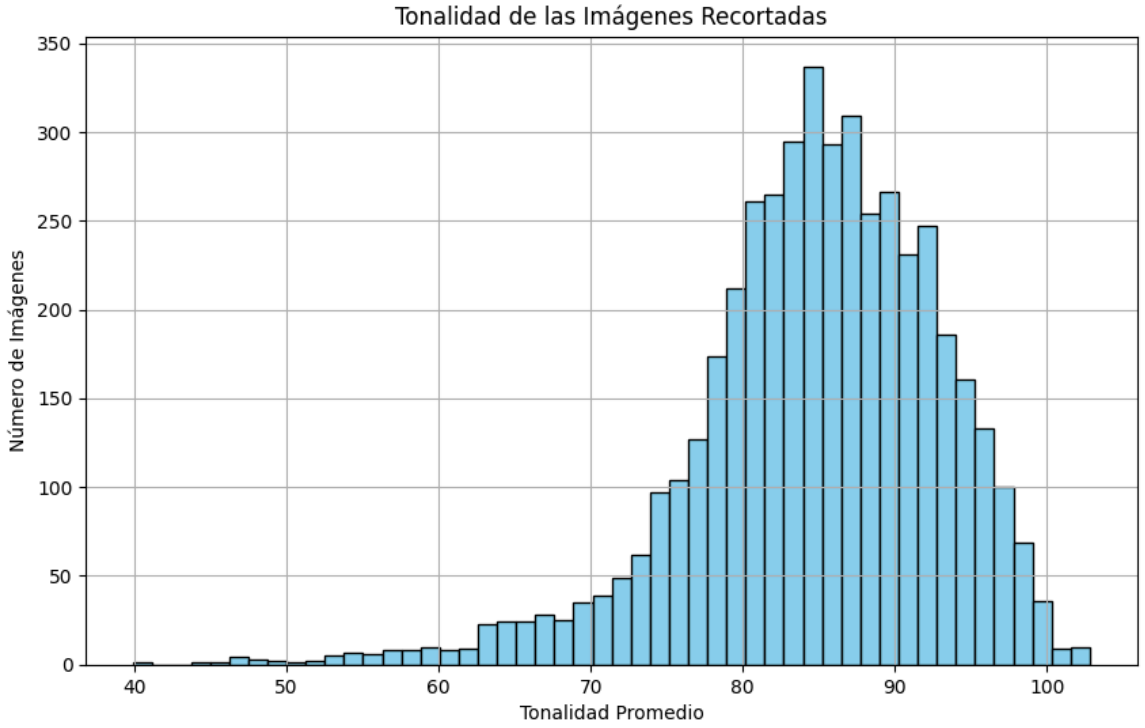


Como podemos observar, en esta imagen (y en todas las demás de cada categoría), el cerebro queda justo en medio y no se ve recortado o afectado por esta primera transformación, por lo que resultó exitosa y lo replicamos en todos los demás datos.

Patrón de Colores en las imágenes

Necesitamos ver si todas las imágenes tienen una tonalidad parecida, puesto que esto nos va a permitir tener un proceso de clasificación más coherente y que pueda ser más confiable para este tipo de problemas sobre clasificar radiografías, ya que la confianza en el modelo debe ser lo mejor posible, aunque claro, al hacer esto vamos a perder variabilidad en los patrones de color si queremos ingresar otros datos, pero usualmente las radiografías siempre suelen tener el mismo tono, por lo que creemos esto será una buena idea e implementación.

Esto lo haremos usando el sistema de tonalidad de grises, ya que nuestras radiografías están en tonos de grises, donde se usa un solo canal de color para representar la intensidad, yendo ed 0 a 255, donde el 0 es negro absoluto y 255 blanco absoluto, además tener todo en esta escala ayuda a no cargar tanto costo computacional.



Podemos ver que la mayoría de las imágenes tiene un tono de grises parecido, cerca del 80 en el valor de escala de grises, no vamos a hacer modificaciones para tener todas del mismo tono, puesto que estaríamos quitando mucha generalidad al

modelo, dejándolo así podemos tener un poco más de variabilidad en los datos que nos puede ayudar a obtener mejores resultados.

Elección del Modelo.

Ya tenemos los datos con su primera preprocesación, la general que es para poder implementarlo a cualquier tipo de modelo, pero bien, ahora la pregunta es, ¿Qué modelo usar?

Bueno, sabemos de primera mano que por la cantidad de datos y etiquetas lo mejor a usar podrían ser Redes Neuronales, usando deep learning, pero este proceso es muy tardado y conlleva alto costo computacional, es por esto que veremos si podemos escoger otro modelo que sea más simple en implementar y ver si tiene buenos resultados.

Proponemos el SVM, este modelo es eficaz cuando se tienen grandes dimensiones de datos, en este caso tenemos imágenes, donde las dimensiones son los pixeles y tenemos 176x176, es por esto que hemos optado por usar este modelo, más aún haremos como tal dos modelos, primero uno donde probaremos el modelo sin realizar una disminución de dimensiones, y luego uno donde aplicaremos PCA para ver si hay alguna mejora, para cada uno haremos ajustes de hiper parámetros necesarios que nos permitan obtener el mejor modelo.

Implementación del Modelo: SVM sin PCA

Más Procesamiento

Una vez que ya vimos que se puede hacer uso de un modelo SVM haremos la implementación usando este modelo, para esto, debemos de normalizar las imágenes para empezar y para poder hacer un mejor uso de este modelo, es decir, se normalizan los valores de los pixeles de cada imagen entre 0 y 1, primero realizaremos esta normalización.

Normalización: Colocamos los datos en una escala similar, teniendo las características de los pixeles en rangos comparables, permitiendo una mejor

búsqueda de soluciones Nosotros restamos la media y dividimos por la desviación estándar.

Aplanamiento: Nuestras imágenes son de 176 x 176 pixeles, pero el SVM necesita entrada de un vector unidimensional, por lo que tendremos un vector de 30,976 valores, concatenando todas las filas o columnas Cada uno representará una característica, como color o textura. De igual forma la escala de grises aquí ayudó mucho, ya que si fuera RGB se tendría un vector de más de 90K datos, la escala de grises nos ayudó a tener un vector más simplificado.

División de Datos

Finalmente tenemos todos los datos ya procesados (al menos lo necesario para esta sección), así que ahora necesitamos dividir los datos en conjunto de entrenamiento y prueba. La división de los datos se hizo tomando 70% para el conjunto train y 30% para el test.

El problema es que como vimos, tenemos un severo desbalance en los datos para la cantidad de las clases, recordemos cuantos datos teníamos:

- Número de imágenes en 'MildDemented': 717
- Número de imágenes en 'ModerateDemented': 52
- Número de imágenes en 'NonDemented': 2000
- Número de imágenes en 'VeryMildDemented': 1792

Es por esto que vamos a realizar una división de los datos de manera estratificada, lo que nos va a garantizar que la proporción en la división en cada clase se mantenga para ambos conjuntos de train y test de manera equilibrada, porque si no hacemos esto, al realizar la división tomando 70% de train, puede ser que todas nuestras imágenes de ModerateDemented sean seleccionadas para el Train, lo que

haría que no tengamos forma de probarlas, o puede pasar al revés, por esto lo haremos de manera estratificada.

