Universidad Autónoma de Querétaro

Facultad de Ingeniería
Maestría en Inteligencia Artificial
Machine Learning
Sheila Leyva López
Cecilia Gabriela Rodríguez Flores

Análisis de Componentes Principales (PCA)

18 de noviembre del 2022

Objetivo

Desarrollar un algoritmo enfocado en la reducción de dimensionalidad de los datos, mediante el lenguaje de programación de python, a fin de tener una mejor interpretabilidad y por ende ser capaz de realizar la tarea de clasificación de enfermedad cardíaca.

Introducción

Dentro del área de la Inteligencia Artificial (IA) existe una sub-área llamada aprendizaje automático o Machine Learning (ML), cuyas herramientas permiten a un sistema aprender patrones y comportamientos de los datos en lugar de aprender mediante la programación explícita. Asimismo, existen distintos tipos de ML: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado y aprendizaje por refuerzo. Específicamente, hay un tipo de algoritmo de aprendizaje no supervisado conocido como análisis de componentes principales (PCA), el cual es utilizado en una variedad de aplicaciones, como son: análisis exploratorio de datos, reducción de dimensionalidad, comprensión de información, eliminación de ruido de datos, etc.

En el presente trabajo se desarrollan las funciones, en lenguaje de Python, necesarias para la predicción de clases de cualquier base de datos, sin embargo, toma como referencia la base de datos: *Indicadores personales clave de enfermedad cardíaca*.

Marco Teórico

Es importante definir conceptos básicos implementados en este trabajo a fin brindar un mejor entendimiento del contexto en el que se trabaja.

Análisis de componentes principales (PCA)

El análisis de componentes principales (PCA) simplifica la complejidad de los datos de alta dimensión al tiempo que conserva tendencias y patrones, en otras palabras, es básicamente un procedimiento estadístico para convertir un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables linealmente no correlacionadas, en donde primeramente se realiza una transformación lineal de los datos en un nuevo sistema de coordenadas ortogonales. En este nuevo sistema de coordenadas las componentes principales están ordenadas automáticamente según la varianza de la proyección de datos, es decir, según la cantidad de información que contengan. Debido a esto, se puede

reducir la dimensión de los datos resultantes en el nuevo espacio eliminando las componentes principales que presenten una menor varianza, es decir, que aporten menos información.

Para poder realizar esta reducción de dimensionalidad con PCA, se tienen que realizar los siguientes pasos:

- Obtener lo datos, en todas sus dimensiones
- Restar la media. Para que PCA funcione de manera correcta, se requiere restar la media de cada dimensión.
- Calcular la matriz de covarianza
- Calcular los eigenvalores y los eigenvectores de la matriz de covarianza
- Escoger los componentes y formar el vector de características, en donde el elemento con el número mayor es el componente principal del set de datos.
- En general, una vez que los eigenvectores se encuentran en la matriz de covarianza, se ordenan de mayor a menor dando estos componentes principales en orden

Así bien, PCA ofrece múltiples beneficios, pero también adolece de ciertas deficiencias.

Ventajas:

1. Fácil de calcular. PCA se basa en álgebra lineal, que es computacionalmente fácil de resolver por computadoras.
2. Acelera otros algoritmos de aprendizaje automático. Los algoritmos de aprendizaje automático convergen más rápido cuando se entrenan en los componentes principales en lugar del conjunto de datos original.
3. Contrarresta los problemas de datos de alta dimensión. Los datos de alta dimensión hacen que los algoritmos basados en regresión se sobreajusten fácilmente. Al usar PCA de antemano para reducir las dimensiones del conjunto de datos de entrenamiento, evitamos que los algoritmos predictivos se sobreajusten.

Desventajas:

1. Baja interpretabilidad de los componentes principales. Los componentes principales son combinaciones lineales de las características de los datos originales, pero no son tan

fáciles de interpretar. Por ejemplo, es difícil saber cuáles son las características más importantes del conjunto de datos después de calcular los componentes principales.

