Tutorial Análisis de Patrones, GITA UdeA

Luis Fernando Torres T[1]

[1]Dpto. de Electrónica y Telecomunicaciones Universidad de Antioquia, Medellín, Colombia
**E-mail: luis.torres1@udea.edu.co*

Keywords: PCA, LDA, KNN, SVM, K-Fold, métricas de desempeño, malla de búsqueda,diagrama de distribuciones, curva ROC y AUC

RESUMEN

En el presente informe, se familiariza, se explica, y se muestra por medio de un ejercicio práctico como es la implementación de cada una de las etapas que componen el análisis de patrones. Se explica la primera etapa que consiste en la adquisición de los datos, luego se realiza el acondicionamiento del audio en la etapa de preprocesado para ser analizados adecuadamente, posteriormente se hace la extracción de características donde se obtienen representaciones que el computador sea capaz de procesar, y luego entonces se procede a realizar el análisis de estos datos utilizando clasificadores como Máquina de Soporte Vectorial (SVM), K Vecinos más Cercanos (KNN), y utilizando con cada uno el método de validación K-Fold y una malla de búsqueda, para encontrar sus parámetros y obtener el mejor desempeño posible.

1 Introducción

El análisis de patrones es una importante área que tiene como propósito extraer información que permita establecer propiedades para clasificar y/o predecir objetos en un número específico de categorías o clases.[1]

Los algoritmos de regresión y clasificación, son algoritmos utilizados dentro de la inteligencia artificial para realizar tareas que están relacionadas con analizar e identificar patrones en un suceso, evento o problema, en el que se pueda extraer información. Para esto se utiliza modelos como *K Nearest Neighbor(KNN)* o *Support Vector Machine (SVM)*, que son los modelos que se utilizan para este tutorial. Se utiliza la librería *skit-learn (sklearn)*, la cual tiene todas las funciones necesarias para realizar el tutorial.

2 Marco teórico

El análisis de patrones contiene una serie de etapas a desarrollar, las cuales permiten un análisis completo que conlleva a buenos resultados, tal como se muestra en la siguiente figura:

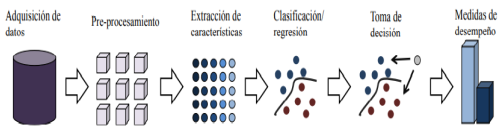

Fig. 1: Metodología para análisis de patrones [2]

2.1 Adquisición de datos

En primer lugar, se obtiene la adquisición de datos, el cual como su nombre lo indica, se encarga de recopilar y sensar toda la información y el conjunto de datos necesarios. Esta etapa depende de la aplicación, es decir, depende si se esta recopilando, imágenes, señales de voz, señales de audio, señales médicas, texto, vídeo, etc.

2.2 Preprocesamiento

Una vez se tenga los datos recopilados, se procede a realizar el preprocesamiento de los datos, es decir, adecuar la señal para obtener una información mejor representada. Para el caso de procesamiento de señales de voz, se normaliza el audio, en donde se debe eliminar el nivel DC que puede traer la señal desde el momento de adquisición de los datos, y esto se hace restando el promedio de la señal en cada punto o instante de tiempo. Por otra parte, para visualizar mejor los datos y estandarizar la señal, es importante normalizar las amplitudes, es decir, hacer que estos valores estén entre -1 y 1, para esto se divide cada valor de la señal sin nivel DC por el valor máximo de amplitud en valor absoluto. De esta manera se obtiene una señal centrada y con valores más sencillos de identificar y tratar.

2.3 Extracción de caracteristicas

Una vez se tiene el conjunto de datos bien organizado para realizar un análisis, la siguiente etapa es la extracción de características, donde se busca extraer e identificar características con las que puedan representar los datos obtenidos, basado en distintas medidas como la energía de la señal, los coeficientes espectrales, o razón de cruce por cero etc, y permita clasificar las distintas clases.

Existen varias técnicas para selección de características tales como análisis de componentes principales (PCA, del inglés *Principal Component Analysis*), análisis discriminante lineal (LDA, del inglés *Linear Discriminant Analysis*).

2.3.1 Análisis de componentes principales PCA: Es una técnica que se utiliza para reducir la dimensionalidad y analizar un conjunto de datos en el que se tienen las características organizadas en manera de tabla o matriz por lo general. Su objetivo es extraer la información importante para representarla como un conjunto de nuevas variables ortogonales llamadas componentes principales, donde se busca maximizar la varianza y reducir

la cantidad de características que aportan poca información de cada muestra.[3]

El método PCA, se basa en la búsqueda de nuevas variables (componentes principales), las cuales son una combinacion lineal de las características originales. La primer componente principal (PCA 1), es la dirección a lo largo de la cual las muestras muestran la mayor varianza de los datos, y el segundo componente principal (PCA 2) es la dirección no correlacionada con el primer componente a lo largo de la cual las muestras muestran la segunda mayor varianza, de esta manera cada componente tiene una varianza asociada que se va acumulando sucesivamente. Para aplicar PCA es necesario normalizar las características, donde los componentes principales son vectores propios normalizados de la matriz de covarianza de los datos y se ordenan según la cantidad de varianza presente en los datos que contienen, los cuales se usan para realizar la transformación de las muestras a un nuevo sub-espacio.[4][5][6]

Al utilizar unos pocos componentes (idealmente 3 componentes), cada muestra puede representarse con relativamente pocos números en lugar de con valores para miles de variables, lo cual permite obtener una representación gráfica y así evaluar visualmente las similitudes y diferencias entre las muestras y determinar si dichas muestras se pueden agrupar por clases. Cabe resaltar que el número de componentes se puede modificar según la cantidad de componentes que se requiera, o según el porcentaje de varianza de los datos que se desea conservar, teniendo en cuenta que el 100 % de la varianza es la matriz de datos antes de aplicarle PCA.

2.3.2 Análisis discriminante lineal LDA: Linear Discriminat Analysis o análisis discriminante lineal es una técnica de reducción de dimensionalidad que se usa comúnmente para problemas de clasificación supervisada, donde el objetivo es proyectar la matriz de datos original, en un espacio de menor dimensión. Para lograr este objetivo, es necesario realizar tres pasos. El primer paso consiste en calcular la separabilidad entre las distintas clases (es decir, la distancia entre las medias de las distintas clases), que se denomina varianza entre clases o matriz entre clases. El segundo paso consiste en calcular la distancia entre la media y las muestras de cada clase, lo que se denomina varianza dentro de la clase o matriz dentro de la clase. El tercer paso es construir el espacio de menor dimensión que maximiza la varianza entre clases y minimiza la varianza dentro de la clase. Se debe tener en cuenta que para realizar LDA el número de componentes viene dado por el numero de clases que contiene el dataset, donde se tiene n clases para $n-1$ componentes, es decir, que si se tienen 2 clases, se tendrá solo una componente.[7]

En otras palabras, LDA intenta encontrar un límite de decisión alrededor de cada grupo de una clase. Luego proyecta los puntos de datos a nuevas dimensiones de manera que los grupos estén tan separados entre sí como sea posible y los elementos individuales dentro de un grupo estén lo más cerca posible del centroide del grupo. Las nuevas dimensiones se clasifican en función de su capacidad para maximizar la distancia entre los grupos y minimizar la distancia entre los puntos de datos dentro de un grupo y sus centroides. Estas nuevas dimensiones forman los discriminantes lineales del conjunto de características.[8][9]

2.3.3 Diferencias entre LDA y PCA: Tanto el análisis discriminante lineal (LDA), como el análisis de componentes principales (PCA), son técnicas de transformación lineal que se utilizan comúnmente para la reducción de la dimensionalidad y contienen un cálculo matemático que se realiza de manera similar. PCA se puede describir como un algoritmo "no supervisado", ya que ïgnora"las etiquetas de clase (solo se utilizan para graficar e identificar visualmente las clases) y su objetivo es encontrar las direcciones (los llamados componentes principales) que maximizan la varianza en un conjunto de datos. A diferencia de PCA, LDA es "supervisado, dado que su objetivo es separar las clases, se debe tener en cuenta las etiquetas. LDA Calcula las direcciones ("discriminantes lineales") que representarán los ejes que maximizarán la separación entre múltiples clases.