Tomaremos 70% de los datos para el conjunto de train y 30% para el test, esto porque hay una característica donde sólo tenemos 52 datos, tomar esta división nos permite obtener una cantidad considerable de datos para cada uno de los conjuntos.

Así como hicimos en el proyecto 2, lo que haremos es tomar 3 semillas: 170119, 2024 y 123456789, para observar si la división de los datos afecta la división de los datos, calcularemos el "mejor" modelo en base a la métrica de accuracy, f1 score, precision y recall, usando un SVM con kernel lineal, kernel polinomial y Gaussiana.

La división estratificada brindó los siguientes resultados en la división de datos:

```
Conjunto de entrenamiento:
Cantidad de Datos en 'MildDemented': 502
Cantidad de Datos en 'ModerateDemented': 36
Cantidad de Datos en 'NonDemented': 1400
Cantidad de Datos en 'VeryMildDemented': 1254

Conjunto de prueba:
Cantidad de Datos en 'MildDemented': 215
Cantidad de Datos en 'ModerateDemented': 16
Cantidad de Datos en 'NonDemented': 600
Cantidad de Datos en 'VeryMildDemented': 538
```

En todos los casos de todas las semillas se tuvieron los mismos resultados, por lo que tomaremos únicamente la semilla 170119 de reproducibilidad para la división de los datos.

Ajustando el modelo SVM con Kernel Lineal sin PCA

De primera mano evaluamos un modelo svm donde sólo se especificó el tipo de kernel, en caso que este sea el modelo seleccionado haremos uso de revisión de hiperparámetros para observar cuáles serán los mejores cambios para obtener los mejores resultados.

El cálculo de las métricas de este modelo nos arrojó los siguientes resultados:

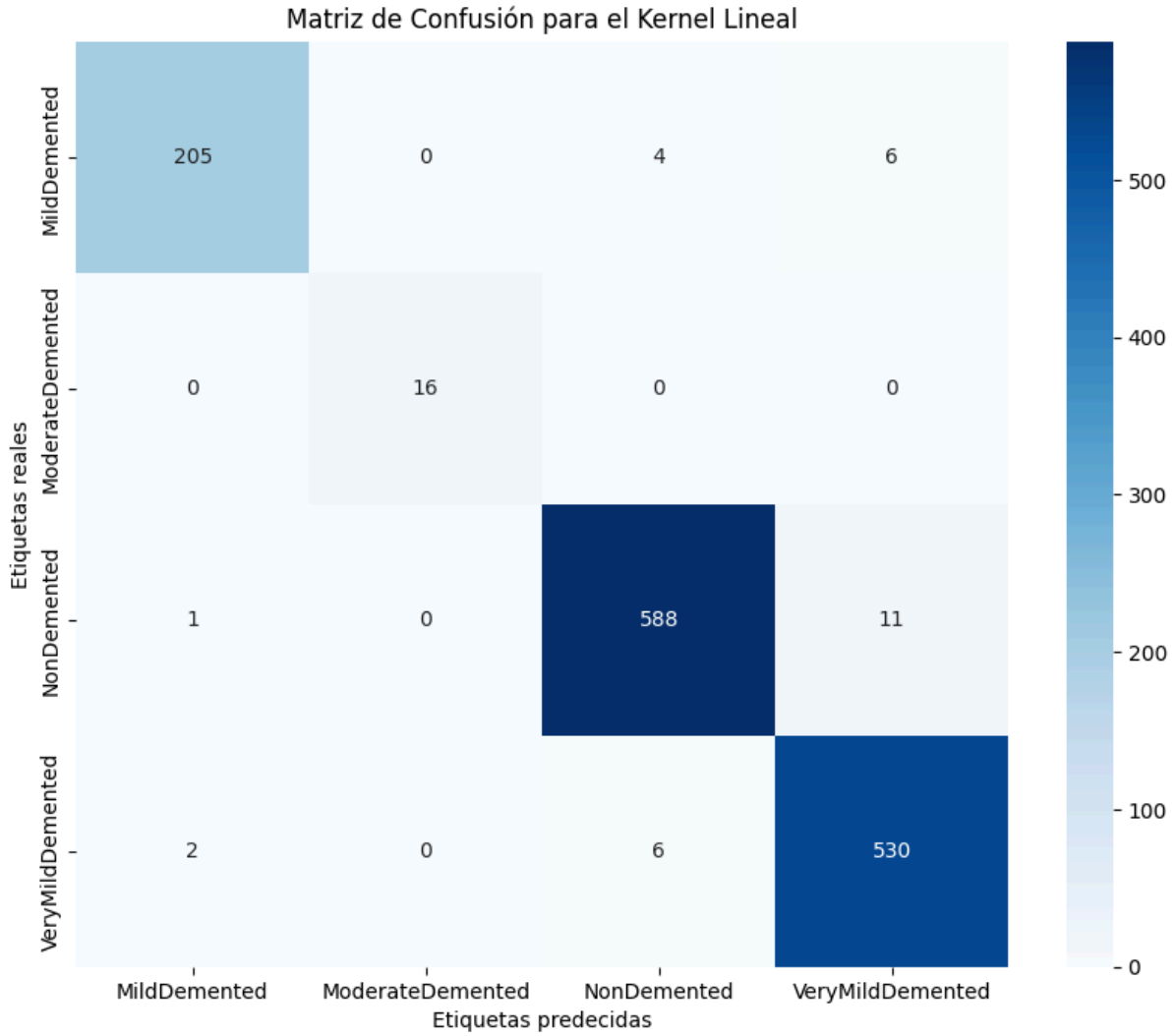
Accuracy del modelo SVM con kernel lineal, semilla 170119, división estratificada: 0.9780861943024105

Precisión del modelo: 0.9781923310013986

Recall del modelo: 0.9780861943024105

F1 score del modelo: 0.9780699471762048

Podemos ver que cada métrica tiene un valor de 0.978, es decir, se ajustan bien los datos al modelo, esto puede estar sobreajustado, deberemos verlo después en caso de continuar con el modelo, la matriz de confusión generada arroja los siguientes resultados:



Podemos ver que la categoría que más nos interesa, tiene una buena precisión, qué es la de VeryMild, puesto que casi todas se clasifican correctamente y son

pocas las que tienen una mala clasificación, aunque esto indica que de 538 personas, a 6 se les dirá que no tienen alzheimer lo que puede ser un problema, es mejor clasificar estas personas como si tuvieran un Alzheimer alto a decirles que no lo tengan, puesto que si les decimos que tienen Alzheimer las personas irán a empezar tratamientos y les harán mejores evaluaciones, pero decirles que no lo tienen, puede hacer que se confíen y que no comiencen los tratamientos a tiempo.