2. La compensación entre la pérdida de información y la reducción de la dimensionalidad. Aunque la reducción de la dimensionalidad es útil, tiene un costo. La pérdida de información es una parte necesaria de PCA.
3. PCA asume una correlación entre características. Si las características no están correlacionadas, PCA no podrá determinar los componentes principales.
4. PCA es sensible a la escala de las características. Al suponer que se cuenta con dos características: una toma valores entre 0 y 1000, mientras que otra toma valores entre 0 y 1. PCA estará extremadamente sesgado hacía que la primera característica sea el primer componente principal, independientemente de la variación máximo real dentro de los datos. Por lo cual, es importante estandarizar los valores primeros.
5. PCA no es robusto frente a valores atípicos. Similar al punto anterior, el algoritmo estará sesgado en conjuntos de datos con fuertes valores atípicos.
6. Las implementaciones técnicas a menudo no asumen valores perdidos. Al calcular PCA utilizando herramientas de software estadístico, a menudo asumen que el conjunto de características no tiene valores perdidos.

K-vecinos más cercanos

El algoritmo K-vecinos más cercanos (KNN) es un tipo de algoritmo de aprendizaje automático supervisado que se utiliza para la clasificación, la regresión y la detección de valores atípicos. Es extremadamente fácil de implementar en su forma más básica, pero puede realizar tareas bastante complejas. Es un algoritmo de aprendizaje perezoso ya que no tiene una fase de entrenamiento especializada. Más bien, utiliza todos los datos para el entrenamiento mientras clasifica (o retrocede) un nuevo punto de datos o instancia.

KNN es un algoritmo de aprendizaje no paramétrico, lo que significa que no asume nada sobre los datos subyacentes. Esta es una característica extremadamente útil ya que la mayoría de los

datos del mundo real no siguen ningún supuesto teórico, por ejemplo, separabilidad lineal, distribución uniforme, etc.

Etapas en la implementación de modelos

Es importante definir conceptos básicos del aprendizaje máquina que permitan entender el contexto en el que se trabaja, como son las etapas fundamentales para la implementación de técnicas de aprendizaje máquina: Preparación de datos, la cual es una etapa fundamental en cualquier proyecto, debido a que por medio de ella se pueden inicializar correctamente los datos para su posterior procesamiento y análisis; Analizar, en esta sección se utilizaran (que dato se van a utilizar) o dividirá los datos ya preparados para el siguiente paso, para este trabajo se optó por crear subconjuntos de datos, en los cuales el 80% de los datos se considera para el entrenamiento, mientras que un 20% corresponde a pruebas del modelo de clasificación.

Métricas

La manera más simple de evaluar un modelo con características nominales y discretas es por medio de diferentes métricas, como son:

Exactitud

La exactitud es la relación entre los simples correctamente clasificados y el número total de simples en el conjunto de datos de evaluación. Esta métrica es conocida por ser engañosa en el caso de diferentes proporciones de clases, ya que asignar simplemente todos los simples a la clase predominante es una forma fácil de lograr una alta precisión [3].

$$ACC = \frac{\#\ correctly\ classified\ samples}{\#\ all\ samples} = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

Precisión

La precisión es la capacidad de un modelo para identificar sólo objetos relevantes, es decir, es el porcentaje de predicciones positivas correctas entre todas las verdades básicas dadas [3].

$$P_r = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ detection} \quad (2)$$

Sensitividad

La sensitividad o recall se define como el porcentaje de predicciones correctas tomadas de la clase de predicciones correctas sin tomar en cuenta los falsos positivos[3].

$$REC = \frac{\#\ true\ positive\ samples}{\#\ samples\ classified\ positive} = \frac{TP}{TP + FN} \quad (3)$$

F1 Score

La F1 Score calcula la media armónica de la precisión y recall de forma que se enfatiza al valor más bajo entre ambas [3].

$$F1 = 2\,x\,\frac{precision\ x\ recall}{precision + recall} = \frac{2\,x\,TP}{2\,x\,TP + FP + FN} \quad (4)$$

Materiales y métodos

Herramientas utilizadas

- Google Collaboratory
- Conjunto de datos: *Indicadores personales clave de enfermedad cardíaca*
- AMD Ryzen 9 5900HS with Radeon Graphics 3.30 GHz
- RAM 16.0 GB
- 64-bit operating system, x64-based processor

Conjunto de datos

En este trabajo, se utilizará el conjunto de datos: *Indicadores personales clave de enfermedad cardíaca*, datos provenientes de 400,000 adultos, obtenidos durante la encuesta anual 2020 de los Centros para el Control y prevención de Enfermedades (CDC, por sus siglas en inglés) pertenecientes al departamento de salud y servicios humanos en los Estados Unidos. Originalmente el conjunto de datos contenía alrededor de 300 atributos, sin embargo, se redujo a solo 18 variables, los cuales son los que se encuentran disponibles públicamente en la plataforma Kaggle [4].