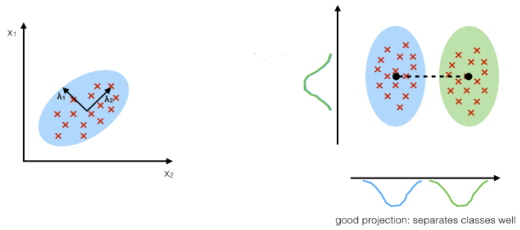

Fig. 2: Gráfico de comparación PCA vs LDA [8]

2.4 Clasificación de datos y toma de decisiones

Para realizar la clasificación de datos, es necesario realizar una partición o división del conjunto de datos la cual se le conoce como un *split_data*, donde se tiene un conjunto de datos entrenamiento los cuales contienen correctamente las etiquetas originales (conjunto de datos base para guiarse) y se le denomina *train_data*, por otra parte se tiene un conjunto datos de prueba denominados *test_data*, los cuales son evaluados por un modelo para predecir la posible etiqueta o clase, y se comparan con las etiquetas originales y así medir el desempeño del modelo.

2.4.1 Clasificación por K-vecinos mas cercanos KNN: Los K-vecinos más cercanos o K-Nearest neighbors (KNN), es un método de clasificación y agrupamiento que se puede aplicar para clasificar elementos de un conjunto en grupos o clases. Recibe este nombre debido a que el algoritmo asocia un nuevo elemento en un grupo según las similitudes o distancias entre los elementos del conjunto.[10], es decir, este algoritmo consiste en seleccionar un valor de K, donde representa los k datos más cercanos al valor que se desea predecir o clasificar (ver figura 3)(Generalmente, se recomienda elegir un número impar si el número de clases es par), cabe resaltar que lo más importante es seleccionar un valor de K acorde a los datos para tener una mayor precisión,

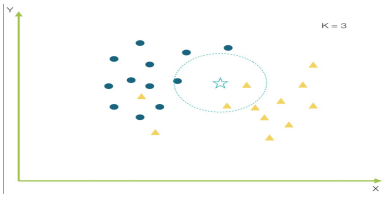

Fig. 3: Gráfico de ilustración para KNN[11]

En pocas palabras, para realizar el método de clasificación KNN se debe seguir los siguientes pasos:

- Calcular la distancia
- Encontrar sus vecinos más cercanos
- Votar a partir de las etiquetas

2.4.2 Máquinas de soporte vectorial SVM: Support vector machine (SVM) o Máquinas de soporte vectorial, es un algoritmo de aprendizaje supervisado que se utiliza en muchos problemas de clasificación y regresión, incluidas aplicaciones médicas de procesamiento de señales, procesamiento del lenguaje natural (NLP) y reconocimiento de imágenes y voz.

El objetivo de SVM es encontrar un hiperplano en un espacio de N dimensiones que clasifique claramente los datos para separar las clases (N el número de características, si el número de características de entrada es 2, el hiperplano es sólo una línea. Si el número de características de entrada es 3, el hiperplano se convierte en un plano bidimensional), en otras palabras, dados los datos de entrenamiento etiquetados el algoritmo genera un hiperplano óptimo que clasifica los nuevos ejemplos en dos espacios dimensionales.[12][13]

El margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores, el objetivo es encontrar un plano que tenga el máximo margen (ver figura 4), es decir, la máxima distancia entre los puntos de datos de ambas clases.

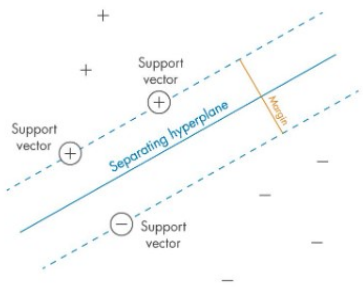

Fig. 4: Hiperplano y margen de SVM

SVM es un algoritmo de Machine Learning que pertenece a la clase de métodos kernel, es decir, donde se utiliza una función de kernel para transformar y separar las características. Las funciones de kernel asignan los datos a un espacio dimensional diferente, que suele ser superior, es decir toman un espacio de entrada de baja dimensión y lo transforman en un espacio dimensional más alto con la expectativa de que resulte más fácil separar las clases después de esta transformación, estas funciones se les conoce como núcleos o kernel. [13], dentro de los posibles Kernel se encuentra el núcleo lineal, polinómico, sigmoide y el más usado Función de base radial (RBF) o kernel gaussiano.

Aunque los algoritmos SVM están formulados para la clasificación binaria, los algoritmos SVM multiclase se construyen combinando varios clasificadores binarios. Además, los kernels hacen que los SVM sean más flexibles y capaces de gestionar problemas no lineales.

Este algoritmo tiene algunos parámetros que pueden ajustarse y que se denominan hiperparámetros, los cuales es importante ajustar antes de entrenar los modelos. Los hiperparámetros son muy importantes para construir modelos robustos y precisos, estos ayudan a encontrar el equilibrio entre el sesgo y la varianza lo que conlleva a evitar que el modelo se ajuste en exceso(overfitting) o en defecto(underfitting).

- **Hiperparámetro *C***: Es un hiperparámetro en SVM para controlar el error, con el cual se añade una especie de penalización por cada punto de datos mal clasificado. Si ***C*** es pequeño, la penalización por puntos mal clasificados es baja, por lo que se elige un límite de decisión con un gran margen a costa de un mayor número de clasificaciones erróneas. Si ***C*** es grande, la SVM intenta minimizar el número de ejemplos mal clasificados debido a la elevada penalización, lo que da lugar a un límite de decisión con un margen menor. La penalización no es la misma para todos los ejemplos mal clasificados. Es directamente proporcional a la distancia al límite de decisión. ***C*** tiene un intervalo normalmente entre 0.1 y 100 $(0{,}1 < C < 100)$
- **Hiperparámetro *Gamma* (Γ)**: Se usa cuando se usa únicamente con el kernel RBF Gaussiano y controla la distancia de influencia de un único punto de entrenamiento.Los valores bajos de ***Gamma*** indican un radio de similitud grande que hace que se agrupen más puntos. Para valores altos de ***Gamma***, los puntos deben estar muy cerca unos de otros para ser considerados en el mismo grupo (o clase). Por lo tanto, los modelos con valores de ***Gamma*** muy grandes tienden a sobre ajustarse.***Gamma*** tiene un intervalo típicamente entre 0.0001 y 10 $(0{,}0001 < \Gamma < 10)$[14]

2.5 Optimización de Modelos

Este apartado hace referencia a que método se utiliza para encontrar el mejor hiperparámetro según la organización de los datos, es decir, a que hiperparámetro escoger en cada uno de los modelos y que técnicas se pueden usar para obtener el mejor rendimiento en el modelo.