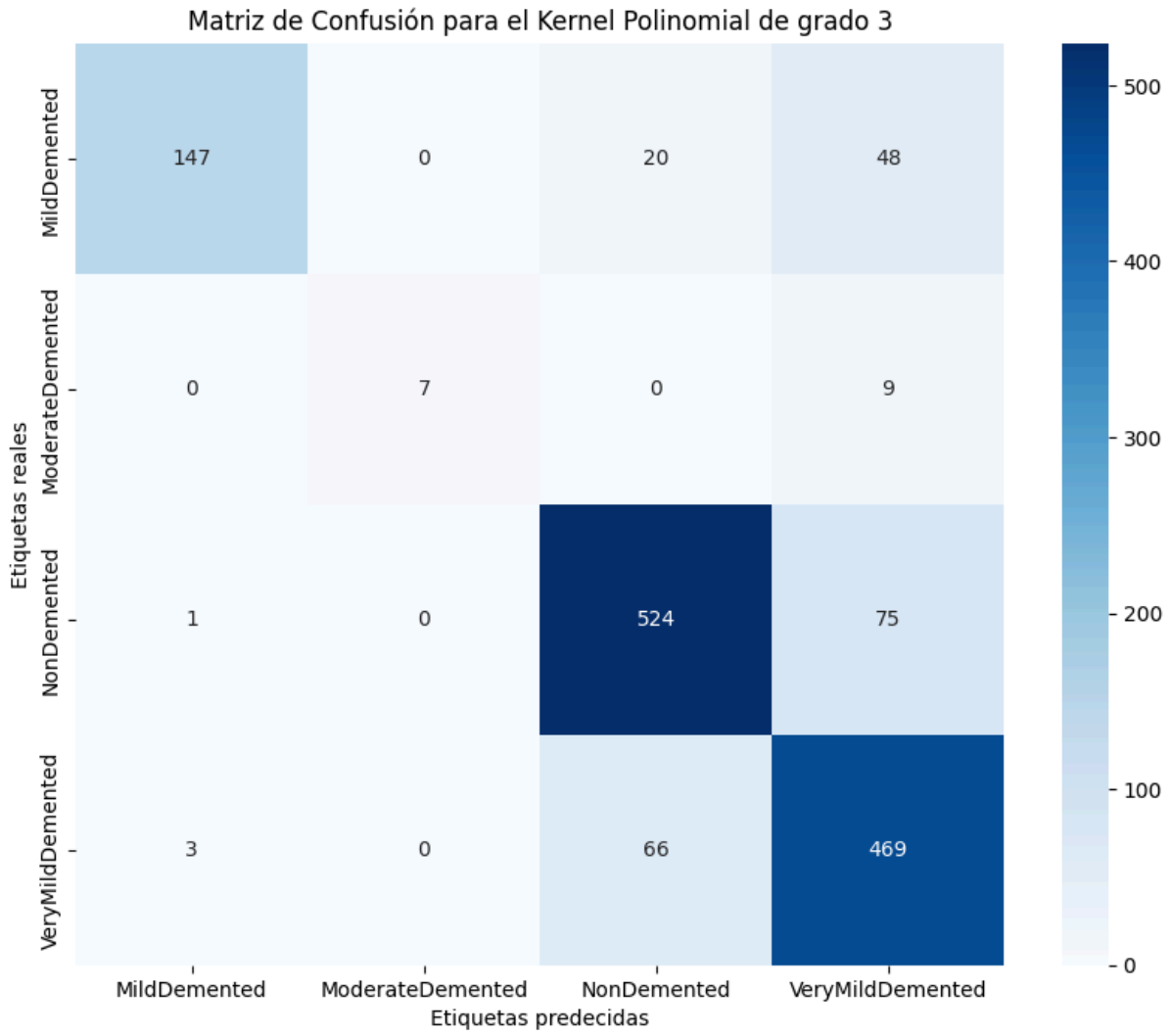
También podemos ver que la categoría de ModerateDemented tiene un 100% de clasificación, lo cual es extraño, seguramente nuestro modelo sí esté sobreajustado, lo vamos a ver si es que decidimos este modelo con kernel lineal.

Ajustando el modelo SVM con Kernel Polinomial sin PCA

Ahora veremos para el kernel polinomial, tomaremos el parámetro de regularización C = 1, en caso de decidir este modelo veremos si es necesario ajustarlo para tener una mayor o menor regularización, tomamos el valor de uno ya que se considera un valor "moderado", y tomamos el polinomio de grado 3.

Precisión del modelo: 0.8477366035523033
Recall del modelo: 0.8378378378378378
F1 score del modelo: 0.8365027709894699
Accuracy del modelo SVM con kernel polinomial de 3 grados y con C=1, semilla 170119, divisón estratificada: 0.8378378378378378

Este modelo tiene peor ajuste a los datos que el Lineal, aunque igual el resultado es bueno, como se mencionó, la categoría de mayor interés es VeryMild, a pesar de estos resultados, puede ser que en dicha categoría el ajuste sea mejor, lo que nos llevará a optar por este modelo, para ello veamos la matriz de confusión:



Podemos ver que no sólo el rendimiento según la medida Accuracy es menor al usar un kernel polinomial, si no que también la categoría que nos interesa clasificar de manera adecuada tiene más problemas, es por esto que no consideramos buena idea mantener este modelo, si bien podemos ajustar hiperparámetros para buscar mejores resultados, esta no será nuestra opción principal.

Ajustando el modelo SVM con Kernel Gaussiano sin PCA

Por último, el modelo SVM con kernel Gaussiano, tomando el valor de la gamma = scale, que se calcula como un inverso de la cantidad de características, y tomando de nuevo C = 1, nuevamente mencionamos que en caso de optar por este modelo, vamos a modificar los hiperparámetros más adelante, estos fueron tomados porque es algo así como el valor "común" que se suele usar.