Casi la mitad de los estadounidenses (47%), incluyendo afroamericanos, indios americanos, nativos de Alaska y blancos; tienen al menos de 1 a 3 factores de riesgo de padecer alguna enfermedad cardíaca. A continuación, se agrega una breve descripción de los atributos incluidos en este conjunto de datos:

- HeartDisease: (atributo de decisión): personas encuestadas que informaron alguna vez haber padecido alguna enfermedad coronaria (CHD, por sus siglas en inglés) o infarto al miocardio(IM, por sus siglas en inglés).
- BMI: Índice de Masa Corporal.
- Smoking: personas encuestadas que han fumado al menos 100 cigarros en su vida entera.
- AlcoholDrinking: corresponde a hombres adultos que beben más de 14 tragos por semana y mujeres adultas que beben más de 7 tragos por semana.
- Stroke: responde a la pregunta: ¿alguna vez le dijeron o usted tuvo un derrame cerebral?
- PhysicalHealth: incluyendo enfermedades y lesiones físicas, responde a la pregunta: ¿durante cuántos días en los últimos 30 días su salud física no fue buena? (de 0 a 30 días)
- MentalHealth: ¿durante cuántos días en los últimos 30 días su salud mental no fue buena? (de 0 a 30 días).
- DiffWalking: responde a ¿tiene serias dificultades para caminar o subir escaleras?
- Sex: hombre o mujer.
- AgeCategory: 14 rangos de edad.
- Race: valor de raza / etnicidad imputada.
- Diabetic: responde a ¿alguna vez ha sido diagnosticada con diabetes?
- PhysiclActivity: adultos que informaron haber realizado actividad física o ejercicio en los últimos 30 días, no incluyendo su trabajo habitual.
- GenHealth: responde a ¿cómo calificarías tu salud en general?
- SleepTime: responde a un promedio de horas que duerme, en un periodo de 24 horas, la persona encuestada.
- Asthma: responde a ¿alguna vez ha sido diagnosticado con asma?

- KidneyDisease: responde a ¿alguna vez le dijeron que tenía una enfermedad renal?, sin incluir cálculos renales, infección de vejiga o incontinencia.
- SkinCancer: responde a ¿alguna vez ha sido diagnosticado de cáncer de piel.

Métodos

Para la realización de esta tarea, fue necesario importar las librerías de Pandas, numpy, etc., así como el montaje de Google Drive en el entorno de ejecución a fin de poder hacer uso de la base de datos propuesta.

Posteriormente, se optó por crear diversas funciones a fin de distribuir las tareas de una manera más eficiente como se muestra a continuación:

- Función norm_min_max: brinda la base de datos normalizada. Esta función requiere como datos de entrada la base de datos propuesta.

```
def norm_min_max(datos):
  lim_sup = []
  lim_inf = []
  rangoDatos =[]
  maxNorm = 1
  minNorm = 0
  rango = maxNorm - minNorm
  for i in range (0,datos.columns.size):
    lim_sup.append(datos.iloc[:,i].max())
    lim_inf.append(datos.iloc[:,i].min())
    rangoDatos.append(lim_sup[i] - lim_inf[i])
  nombres = datos.columns.values.tolist()
  datosNorm = pd.DataFrame(columns = nombres)

  for j in range(len(datos.columns)):
    varNorm = []
    var = datos.iloc[:,j]
    for i in range(len(datos)):
      D = var[i] - lim_inf[j]
      DPct = D/rangoDatos[j]
      dNorm = rango*DPct
      varNorm.append(minNorm+dNorm)
    datosNorm.iloc[:,j] = varNorm
  datos = datosNorm
  return datos
```

Ilustración 1.Función propuesta para la obtención de la normalización.