2.5.1 Validación cruzada K-fold: La validación cruzada es un método de evaluación de modelos que tiene como objetivo hacer un uso efectivo de todas las características tanto para entrenamiento como para pruebas,permitiendo así obtener modelos más estables, reduciendo el sobre ajuste, ya que permite utilizar todos los datos del dataset para entrenar el modelo así entonces identificar la existencia de diferentes problemas durante el proceso de entrenamiento.[15]

La validación cruzada de k iteraciones o k-fold cross validation, consiste en dividir los datos originales en k subconjuntos, donde al realizar el entrenamiento de un modelo, se toma cada k subconjunto como conjunto de prueba del modelo, mientras que el resto de subconjuntos (k-1) se toman como conjunto de entrenamiento. El proceso de validación cruzada se repite k veces, y en cada iteración se toma un conjunto de prueba diferente, Al finalizar las k iteraciones, se calcula el promedio de los resultados de precisión y error obtenidos para cada subconjunto de prueba.[16]

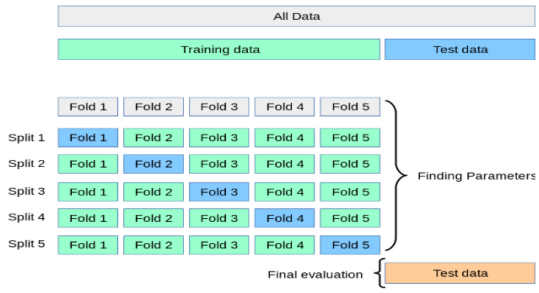

Fig. 5: K-Fold cross validation

2.5.2 Optimización de hiperpárametros por malla de búsqueda: Dado que cada modelo puede obtener distintos resultados acorde a los ajustes que se le haga, es importante encontrar los hiperparámetros adecuados para lograr un modelo con el mayor éxito posible.

Para ello se utiliza una malla de búsqueda (*Gridsearch*) que lo que hace es variar los hiperparámetros hasta encontrar los mejores parámetros, esto suele hacerse en combinación con K-Fold para encontrar el mejor algoritmo posible que se acomode a todo el dataset completo y que tenga el mejor rendimiento. (ver figura 6).

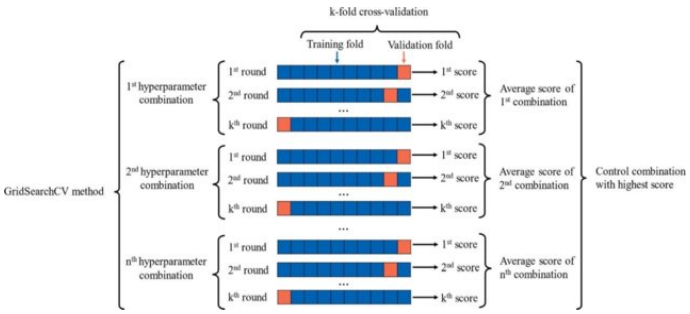

Fig. 6: Malla de búsqueda de hiperparámetros en cada Fold[17]

Esta malla de búsqueda se realiza con los hiperparámetros según el modelo, es decir para un modelo de SVM se realiza la búsqueda del mejor hiperparámetro C y Γ, para un KNN el mejor valor para K, y así para cada uno.

2.6 Métricas

Es importante evaluar que tan exitoso es el modelo construido, y esto se hace a través de algunas métricas predeterminadas. Cuando se trata de construir modelos de clasificación, lo más probable es que se use una métrica denominada matriz de confusión y métricas relacionadas con esta matriz para evaluar, supervisar y gestionar el desempeño del modelo.

2.6.1 Matriz de confusión: Una matriz de confusión o también conocida como matriz de error, es una matriz que se presenta en forma de tabla resumida que permite evaluar y analizar el rendimiento de un modelo[18]. Esta matriz pone en relación las predicciones realizadas por el modelo con los resultados correctos que debería haber mostrado, de esta manera cada índice o posición de la matriz brinda una medida de desempeño o métrica en relación a qué tipo de errores y de aciertos tiene el modelo sobre los datos propuesto, lo cual determinada que tan preciso es el modelo(ver figura 13).

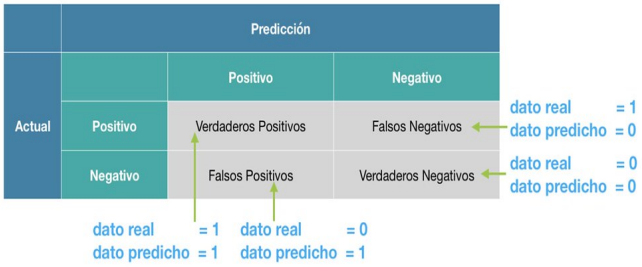

Fig. 7: Matriz de confusión

La palabra Verdadero hace referencia a que efectivamente ha predicho bien una clase, por el contrario la palabra Falso hace referencia a que se ha predicho erróneamente la clase. Por otra parte la palabra positivo hace referencia a la clase 1 y negativo a la clase 0.

- **Verdaderos Positivos o True Positive**: Los TP o VP es el resultado en el que el modelo predice correctamente la clase positiva, es decir son los casos en los que los datos reales son los correctos (Verdadero) y la predicción también es correcta (Verdadero).Está en la posición [0,0] de la matriz.
- **Falsos Negativos**: Los FN es un resultado en el que el modelo predice incorrectamente la clase negativa cuando en realidad es positiva, es decir son los casos en que los datos reales indica que es verdadero y el pronóstico es falso, ocasionando que la predicción ha sido incorrecta. Está en la posición [0,1] de la matriz.
- **Falsos Positivos**: Los FP es el resultado donde el modelo predice incorrectamente la clase positiva cuando en realidad es negativa, es decir son los casos en que los datos reales indica que es falso y la predicción indica que es verdadero, es decir la predicción ha sido errónea. Está en la posición [1,0] de la matriz.
- **Verdaderos negativos**: Los TN o VN es el resultado donde el modelo predice correctamente la clase negativa, es decir

son los casos en los que los datos reales son negativos y el pronóstico también es negativos. Está en la posición [1,1] de la matriz.

2.6.2 Métricas de la matriz de confusión: La información suministrada por la matriz de confusión permite generar las principales métricas que posibilita un análisis de las probabilidades de éxito del modelo al momento de calificarlo.

- **Accuracy**: Es una métrica donde se mide la proporción de predicciones que el modelo clasificó correctamente.[19]

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

Se obtiene un número que se encuentra en el intervalo entre 0 y 1, donde si el resultado es cercano a 1, da a entender que el modelo tuvo muy pocos errores al momento de realizar la predicción.