Precisión del modelo: 0.788902015034208

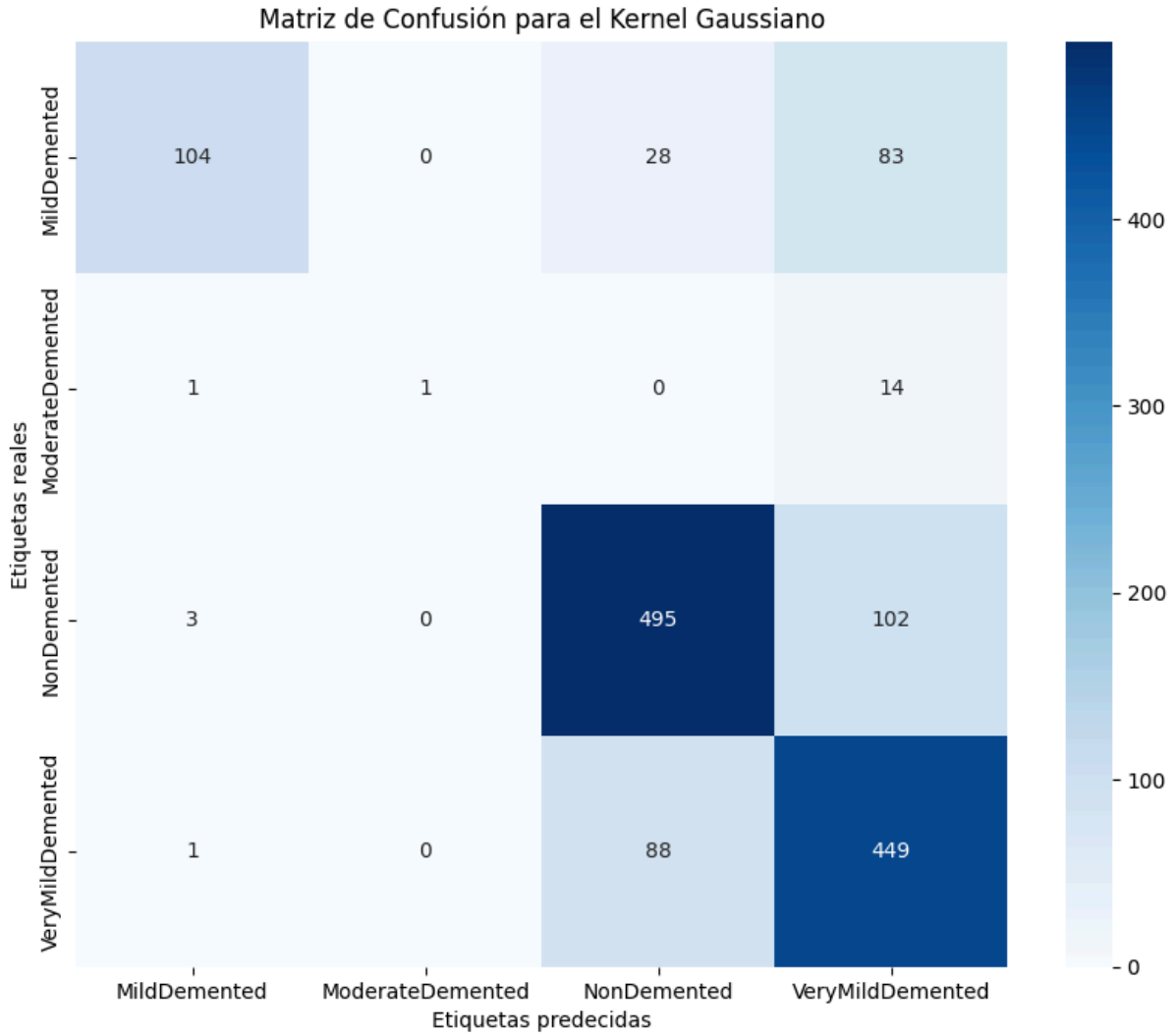
Recall del modelo: 0.7662527392257122

F1 score del modelo: 0.7580471717397034

Accuracy del modelo SVM con kernel Gaussiano, con gamma = scale y con C=1, semilla 170119, divisón estratificada: 0.7662527392257122

Este modelo sólo tuvo alrededor del 75% en cada métrica, siendo el peor de los tres, aún así no son tan malos resultados.

Ahora veamos la matriz de confusión de nuestra última prueba de kernel



Este modelo resultó peor para clasificar lo que queríamos, es por esto que después de ver todos los modelos, optamos por un kernel lineal y ajustar los hiperparámetros, ya que presentó mejores resultados y además, tardó menos tiempo en compilación, lo que es mejor para nosotros.

Ajustando hiperparámetros para SVM Kernel Lineal

En esta sección vamos a evaluar el modelo SVM con kernel lineal y ver cuales son los mejores hiperparámetros que podemos usar, vamos a hacer una búsqueda de malla, definida como:

'C': [0.1, 1, 10, 100],
'class_weight': [None, 'balanced']

Donde C es el parámetro de regularización, y class_weight son los pesos, vamos a probar el modelo cambiando cada valor de C y los tipos de pesos, para ver cual tiene aún mejores resultados, además de tomar 5 de CV, teniendo así un total de 40 fits a probar hasta encontrar el mejor modelo.

Nota: Para esta parte colab nos soltó un mensaje de que tardaría 83 horas en compilar, lo corrimos en un entorno local con 32GB de RAM y tardó alrededor de una hora y media, por lo que sugerimos no compilar esta sección. Tristemente por los problemas de compilación, únicamente calculamos aquí la accuracy, ya que esto fue antes de agregar las demás métricas, pero creemos que deben ser casi iguales así como se ha visto el comportamiento en las demás.

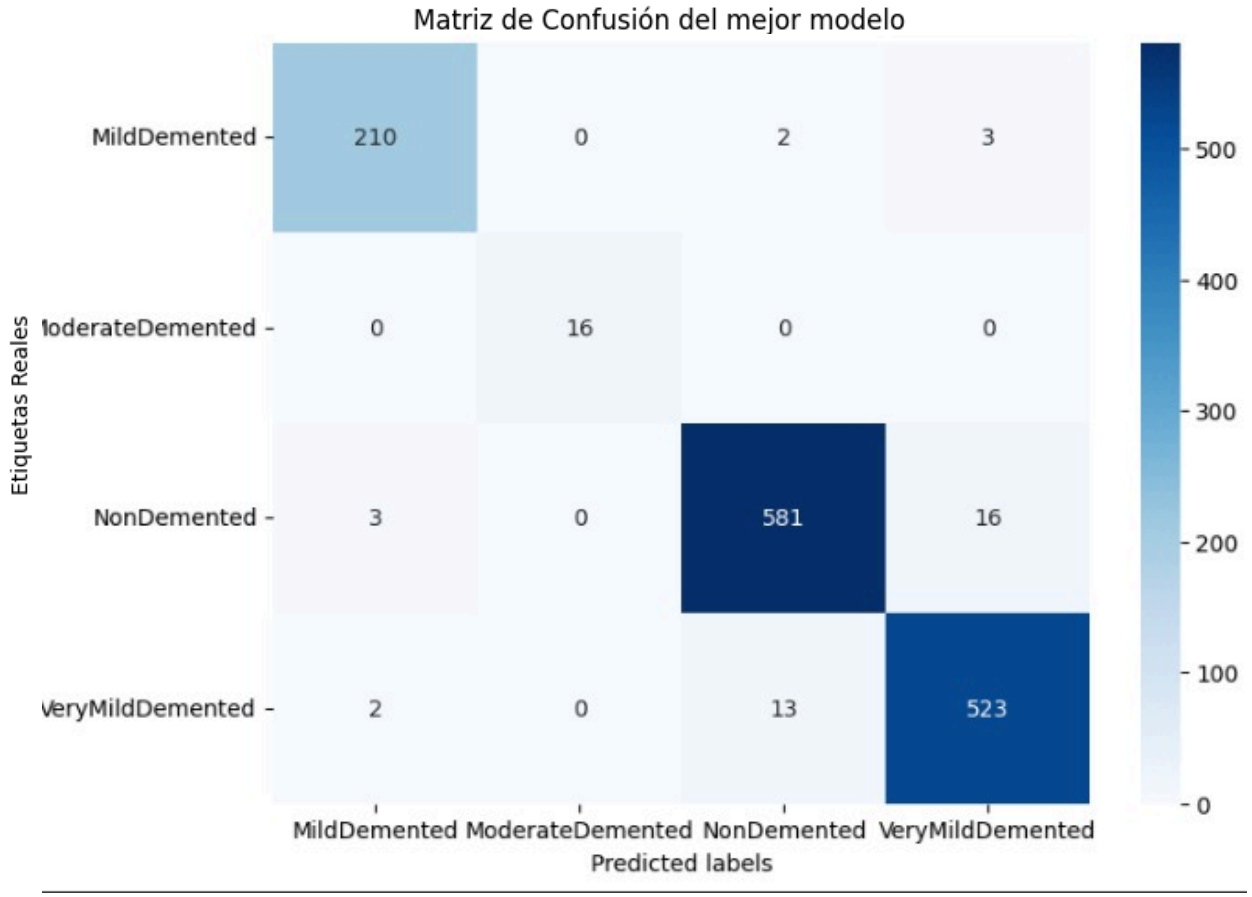
Los resultados son los siguientes:

Mejores hiperparámetros para el modelo SVM con kernel Lineal:

'C': 0.1, 'class_weight': None

Accuracy del modelo con los mejores hiperparámetros: 0.9817384952520087

La accuracy del modelo ahora con un valor de regularización igual a 0.1 tiene un valor del 98.17%, es decir, mejoró en alrededor de un 0.5%, lo cual es muy bueno puesto que queremos una exactitud casi del 100% para esta clasificación, aunque claro está, debemos ver que no haya un sobreajuste en nuestro modelo, pero veamos como se ve la matriz de confusión.



Podemos ver que si tiene mejores clasificaciones que el modelo antes de ajustar hiperparámetros, sobretodo en la parte de NonDemented, puede tener mejor clasificación para decirles a las personas que no tienen la enfermedad, y en general parece presentar mejores resultados, pero si notamos en la categoría de VeryMild, la que nos interesa, podemos ver que se presentan problemas, puesto que tiene una cantidad poco mayor de datos mal clasificados, donde categoriza 13 de los datos de VeryMild como NonDemented, mientras que en el modelo antes de ajustarlos sólo hubo 6 datos mal clasificados, es por este hecho que decidimos mejor conservar únicamente el modelo con kernel lineal, puesto que tiene mejor clasificación para resolver el problema al que nos enfrentamos.

Verificando SobreAjuste del modelo seleccionado

Podemos ver que nuestro modelo al que sólo le especificamos el tipo de kernel como lineal tiene un 97% - 98% en todas las métricas, y hay una de las características que tiene un 100% de efectividad, pues ahora necesitamos ver si el modelo no presenta problemas de sobreajuste, queríamos observar el sobreajuste a

partir de la curva de aprendizaje, pero después de una hora ejecutando, vimos que este proceso es demasiado tardado por la naturaleza de los datos, es por esto que optamos por otra manera de ver la posible existencia de sobreajuste calculando la Accuracy del modelo para cada conjunto, si la accuracy para el conjunto train es mayor que la de test, podemos inferir en que el modelo está sobreajustado.

Con los resultados obtenidos:

Accuracy del conjunto de Entrenamiento: 1.0
Accuracy del conjunto de Prueba: 0.9715120525931337

Indica que el modelo tuvo una accuracy perfecta en el conjunto de entrenamiento, esto a veces puede ser algo malo debido a que puede tomarse de inmediato como un sobreajuste, pero podemos ver que el Accuracy del conjunto de Prueba es de 0.97, lo que indica que el modelo también se está ajustando bien a los datos de prueba, por lo que se concluye que puede haber un pequeño sobreajuste, pero este no afecta demasiado a la calidad del modelo, por lo que este modelo resultó ser el más eficiente.

Conclusiones modelo SVM sin PCA

Al evaluar el modelo con diversos kernels vimos mucha variabilidad en lo que respecta a los puntajes de cada uno, el mejor fue el lineal con un 97% y además se desempeñó bien en el papel de la categoría que nos interesaba clasificar correctamente, después intentamos ver si no había alguna forma de mejorarlo haciendo uso de GridSearchCV para verificar los mejores hiperparámetros, esta modificación tuvo una mejoría hasta llegar al 98%, pero presentó más problemas en la categoría que nosotros tomamos como la más importante, teniendo más errores de clasificación tipo 2, es decir, falsos negativos, decirle a una persona que no está enferma cuando en realidad sí lo está, este tipo de errores pueden afectar la vida de las personas, por lo que tuvimos que optar por el modelo sin los hiperparámetros ya que presentaba menor cantidad de estos errores.

Al evaluar el sobreajuste del modelo, pudimos notar de manera numérica calculando la Accuracy para cada conjunto que no había tanto sobreajuste, por lo que el modelo terminó cumpliendo su propósito.

A pesar de que vimos que existe un mejor modelo que tiene una precisión mucho mejor, tuvimos que descartarlo, ya que no cumplió las expectativas pensadas para dar resolución al conflicto al que nos enfrentamos, pero aún con esto, nuestro modelo tiene un 97% de exactitud, precisión, recall y f1, es buen modelo y tiene una excelente evaluación, de igual manera hay que tener en cuenta que las imágenes colocadas son todas casi iguales, sería interesante ver que ocurre con una radiografía de un cerebro volteado, o que esté a color, pero igual podríamos afrontar estos problemas si modificamos el preprocesamiento de los datos.

Implementación del Modelo: SVM con PCA

Más Procesamiento parte 2

Primero vimos el desempeño del modelo svm usándolo sin PCA donde se obtuvieron muy buenos resultados, pero para poder hacer un mejor uso de este modelo usaremos el PCA para reducir la dimensión de los datos y así quizás ayudar a mejorar el desempeño, en este nuevo procesamiento vamos a realizar el aplanamiento de las imágenes ya preprocesadas (las que vimos que tienen el mismo color y tamaño), para tenerlas como vectores unidimensionales, donde cada imagen se convierte en un vector, y cada elemento del vector corresponde a un píxel de la imagen, para poder después realizar normalización que nos ayudará a que todos los píxeles tengan la misma escala y con esto, poder hacer PCA para reducir la dimensionalidad.