- Función matriz_cov: se encarga de obtener la matriz de covarianza de los elementos de la base de datos en cuestión.

```
def matriz_cov(data):
    atributos = data.columns
    n = len(atributos)
    m = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            X = data[atributos[i]]
            Y = data[atributos[j]]
            m[i][j] = (((X-X.mean())*(Y-Y.mean())).sum())/(len(X)-1)
    return m
```

Ilustración 2.Función propuesta para la obtención de la matriz de covarianza.

- Función PCA: brinda los porcentajes de cada uno de los atributos de acuerdo con su relevancia dentro de la base de datos, a fin de discernir los atributos con mayor peso.

```
def PCA(datos,col_decision):
  datos1 = datos.drop([col_decision],axis=1) #Eliminando el atributo de decisión
  #Ajustar los datos restando la media a cada atributo
  datos_A = pd.DataFrame(columns=datos1.columns,index=range(len(datos1)))
  for i in datos_A.columns:
      datos_A[i] = datos1[i] - datos1[i].mean()
  #datos_A
  matrix = matriz_cov(datos_A)
  #sns.heatmap(matrix)
  L,V =  np.linalg.eig(matrix)
  #Obtener el porcentaje de covarianza de cada uno de los atributos
  total = L.sum()
  p = (L/total)*100
  pca =[]
  columnas1 = datos_A.columns.values
  for index, row in enumerate(p):
    print(columnas1[index] + ':',row)
```

Ilustración 3. Función propuesta para PCA.

- Función method_8020: realiza la separación del conjunto de datos de acuerdo con el porcentaje de 80% para el conjunto de entrenamiento y 20% para el conjunto de prueba. Esta función requiere como entradas los datos y las etiquetas.

```
def method_8020(x,y):
  train_x = x[0 : int(len(x)*0.8)]
  train_y = y[0 : int(len(y)*0.8)]
  test_x = x[int(len(x)*0.8) : ]
  test_y = y[int(len(y)*0.8) : ]
  return train_x, train_y, test_x, test_y
```

Ilustración 4.Función propuesta para la división del conjuntos de datos.

- Función euclidean_distance: realiza la obtención de la distancia euclidiana. Esta función requiere como entradas los valores de la columna 1 y 2.

```
def euclidean_distance(row1, row2):
  distance = 0.0
  for i in range(len(row1)-1):                    #for k in range(23)
    distance += (row1[i] - row2[i])**2
  return sqrt(distance)
```

Ilustración 5.Función propuesta distancia euclidiana.

- Función get_neighbors: realiza la extracción de los n vecinos más cercanos, de acuerdo a valor de k propuesto. Esta función requiere como entradas el número de n vecinos, la lista de distancias ordenadas.

```
def get_neighbors(k,distances_ord):
  neighbors = list()
  indices = list()
  for i in range(k):
    neighbors.append(distances_ord[i][0])
    indices.append(distances_ord[i][1])
  return neighbors,indices
```

Ilustración 6.Función propuesta para obtener los vecinos cercanos.

- Función most_common: realiza el conteo de la cantidad de ceros y unos. Esta función requiere los valores de la columna de etiquetas.

```
def most_common(output_values):
    return max(set(output_values), key=output_values.count)
```

Ilustración 7.Función propuesta most_common.

- Función predict_classification: realiza el método de knn. Esta función requiere como entradas los datos, las etiquetas y el número de n vecinos.

```
def predict_classification(x,y, k):
    predict = []
    for j in range(0,len(x)):
        # Inicializacion de las distancias.
        distances = []
        x = np.array(x)
        for i ,example in enumerate(x):
            distance = euclidean_distance(example,x[j])# Calculate the Euclidean distance between two vectors
            distances.append((distance, i))
        distances.pop(j)
        distances_ord = sorted(distances)
        neighbors,indices = get_neighbors(k,distances_ord)
        output_values = y.iloc[indices]
        output_values = output_values.to_numpy().tolist()
        predict.append(most_common(output_values))
    return predict
```

Ilustración 8.Función propuesta para el método de knn.