- **Sensibilidad o Recall**: También conocido como la tasa verdadera positiva (TPR), es una métrica que representa la capacidad del modelo de detectar los casos positivos.[20], es decir, es la relación entre las predicciones positivas correctas y el número total de predicciones positivas, en otras palabras, cuán sensible es el clasificador para detectar instancias positivas.

$$Sensibilidad = \frac{TP}{TP + FN} \quad (2)$$

- **Especificidad**: Tambien conocida como tasa negativa real (TNR), es una métrica que mide la proporción de negativos reales que se identifican correctamente. es decir, representa la capacidad del modelo para identificar una muestra de la clase negativa.Es lo opuesto a la sensibilidad.

$$Especificidad = \frac{TN}{FP + TN} \quad (3)$$

- **Precisión**: Se conoce como valor predictivo positivo y es la proporción de instancias relevantes entre las instancias recuperadas[19].Es una métrica que relaciona la cantidad de muestras de la clase positiva sobre la cantidad total de las predicciones que se hicieron para la clase positiva.

$$Precisin = \frac{TP}{TP + FP} \quad (4)$$

- **F1 Score**: El puntaje F1 es una métrica de la precisión de una prueba, es decir, es la media armónica de precisión y la sensibilidad. En general, es una medida de la precisión y robustez de su modelo[19].

$$F1Score = \frac{2TP}{2TP + FP + FN} \quad (5)$$

Puede tener una puntuación máxima de 1 (precisión y recuerdo perfectos) y una mínima de 0.

2.6.3 Diagrama de distribución, curva ROC y AUC: El diagrama de distribución es una forma de ver los resultados obtenidos en la matriz de confusión de manera gráfica por medio de un histograma, el cual representa la distribución de los datos encontrados. Para ello es necesario usar los scores obtenidos (TN.TP,FN.FP), ya que dado esto, los datos de las clases se separan a través de un umbral, el cual dependiendo del clasificador está ubicado normalmente en 0 o 0.5.

Todos los valores positivos por encima del umbral serán "verdaderos positivos" y los valores negativos por encima del umbral será "falsos positivos", ya que se predicen incorrectamente como positivos.Todos los valores negativos por debajo del umbral serán "verdaderos negativos" y los valores positivos por debajo del umbral serán "falsos negativos", ya que se pronostican incorrectamente como negativos.

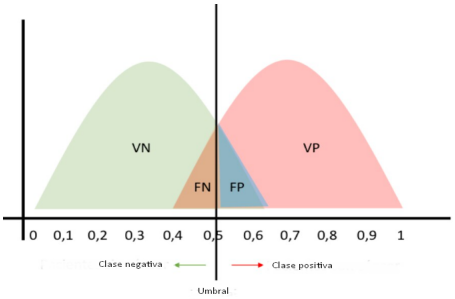

Fig. 8: Diagrama de distribución

A partir del diagrama de distribución, se puede obtener la curva ROC(Receiver operating characteristic), lo cual es una representación gráfica de la sensibilidad frente a la 1-especificidad. Esta gráfica siempre debe ser cuadrada, es decir debe tener la misma dimensión espacial a la hora de ser graficada.

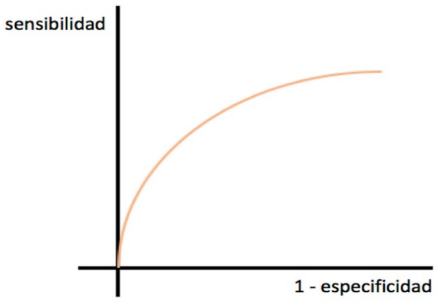

Fig. 9: Curva ROC

Para realizar la curva ROC se recorre todo el diagrama de distribución desde el inicio hasta al final moviendo el umbral a puntos específicos (thresholds), en donde se encuentra la relación entre la sensibilidad y especificidad, y se dibuja en la curva. Si no se tienen errores de clasificación la curva ROC tiende a ser totalmente recta, mientras que si se obtiene algún fallo (FN o FP), la recta se curva hasta encontrar el siguiente punto del umbral y así se obtiene una curva ROC completa.

Una vez obtenida la curva ROC, se procede a encontrar el AUC de la curva ROC (Area Under Curve o área bajo la curva), la cual indica que tan bueno está funcionando el modelo al momento de clasificar.

El área bajo la curva depende del diagrama de distribución ya que si se presenta una situación ideal, en la que las dos curvas no se superponen en absoluto, el modelo es perfectamente capaz de distinguir entre clase positiva y clase negativa dando como resultado un $AUC = 1$,

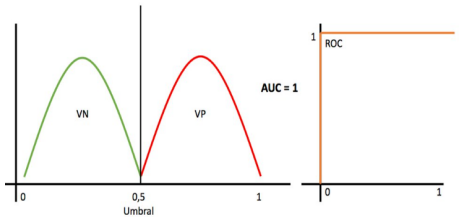

Fig. 10: Situación ideal con $AUC = 1$

Por otra parte si en el diagrama de distribución se superponen las curvas, es decir se tienen errores (FN y FP). El AUC dará un número que está entre 1 y 0.5,lo cual significa que hay un cierto % de probabilidad de que el modelo pueda distinguir entre clase positiva y clase negativa.

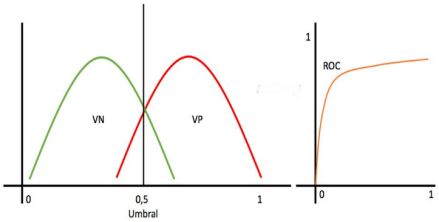

Fig. 11: Situación con FN y FP

Si se obtiene una AUC muy cercano a $0{,}5$, es el peor de los casos, pues el modelo no tiene capacidad de discriminación para distinguir entre clase positiva y clase negativa.

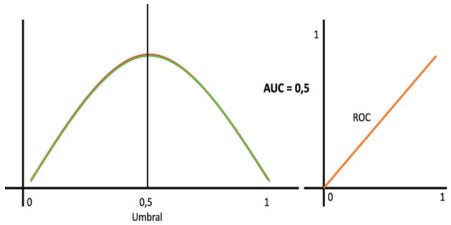

Fig. 12: Peor de los casos para un modelo según AUC

Si se obtiene que $AUC = 0$, da a entender que el modelo está clasificando todo las clases al contrario, esto significa que el modelo predice la clase negativa como una clase positiva y viceversa.

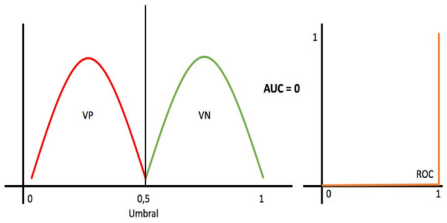

Fig. 13: Clasificación contraria de clases según la AUC

3 Análisis de resultados

3.1 Preprocesamiento

En primer lugar se obtiene una señal de un audio donde después de obtener sus datos, se gráfica tal cual como se tiene los datos

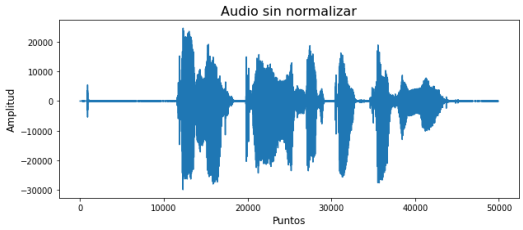

Fig. 14: Gráfica de audio sin normalizar

Se procede a normalizar el audio para poder obtener, valores sin nivel DC y que se encuentren entre -1 y 1, además, se realiza un ajuste al eje x, para que se tenga el tiempo de duración del audio y no puntos o muestras que es lo que se tiene normalmente.