Ajustando el modelo SVM con Kernel Lineal con PCA

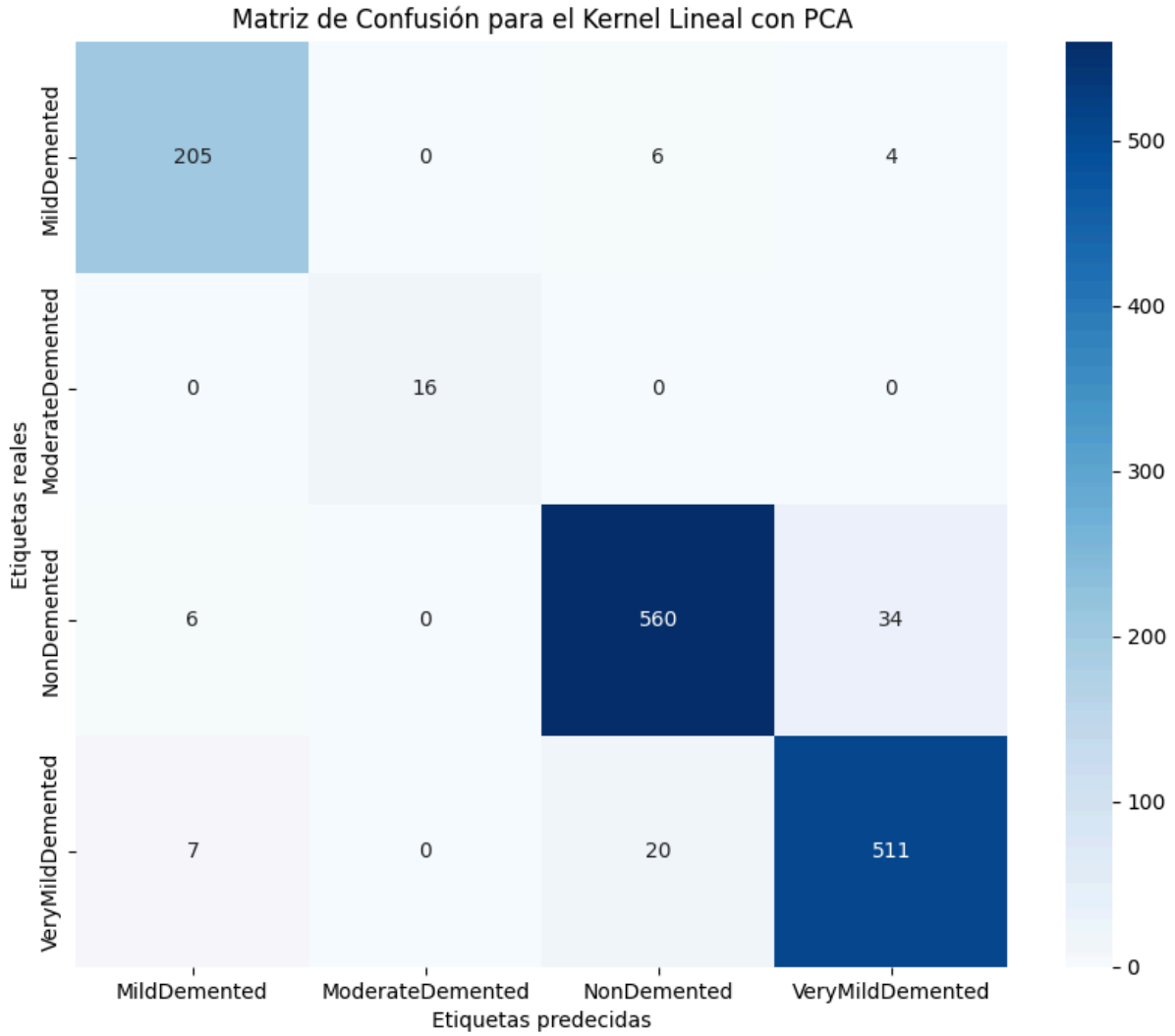
Vamos a usar la misma semilla de reproducibilidad para los datos, tomando una división estratificada, la semilla en cuestión es 170119, primero probaremos únicamente el kernel lineal. Donde resultó:

Precisión del modelo: 0.9439876720293405
Recall del modelo: 0.943754565376187

F1 score del modelo: 0.9437684578749922

Accuracy del modelo SVM con kernel lineal, semilla 170119, divisón estratificada y PCA: 0.943754565376187

Podemos ver que la accuracy del modelo con kernel lineal post pca tiene una accuracy peor a antes de usar el PCA, siendo de 94% que es un resultado muy eficiente aunque resulta menos eficiente comparado al 97% antes obtenido, pero la matriz de confusión y los resultados particulares de la categoría VeryMildDemented son los que realmente influyen en si escoger o no este modelo.



Este modelo tiene altas deficiencias en la categoría que nos resulta de interés, por lo que sería descartado de manera inmediata.

Ajustando el modelo SVM con kernel Polinomial con PCA

Ahora veremos para el kernel polinomial, tomaremos el parámetro de regularización C = 1, en caso de decidir este modelo veremos si es necesario ajustarlo para tener una mayor o menor regularización, tomamos el valor de uno ya que se considera un valor "moderado", y tomamos el polinomio de grado 3.

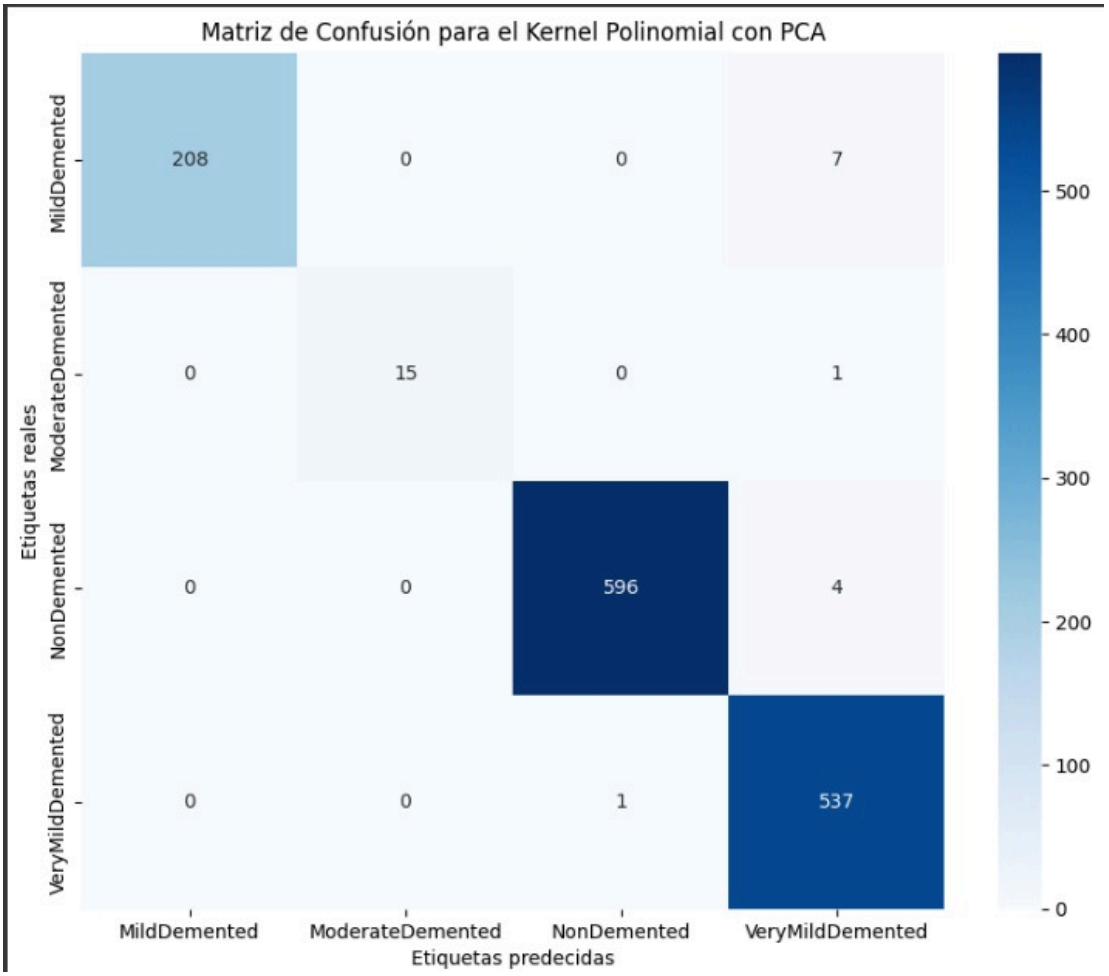
Precisión del modelo: 0.9827873551232355