- Función métricas: realiza la evaluación del modelo por medio de diversas métricas. Esta función requiere como entradas los datos predichos y los datos reales.

```
def metricas(claseP,true_train):
    TP = 0
    FP = 0
    TN = 0
    FN = 0
    #Determinar los TP,TN,FP y FN para la matriz de confusión del entrenamiento
    for i in range(len(claseP)):
        if (true_train[i] == claseP[i]) and true_train[i] == 1:
            TP += 1
        elif (true_train[i] == claseP[i]) and true_train[i] == 0:
            TN += 1
        elif (true_train[i] != claseP[i]) and true_train[i] == 0:
            FP += 1
        else:
            FN += 1

    accuracy = ((TP+TN)/(TP+TN+FP+FN))

    if TP + FP != 0:
        precision = TP/(TP+FP)
    else:
        precision = 0
    if TP + FN != 0:
        sensitividad = TP/(TP+FN)
    else:
        sensitividad = 0
    if precision != 0 and sensitividad != 0:
        f1 = (2*TP)/(2*TP+FP+FN)
    else:
        f1 = 0
    return [accuracy, precision, sensitividad, f1]
```

Ilustración 9.Función propuesta para la evaluación del modelo.

Diagrama de metodología

Para el análisis de cualquier base de datos se deben seguir los siguientes pasos:

1. Recolección: Definir y cargar la base de datos a utilizar.
2. Preparar: Convertir las características lingüísticas a valores categóricos, comprobar la no existencia de datos faltantes.
3. Analizar: conocer la distribución de la información por cada atributo, normalizar los datos, reducción de dimensionalidad.
4. Dividir los conjuntos de entrenamiento y prueba del modelo, asignando el 80% de ellos para el entrenamiento y el 20% restante a la etapa de prueba.
5. Selección e implementación de un algoritmo.
6. Por último, se calculan las métricas de exactitud y precisión para conocer el desempeño del algoritmo.

A continuación, se muestra el diagrama de flujo que representa el proceso de desarrollo de la presente tarea, el cual fue implementado en un programa basado en lenguaje Python.

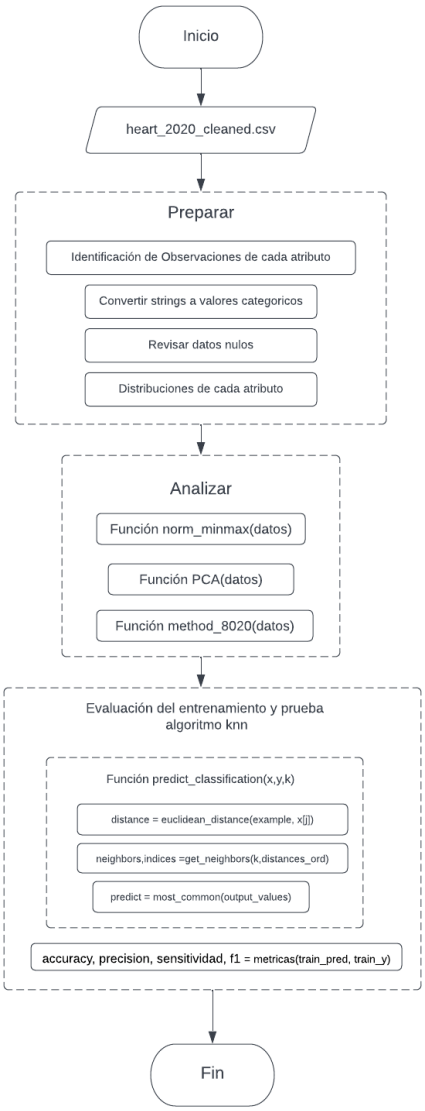


Ilustración 10.Diagrama de flujo para el análisis del conjunto de datos.

Resultados y discusión

A partir de las funciones desarrolladas, con las cuales se analizaron y se calcularon los componentes principales del conjunto de datos, se realizó una comparación entre el modelo de clustering KNN utilizando PCA y no utilizándolo.

Inicialmente, durante la preparación de los datos se obtuvo como resultado la codificación mostrada en las dos siguientes tablas.