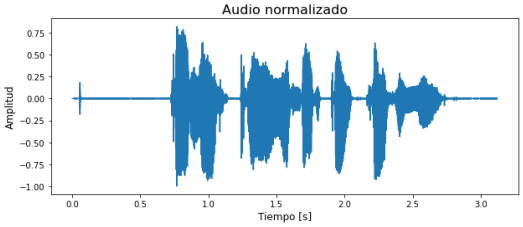

Fig. 15: Gráfica de audio normalizado

Cabe resaltar que normalizar un audio, sirve para estandarizar la escala y así visualizar y analizar de manera más organizada la información que contiene de manera gráfica y numérica, ya que el contenido no cambia en nada.

Luego entonces se procede a realizar el proceso de enventanado, es decir, tomar pequeñas porciones del audio, para obtener información en pequeños segmentos que sean más sencillos de identificar y analizar.

3.2 *Extracción de caracteristicas*

Para este tutorial se tienen un dataset que contiene un conjunto de muestras con una cantidad establecida de características para determinar si una persona tiene altas emociones o bajas emociones. Dentro de los datos se tiene que cada emoción tiene una dimensión asociada, para las emociones altas se tiene una dimensión de 313 muestras para el cual se obtiene 6373 características, y para las emociones bajas una dimensión de 222 muestras con igual cantidad de características, lo cual al juntar todos los datos se obtiene una matriz resultante de dimensión $535 \cdot 6373$.

3.2.1 Análisis de componentes principales PCA: Dado que se tiene una matriz de dimensión $535 \cdot 6373$, es imposible de visualizar, razón por la cual se puede utilizar PCA para poder evaluar el comportamiento de los datos de manera gráfica, y eliminar datos que aporten poca información. Para ello, se utiliza la librería *sk-learn de python*[20], y así obtener la siguiente gráfica:

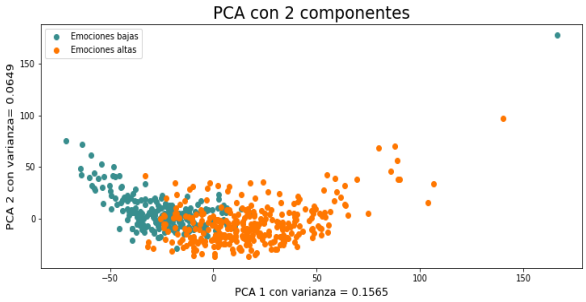

Fig. 16: Visualización de los datos utilizando PCA

Se puede observar que al realizar PCA con 2 componentes, se puede visualizar los datos que se dividen en 2 clases, altas emociones (color naranja) y bajas emociones (color azul), donde se puede observar que se tiende a tener un agrupamiento o patrón según la clase, aunque en algunos casos se solapen algunas muestras.

Para este análisis se tomó 2 componentes, para las cuales cada una tiene una varianza asociada, para el caso de la primera componente $PCA1 = 15,65\,\%$ de la varianza y para la segunda componente $PCA2 = 6,49\,\%$de la varianza total, esto quiere decir que se tiene una varianza del $22,13\,\%$. Dado que inicialmente se tienen 6373 características y se están agrupando en 2 componentes, por lo tanto, la perdida de la información es notable, pero aún así se tiene toda la información de manera visual.

3.2.2 Análisis discriminante lineal LDA: Al igual con el PCA se utiliza LDA como método para reducción de dimensionalidad y poder visualizar la información que brinda el dataset, pues se tiene el conjunto de datos de dimensión $535 \cdot 6373$. Se utiliza la librería sklearn de python[21] con la cual se normalizan las características. Para este método se tiene un arreglo el cual obtiene la matriz de características y otro con las etiquetas de cada una de ellas, ya que es necesario para poder clasificar, y obtener la siguiente figura:

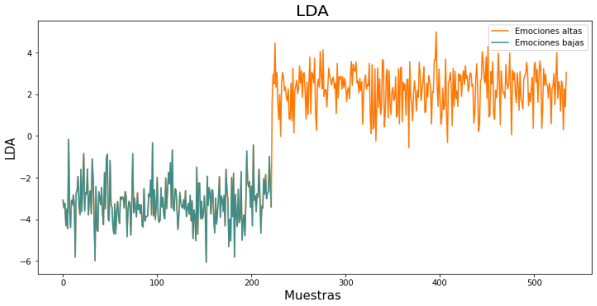

Fig. 17: Grafica de LDA

Se puede observar a partir de la gráfica, que las dos clases de características (emociones altas y bajas), están separadas por color, se puede identificar que la clase que está de color azul (emociones bajas), se tienen en el eje de las muestras desde la muestra 0 hasta la muestra número 222, las cuales son las correctas, y se puede observar que todas dichas muestras están en un rango menor que cero en el eje de LDA, esto quiere decir que su distancia está por debajo del cero y que se esta observando la característica de emociones bajas. Por otra parte las emociones altas se ven reflejadas en el color naranja, el cual se tiene desde las muestra 223 hasta la 535, lo cual está correcto y además su valor de LDA está por encima del cero.

3.2.3 Comparación entre PCA y LDA: Para poder observar de manera gráfica es mejor utilizar PCA que LDA, pues se puede observar de una mejor manera como están los datos organizados en el caso de que se quiera hacer una clasificación, además es beneficioso que PCA se pueda aplicar tanto a datos etiquetados como no etiquetados, ya que no depende de las etiquetas de salida. Por otro lado, LDA requiere clases de salida para encontrar discriminantes lineales y, por lo tanto, requiere datos etiquetados.

3.3 *Clasificación por KNN*

Para clasificar las clases, se utiliza los datos obtenidos con el PCA con 2 componentes, y se realiza un split de la data del 30 % para test (159 muestras) y 70 % para train(376 muestras). Se realiza la clasificación de las clases utilizando KNN en Python [22], y se hace uso de las técnicas de optimización[23][24] para mirar la evolución y rendimiento del modelo.

3.3.1 KNN: En primer lugar, se escoge un K al azar ($k = 5$) para entrenar el modelo y se coge como referencia el gráfico original de los datos con el test de 30 % del PCA y 2 componentes (figura 18), y a partir de dicho gráfico obtener una comparación visual con el gráfico de los datos predichos por el modelo de KNN entrenado (figura 19).

Al analizar las gráficas 18 y 19, se puede observar que la mayoría de los datos están clasificados correctamente en su clase,

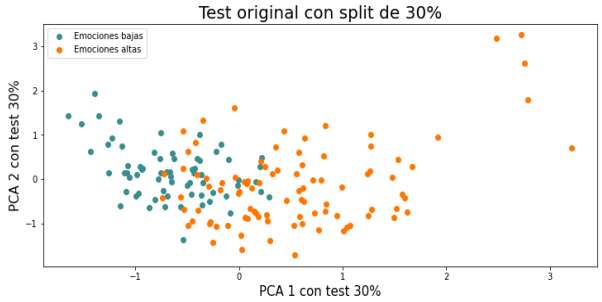

Fig. 18: Gráfico original de PCA del test 30 %

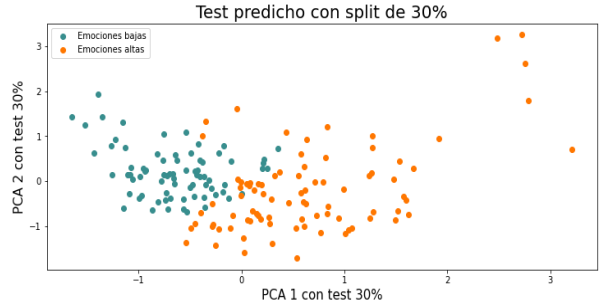

Fig. 19: Gráfico predicho por modelo con PCA del test 30 %

pero existen ciertas diferencias en los gráficos, dado que los vecinos cercanos a evaluar son 5 y KNN al ser un método basado en distancias, hay algunos puntos en los que se tiene cerca más puntos de la otra clase, y provoca que el modelo los clasifique erróneamente. Esto se puede observar analizando las métricas obtenidas:

```
Matriz de confusión: [[58  9]
 [17 77]]
Accuracy:  0.8385093167701864
Sensibilidad:  0.8191489361702128
Precisión:  0.8953488372093024
Especificidad:  0.7733333333333333
```