Recall del modelo: 0.9824689554419284

F1 score del modelo: 0.9820946901103326

Accuracy del modelo SVM con kernel polinomial de 3 grados y con C=1, semilla 170119, divisón estratificada y PCA: 0.9905689554419284

Con un 99.05% de accuracy y un valor de aproximadamente 98.2% para todas las demás métricas podemos ver que este modelo ha sido el mejor clasificador hasta ahora, teniendo la siguiente matriz de confusión:



Teniendo únicamente un falso negativo en la categoría de mayor importancia para el contexto el proyecto, este modelo presenta resultados impresionantes, debemos ver si esto no tiene sobreajuste y en caso de ser el mejor, podemos probar con el conjunto test que tiene el repositorio de kaggle para probar la evaluación del modelo.

Ajustando el modelo SVM con kernel Gaussiano con PCA

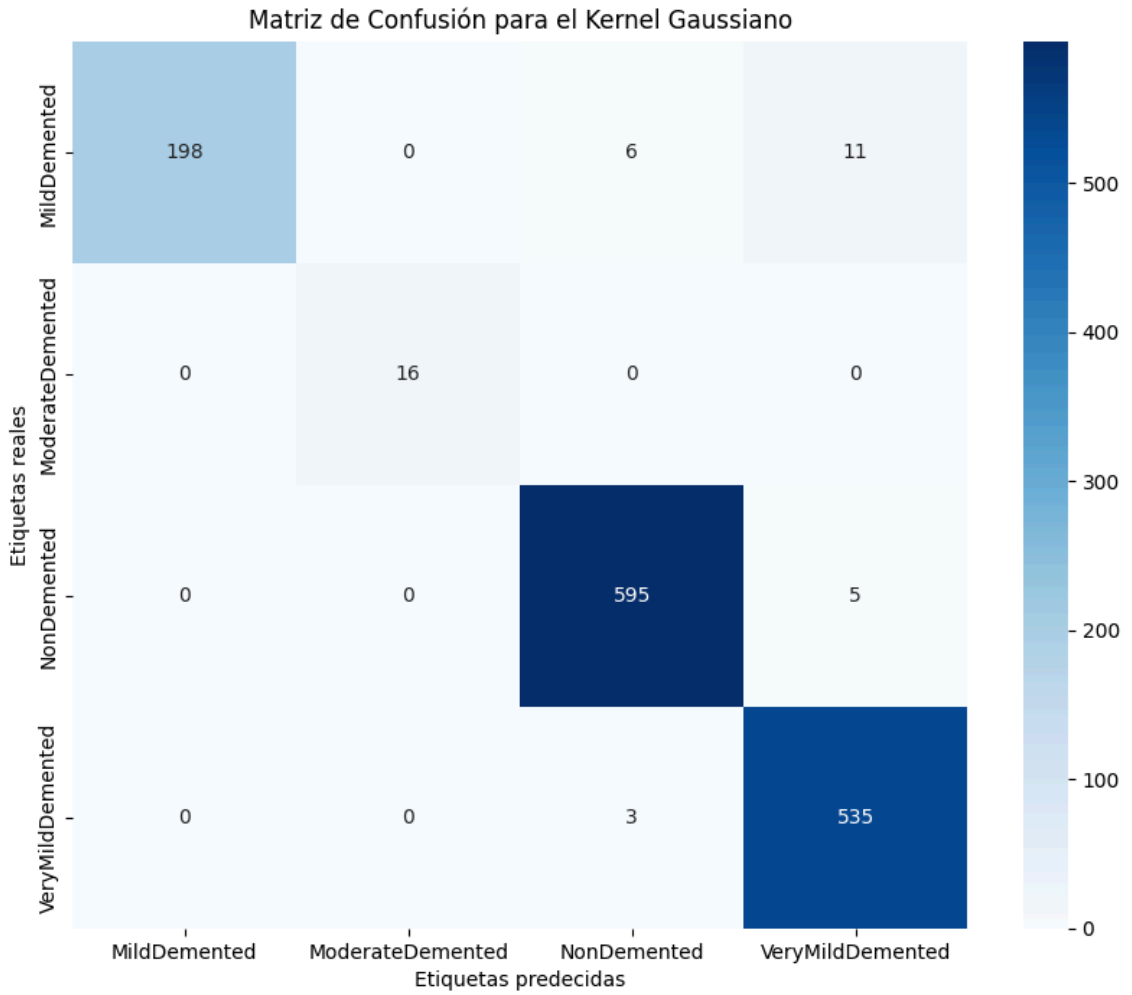
Por último, el modelo SVM con kernel Gaussiano, tomando el valor de la gamma como un inverso de la cantidad de características, y tomando de nuevo C = 1, nuevamente mencionamos que en caso de optar por este modelo, vamos a modificar los hiperparámetros más adelante, estos fueron tomados porque es algo así como el valor "común" que se suele usar.

Precisión del modelo: 0.9820577780217686
Recall del modelo: 0.9817384952520087
F1 score del modelo: 0.9815827492144292
Accuracy del modelo SVM con kernel Gaussiano, con gamma = scale y con C=1, semilla 170119, divisón estratificada y PCA: 0.9817384952520087

Con una accuracy del 98%, este modelo presenta mejores resultados generales al kernel lineal, pero peores al polinomial, también, podemos darnos cuenta que resultó con mejores evaluaciones al modelo gaussiano donde no habíamos procesado con PCA, por lo que el PCA sí dió mejores resultados en general.

En la matriz de confusión vemos que también se tienen muy buenos resultados, únicamente teniendo 3 falsos negativos en la categoría VeryMildDemented.

Aún con esto no fue capaz de destronar al modelo de kernel polinomial, por lo que vamos a proceder a ver los mejores hiperparámetros para el kernel polinomial.