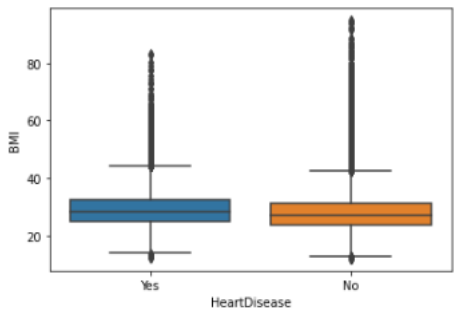
Tabla 1. Codificación de string a número para el atributo Race.

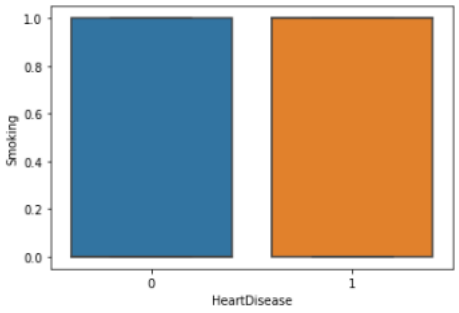
Categórico	Numérico
White	0
Black	1
Asian	2
American Indian/ Alaskan Native Race	3
Other	4
Hispanic	5

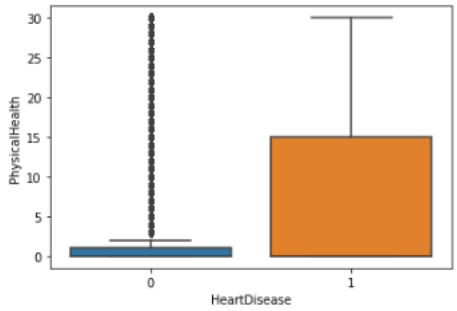
Tabla 2. Codificación de string a número para el atributo Diabetic, Smoking, AlcoholDrinking, Stroke, DiffWalking, PhysicalActivity, Asthma, KidneyDisease, SkinCancer...

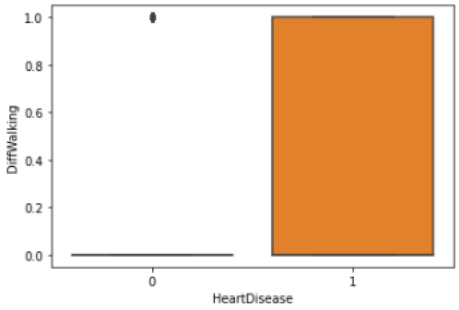
Categórico	Numérico
No	0
Yes	1
No, borderline diabetes	2
Yes (during pregnancy)	3

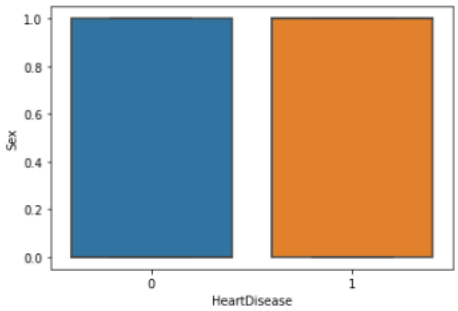
En segundo lugar, se llevó a cabo la identificación de valores faltantes y de outliers a fin de decidir si es necesario la implementación de alguna técnica de imputación en ella, en donde se observó que la base datos en cuestión no cuenta con valores faltantes. A continuación, se muestran las gráficas de cajas de los atributos que contiene el conjunto de datos, a partir de las cuales resulta más sencillo observar si existen outliers.