Fig. 20: Métricas de desempeño del modelo KNN con K=5

Teniendo en cuenta las 159 muestras que se tienen para test, por medio de la matriz de confusión se puede notar que 58 de la clase bajas emociones están correctamente predicho, 77 de la clase altas emociones están correctamente predicho, se tiene 9 de la clase emociones bajas predicho erróneamente como emociones altas, y se tiene 17 de la clase emociones altas predicho de manera equivocada como emociones bajas.

Esto se observa al analizar el Accuracy ya que es aproximadamente del 0.8385,lo cual quiere decir que se tiene un clasificador medianamente bueno, dado se tienen bastantes errores.

3.3.2 KNN usando validación cruzada: Para este caso se entrena el modelo con K-fold cross validation, donde se realiza el entrenamiento con 10 folds en el *dataset* completo, utilizando 30 % para el *test* de los datos, 70 % para *train*, y usando 5 vecinos más cercanos para cada fold. Dado que en cada fold se tiene unas métricas distintas, se saca un promedio de los 10 resultados obtenidos y se procede a calcular la desviación estándar para mirar como es el comportamiento general del modelo con los folds.

```
El promedio de accuracy:  0.8577568134171907 ,La desviación estandar:  0.046555109015879656
El promedio de la sensibilidad:  0.8699757303205577 ,La desviación estandar:  0.07089163759504713
El promedio de Precisión:  0.8883250758089352 ,La desviación estandar:  0.05711632525100198
El promedio de Especificidad:  0.8260283983327461 ,La desviación estandar:  0.09603689991085795
```

Fig. 21: Métricas de KNN con 10 folds

Ahora al analizar el desempeño, se observa que mejora en un pequeño porcentaje con respecto al anterior (sin usar K-fold), se obtiene un Acurracy de aproximadamente un $85{,}77\,\%$con una desviación estándar de $\pm 0{,}0465$, lo cual indica que tan lejos se puede estar de la media del Accuracy, este valor es muy bueno pues se tiene una desviación pequeña y un Accuracy grande.Esto se da gracias a que los datos son entrenados con todos los datos y entonces así, se puede tener más casos para evaluar.

3.3.3 KNN usando K-fold y Gridsearch: Para este caso se usa K-fold con 10 folds y malla de búsqueda para verificar cual K_nearest es el mejor para cada uno de los folds y así obtener el mejor resultado posible.

```
El promedio de accuracy:  0.872921034241789 ,La desviasión estandar:  0.03590290845731309
El promedio de la sensibilidad:  0.859297221724224 ,La desviasión estandar:  0.056566878652549923
El promedio de Precisión:  0.916287593081948 ,La desviasión estandar:  0.036974724551909276
El promedio de Especificidad:  0.8160928292721771 ,La desviasión estandar:  0.07804199381529511
```

Fig. 22: Métricas de KNN con 10 folds y malla de búsqueda

Como se puede observar, el Accuracy mejora con respecto a los 2 obtenidos anteriormente, esto se debe a que la distribución de los datos para cada uno de los folds no es exactamente la misma, el número óptimo de vecinos cambia para cada fold, lo cual hace que se tenga un modelo mejor entrenado y así lograr el Accuracy más alto posible para todos los casos, obteniendo así finalmente aproximadamente un $0{,}8729$ de Accuracy con una desviación estándar de $\pm 0{,}0359$.Dado que se tiene una desviación muy pequeña con respecto al a media, el valor de Accuracy mínimo alcanzado sería del orden de $0{,}83$.

Con la malla de búsqueda se varía el parámetro K de KNN desde 1 hasta 10 vecinos, en donde se encuentra el mejor para cada caso (ver figura 23)

```
El mejor N para cada fold es:  [9. 8. 8. 9. 9. 8. 9. 5. 8. 8.] La moda es:  ModeResult(mode=array([8.]), count=array([5]))
```

Fig. 23: Moda y conteo para el N de KNN de cada fold

La moda que se obtiene es de un $N = 8$ para KNN, el cual aparece 5 veces de los 10 folds, razón por la cual la mejor opción para escoger un K para el entrenamiento del modelo KNN es un $N = 8$.

3.3.4 Diagrama de distribución, curva ROC y AUC: Este diagrama de distribución se hace basado en el KNN utilizando malla de búsqueda y K-fold, es decir se sacan los Scores con respecto a cada uno de los folds y se almacenan en un arreglo de tal manera que luego se puedan graficar. Para para graficar el número de muestras contadas en el diagrama de distribuciones, se asume que es una probabilidad, es decir, se hace la predicción y se evalúa la cantidad de puntos correctos sobre N-neighboors.

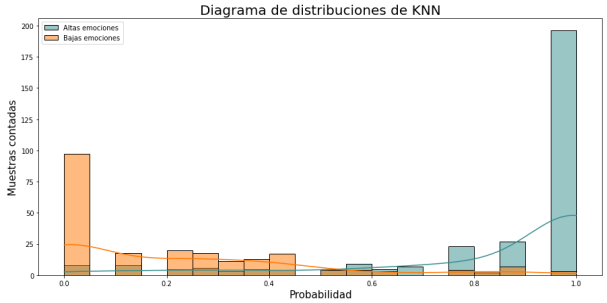

Fig. 24: Diagrama de distribución de KNN

Se puede extraer la curva ROC y AUC del diagrama de distribuciones, lo va a dar una idea de que tan bien funciona en general el modelo de KNN previamente entrenado. El umbral está ubicado en 0.5.

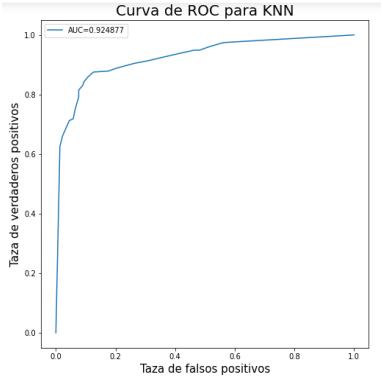

Fig. 25: Curva ROC y AUC de KNN usando k-fold y malla de búsqueda

Como se puede observar en la figura 25, se obtuvo que el área bajo al curva es aproximadamente de 0,9248, lo cual representa un buen modelo, esta cerca del 1 que es ideal, además se puede ver como la curva tiene una curva bastante pronunciada, lo cual da a entender que el modelo tiene buena capacidad de clasificación.