Ajustando hiperparámetros para SVM Kernel Polinomial

Una vez visto que el mejor modelo resultó ser el modelo con kernel polinomial post-pca, es momento de ver cuáles serían los mejores hiperparámetros para nuestro modelo, tomaremos de nuevo el uso de gridsearchCV con 5 de CV, tomando una malla donde se irá cambiando lo siguiente:

'C': [0.1, 1, 10],
'degree': [2,3,4]

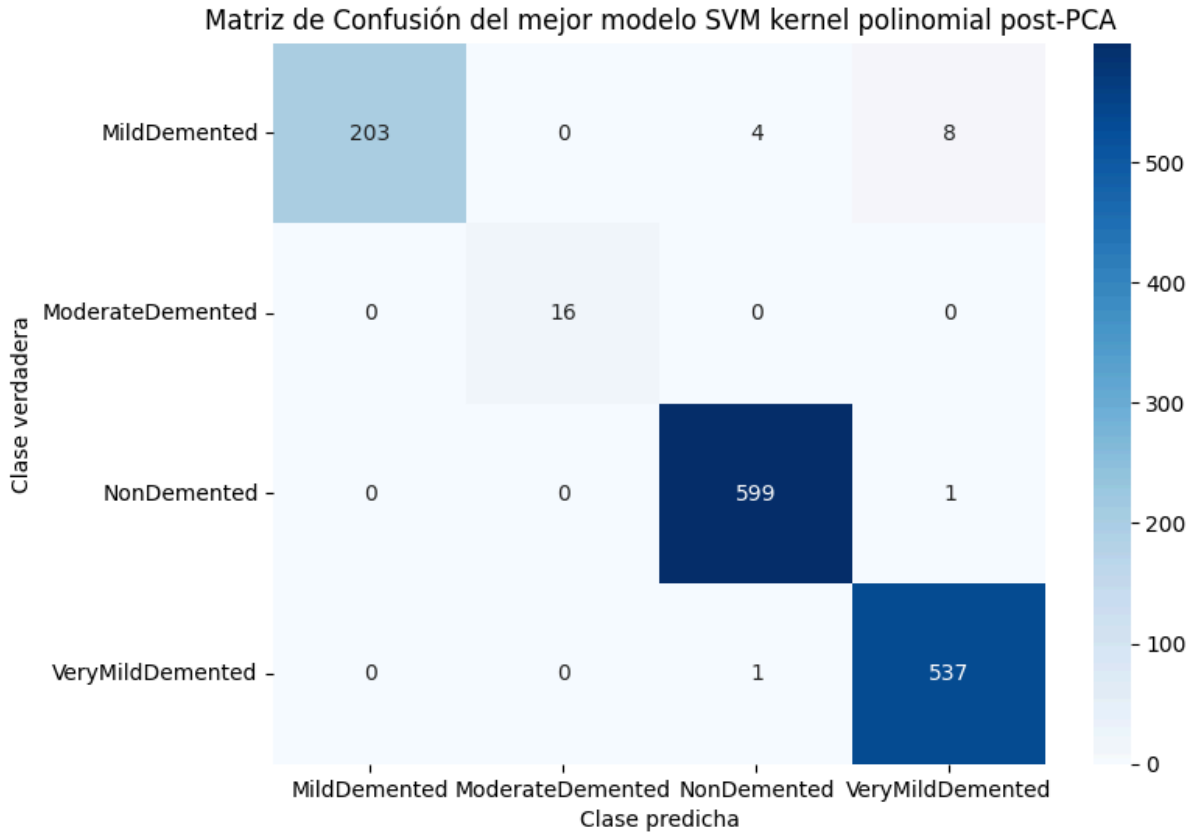
Así es como tendremos al final un total de 45 fits, donde se calcularán los mejores hiperparámetros para el modelo.

No usaremos ahora el hiperparámetro de los pesos, como la división está estratificada, esto no sentimos que sea tan necesario de verificar.

Mejores hiperparámetros: {'C': 10, 'degree': 2}

Precisión del modelo: 0.9898940691852809
Recall del modelo: 0.9897735573411249
F1 score del modelo: 0.9896819710318369
Accuracy del mejor modelo SVM polinomial con división estratificada y PCA: 0.9941735573411249

Este mejor modelo presenta una accuracy de 99.41%, que sí es mejor al modelo antes de ajustar los mejores hiperparámetros por un 0.36%, además también tiene una mejoría en cada una de las métricas, ahora veamos la matriz de confusión para ver que tan bien resulta este modelo:



Podemos ver que este modelo mejora un poco las demás categorías, mientras que la de VeryMildDemented se mantiene de igual manera, es por esto que decidiremos optar por este mejor modelo, presenta buenos resultados en todas las categorías y además, mantiene los excelentes resultados en la categoría VMD que es la de mayor importancia en este proyecto.

Verificando Sobreajuste del modelo seleccionado

Ahora veremos el sobreajuste que pueda tener el mejor modelo: Polinomial con grado 2, valor de regularización igual a 10 después de usar PCA, calculando las diferencias en la métrica accuracy para cada uno de los conjuntos train y test.

Accuracy del conjunto de Entrenamiento: 1.0
Accuracy del conjunto de Prueba: 0.9897735573411249

Con puntuaciones casi perfectas para ambos conjuntos, podemos inferir que el modelo clasifica bien ambas categorías y no hay suficiente evidencia de sobreajuste, lo que sugiere que el modelo es casi perfecto y soluciona bien nuestro problema.

Conclusiones modelo SVM con PCA

Aplicar PCA a los datos no solo mejoró considerablemente el modelo para el kernel polinomial, si no que también lo hizo para el gaussiano, en el caso del lineal empeoró un poco, pero aún así los resultados fueron muy buenos, por lo que el PCA ayudó mucho al ajuste de los datos, llegando a tener hasta un 99.41% de precisión de clasificación

Conclusiones finales

Llegamos a una precisión bastante buena superando estrepitosamente nuestra hipótesis inicial de un 90% de efectividad del modelo, teniendo una gran capacidad de clasificación en cada categoría, en nuestra categoría de interés tenemos sólo un falso negativo, para mejorarlo podríamos considerar más datos, considerar observar una red neuronal o ver más hiperparámetros, pero necesitamos un mayor poder computacional y conocimientos sobre cómo realizar los métodos descritos.

Aún con nuestro 99.41% de accuracy, debemos recordar que esto no podría sustituir el diagnóstico médico, puesto que este modelo y otros parecidos llegarían a ser sólo un apoyo a la medicina, más no una sustitución.

Referencias:

Alzheimer’s Association. (2024). Datos y cifras sobre la enfermedad de Alzheimer. Recuperado el 8 de mayo de 2024, de https://www.alz.org/alzheimer-demencia/datos-y-cifras

CDC. (2024). The Truth About Aging and Dementia. Recuperado el 8 de mayo de 2024, de https://www.cdc.gov/aging/publications/features/Alz-Greater-Risk.html

Alzheimer’s Association. (2024). ¿Qué es la enfermedad de Alzheimer? Recuperado el 8 de mayo de 2024, de https://www.alz.org/alzheimer-demencia/que-es-la-enfermedad-de-alzheimer

https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images