3.4 Clasificación por SVM

Al igual que con KNN, se utiliza el mismo split para test del 30 % de los datos (161 muestras al azar) con un PCA de 2 componentes, y se compara el desempeño del modelo con y sin métodos de optimización. Cabe resaltar que para este entrenamiento se utiliza un kernel Gaussiano (rbf) y se usan los hiperparámetros por defecto de SVM.

3.4.1 SVM: En primer lugar, se realiza el entrenamiento de los datos y se gráfica el test original (figura 26) y el predicho por la máquina vectorial de soportes (figura27) para visualizar como funciona SVM. Se usan los hiperparámetros por defecto de SVM.

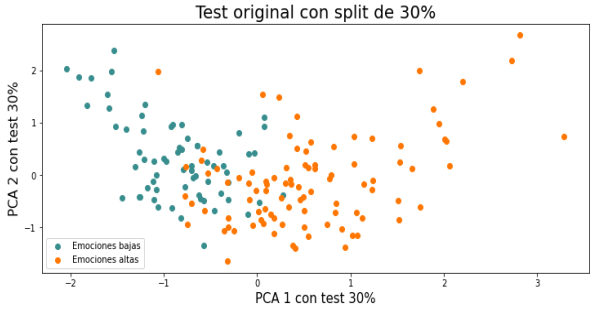

Fig. 26: Datos originales del test 30 % con PCA 2 componentes para SVM

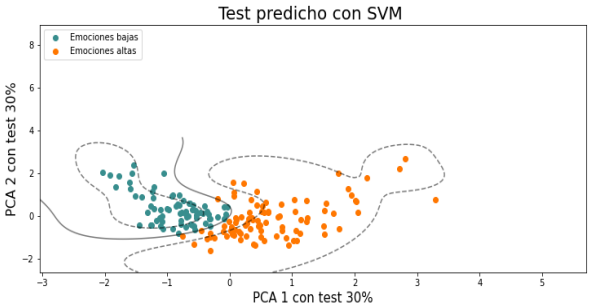

Fig. 27: Datos clasificados por la SVM

Como se puede observar en la figura 27, SVM gráfica los datos clasificados de manera muy visible, los separa a nivel visual. La linea continua que se encuentra es el hiperplano obtenido, y la linea punteada son los margenes de decisión.

A simple vista se puede observar como se hizo la clasificación pero no es posible obtener conclusiones de como es es el desempeño, por lo cual es importante obtener las métricas a diferencia de KNN.

```
Matriz de confusión: [[59  8]
 [13 81]]
Accuracy:  0.8695652173913043
Sensibilidad:  0.8617021276595744
Precisión:  0.9101123595505618
Especificidad:  0.8194444444444444
```

Fig. 28: Métricas de desempeño del modelo con SVM

Teniendo en cuenta las 161 muestras que se tienen para test, por medio de la matriz de confusión se puede notar que 58 de la clase bajas emociones están correctamente predicho, 81 de la

clase altas emociones están correctamente predicho, se tiene 8 de la clase emociones bajas predicho erróneamente como emociones altas, y se tiene 13 de la clase emociones altas predicho erróneamente como emociones bajas.

Esto se observa al analizar el Accuracy ya que es aproximadamente del $0{,}8695$,lo cual quiere decir que se tiene un clasificador bueno (Muy cercano a KNN con optimización), con una taza pequeña de errores.

3.4.2 SVM usando K-fold: Ahora bien, se analiza como es el desempeño para clasificar las dos clases con SVM utilizando 10 folds. se promedia el cada una de las métricas de cada fold y se calcula la desviación estándar para saber que tanto se aleja el modelo de la media tal como se muestra en la figura 29. Se usan los hiperparámetros por defecto de SVM.

```
El promedio de accuracy:  0.8727812718378756 ,La desviación estandar:  0.05732589945189802
El promedio de la sensibilidad:  0.8738481725790358 ,La desviación estandar:  0.07464300030155048
El promedio de Precisión:  0.9022105821965143 ,La desviación estandar:  0.062149745640298466
El promedio de Especificidad:  0.839579443947865 ,La desviación estandar:  0.085925218960614
```

Fig. 29: Métricas de SVM con 10 folds

Se puede observar que el Accuracy es aproximadamente de $0{,}8727$, con una desviación estándar aproximada de $\pm 0{,}0573$. lo cual quiere decir , que se puede obtener un Accuracy mínimo del orden de $0{,}82$, un valor muy similar al que se obtuvo con KNN pero teniendo en cuenta que sería el peor de los casos, esto se debe a que la distribución de los datos es al azar y en algunas ocasiones al aplicar la función kernel, no se tiene el mejor espacio de características transformado para dicho fold.

3.4.3 SVM usando K-fold y malla de búsqueda : Ahora bien, para obtener un desempeño más acertado, teniendo en cuenta el sobre-ajuste y sub-ajuste, se entrena un modelo usando una malla de búsqueda para encontrar los valores más óptimos para los hiperparámetros C y Γ para cada fold.

```
El promedio de accuracy:  0.8689028651292802 ,La desviación estandar:  0.0514673510562656
El promedio de la sensibilidad:  0.8714139384708645 ,La desviación estandar:  0.05524241327988556
El promedio de Precisión:  0.9080905551767622 ,La desviación estandar:  0.057240590542618736
El promedio de Especificidad:  0.8183129227510786 ,La desviación estandar:  0.08662657539961724
```

Fig. 30: Métricas de SVM con 10 folds

Se observa que se tiene un Accuracy aproximado de 0.8689 con una desviación estándar de 0.0514, lo cual indica que se tiene un desempeño bueno del modelo para clasificar las 2 clases, pero este Accuracy es menor que el obtenido únicamente con K-fold, esto no quiere decir necesariamente que el modelo este mal entrenado, sino más bien que al tener los hiperpárametros correctos el modelo es capaz de tener un desempeño que evita un sobre-ajuste, dando como resultado un desempeño más bajo pero con mayor capacidad de evitar errores. Esto se puede realizar observando las demás métricas, dado que al obtener datos al azar se puede tener un fold que esté desbalanceado y genera un Accuracy bajo, pero pese a eso, el modelo está clasificando lo mejor posible.

```
El vector de los valores C para cada fold: [ 1.   1. 100.   1. 100. 100. 100.   0.1 100. 100. ] , La moda es:  ModeResult(mode=array([100.]), count=array([6]))
El vector de los valores gamma para cada fold: [1. 1. 1. 1. 1. 1. 1. 1. 0.1 1. ]  La moda es:  ModeResult(mode=array([1.]), count=array([9]))
```

Fig. 31: Métricas de SVM con 10 folds

La moda del hiperparámetro C es el valor de 100 y aparece 6 veces dentro de los folds, y para el Γ la moda es 1.0 el cual aparece 9 veces. Esto quiere decir que el modelo se puede entrenar con estos valores para obtener un buen resultado.

3.4.4 Diagrama de distribuciones, curva ROC y AUC para SVM: Otra manera de identificar si el modelo tiene buena clasificación es utilizando el diagrama de distribuciones, con el cual se obtiene la curva ROC y AUC del modelo, para el caso del clasificador SVM, el diagrama de distribuciones es la relación entre las muestras contadas y la distancia al hiperplano de cada una.

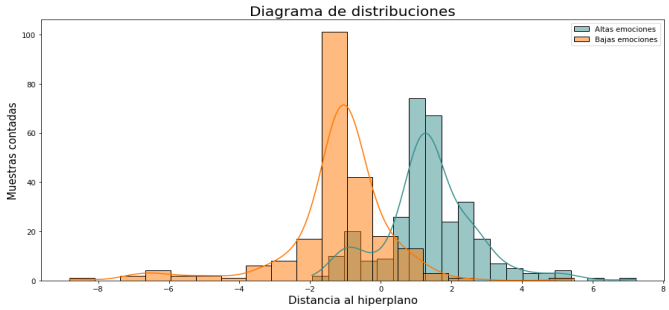

Fig. 32: Diagrama de distribuciones de SVM

Se puede notar que los datos tienden a estar distribuidos en forma de campana, esto se da porque los datos están normalizados y eso hace que se comporten de esta manera. A partir de este diagrama se puede obtener la curva ROC donde el umbral está ubicado en 0.

Al analizar la figura 33, se observa que se obtiene un área bajo la curva de aproximadamente $0{,}9296$, lo cual representa un modelo con alto desempeño de clasificación.

4 Conclusiones

- En el caso de datos distribuidos uniformemente, LDA en la mayoría de los casos obtiene mayor precisión que PCA. Sin embargo, si los datos están muy sesgados o se tiene un dataset desbalanceado (distribuidos de manera irregular), se recomienda utilizar PCA ya que LDA puede estar sesgado hacia la clase mayoritaria.
- Para SVM, si gamma es grande, el efecto de c es insignificante. Si gamma es pequeño, c afecta al modelo de la misma manera que afecta a un modelo lineal.

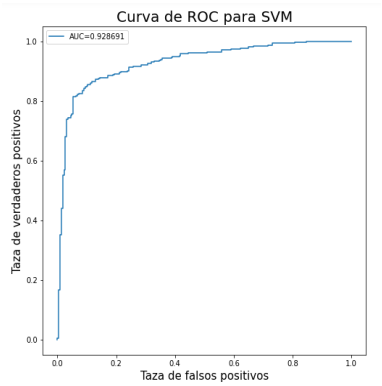

Fig. 33: Curva ROC y AUC del clasificador SVM con k-fold y malla de búsqueda

- La métrica de Accuracy puede ser engañosa, ya que si se tiene un conjunto de muestras desbalanceado, el Accuracy es alto a pesar de que se estén realizando predicciones incorrectas, un ejemplo de esto es que si se tiene el 90 % de las muestras de la clase positiva y solo 10 % de la clase negativa, el modelo de clasificación podría decir que todas las muestras del sistema son de clase positiva y aun así obtener un accuracy de 0.9 , pero las muestras de la clase negativa fueron incorrectamente clasificadas.
- El hecho de tener mejor Accuracy no significa tener mejor desempeño, pues toca evaluar que el modelo no tenga sobre-ajuste y toca tener en cuenta que los datos que se tienen, estén correctamente balanceados en cantidad para ambas clases.
- No todas las veces que se realice un entrenamiento se va obtener el mismo desempeño puesto que los datos escogidos son de manera aleatoria si se está utilizando o no, el parámetro shuffle de la librería.
- La validación cruzada es una herramienta que permite obtener mejor entrenamiento dado que se prueban todos los datos del dataset y así no se está perdiendo posible información que puede ser beneficiosa para el modelo, además permite evitar el sobre y sub-juste.

5 References

[1] Reynaga, R. and Mayta, W., 2022. INTRODUCCION AL RECONOCIMIENTO DE PATRONES. [online] Scielo.org.bo. Available at: <http://www.scielo.org.bo/scielo.php?script=sci$_arttextpid = S2071-081X2009000100005 >$ $[Accessed 26 September 2022]$.

[2] Tomás Arias Vergara, Universidad de Antioquia, Departamento de ingeniería electrónica y telecommunications, Presentación de las diferentes etapas de un sistema de análisis de patrones, (2016)

[3] Abdi, H., Williams, L. J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4), 433–459. doi:10.1002/wics.101

[4] Ringnér, M. What is principal component analysis?. Nat Biotechnol 26, 303–304 (2008). https://doi.org/10.1038/nbt0308-303

[5] StatQuest with Josh Starmer. StatQuest: Principal Component Analysis (PCA), Step-by-Step. [Online] https://www.youtube.com/watch?v=FgakZw6K1QQt=281s

[6] Zakaria Jaadi,A Step-by-Step Explanation of Principal Component Analysis (PCA), [Online] https://builtin.com/data-science/step-step-explanation-principal-component-analysis, (2022)

[7] Tharwat, A., Gaber, T., Ibrahim, A., Hassanien, A. E. (2017). Linear discriminant analysis: A detailed tutorial. AI Communications, 30(2), 169–190. doi:10.3233/aic-170729

[8] sebastián raschka,Análisis Discriminante Lineal (2014), [online]: https://sebastianraschka.com/Articles/2014$_python_lda.html$

[9] StatQuest with Josh Starmer.StatQuest: Linear Discriminant Analysis (LDA) clearly explained,[Online]:https://www.youtube.com/watch?v=azXCzI57Yfcab$_channel = StatQuestwithJoshStarmer$

[10] IBM,Algoritmo de k vecinos más cercanos, [Online]: https://www.ibm.com/co-es/topics/knn

[11] Ligdi Gonzalez,K Vecinos más Cercanos – Teoría, [Online]: https://aprendeia.com/algoritmo-k-vecinos-mas-cercanos-teoria-machine-learning/

[12] Pisner, D. A., Schnyer, D. M. (2020). Support vector machine. Machine Learning, 101–121. doi:10.1016/b978-0-12-815739-8.00006-7

[13] Rohith Gandhi,(2018),Support Vector Machine — Introduction to Machine Learning Algorithms, [Online]:https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[14] Soner Yıldırım,(2020),Hyperparameter Tuning for Support Vector Machines — C and Gamma Parameters, [Online]:https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167

[15] P. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, New York, EEUU: Pearson Education, 2006

[16] Leticia L.Ochoa,Universidad Nacional de San Agustín de Arequipa, Perú, (2019),Evaluation of Classification Algorithms using Cross Validation

[17] Shatnawi, Amjed Alkassar, Hana Moneem, Nadia A. Al-Hamdany, Emadaldeen Luí, Bernardo, Filipe Imran, Hamza. (2022). Shear Strength Prediction of Slender Steel Fiber Reinforced Concrete Beams Using a Gradient Boosting Regression Tree Method. Buildings. 1.1 MG.

[18] Terence S,Comprensión de la Matriz de Confusión y Cómo Implementarla en Python, (2020), [Online]:https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

[19] Powers, David M. W. (2011). .Evaluation: From

Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation". Journal of Machine Learning Technologies. 2 (1): 37–63. [Online]:https://www.researchgate.net/publication/228529307_Evaluation_From_Precision_Recall_and_F-Factor_to_ROC_Informedness_Markedness_Correlation

[20] Sckit learn, sklearn.decomposition.PCA,[Online] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[21] Sckit learn, sklearn.discriminant_analysis, [Online] https://scikit-learn.org/stable/modules/generated/sklearn.discriminan _analysis.LinearDiscriminantAnalysis.html

[22] Sckit learn, sklearn.neighbors.KNeighborsClassifier, [Online] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[23] Sckit learn, sklearn.model_selection.GridSearchCV, [Online] https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.GridSearchCV.html$

[24] Sckit learn, sklearn.model_selection.KFold, [Online] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html