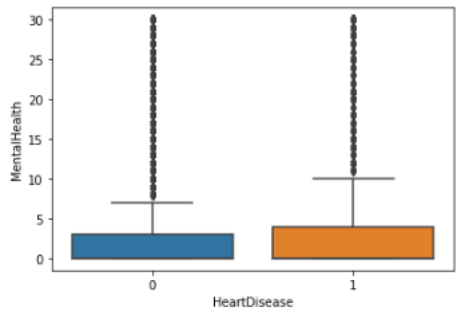

BMI
80
60
40
20
Yes
No
HeartDisease

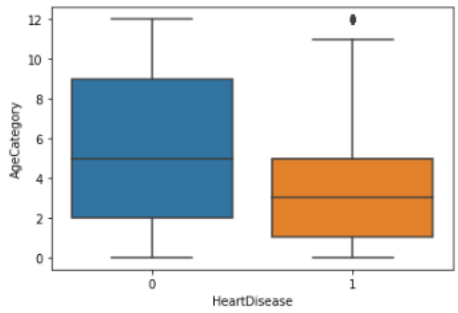

Smoking
1.0
0.8
0.6
0.4
0.2
0.0
0
1
HeartDisease

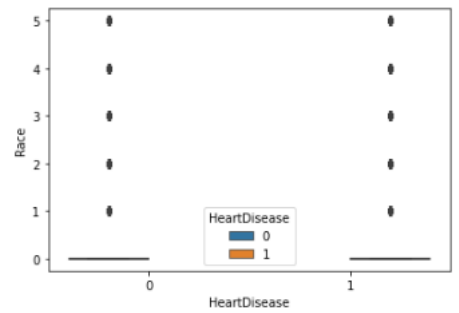

PhysicalHealth
30
25
20
15
10
5
0
0
1
HeartDisease

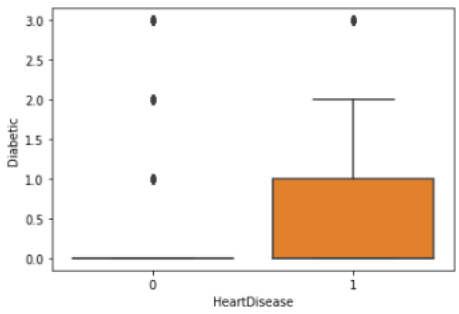

DiffWalking
1.0
0.8
0.6
0.4
0.2
0.0
0
1
HeartDisease

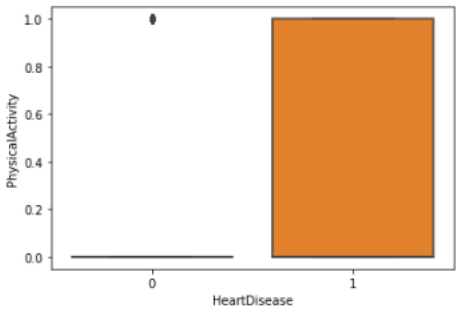

Sex
1.0
0.8
0.6
0.4
0.2
0.0
0
1
HeartDisease

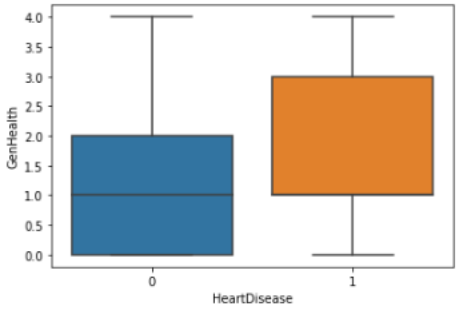

MentalHealth
30
25
20
15
10
5
0
0
1
HeartDisease

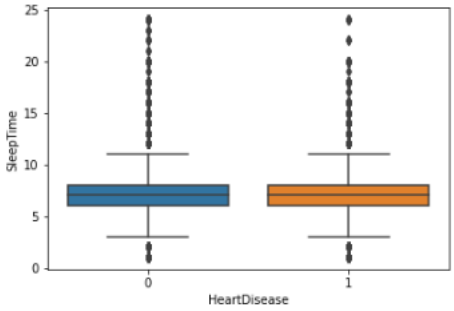

AgeCategory
12
10
8
6
4
2
0
0
1
HeartDisease

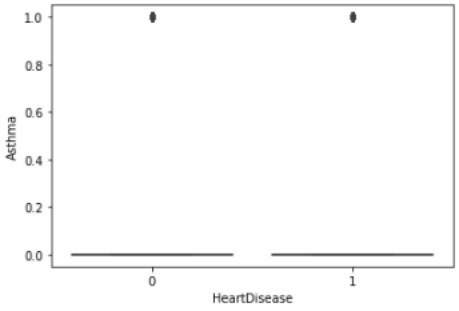

Race
5
4
3
2
1
0
HeartDisease
0
1
0
1
HeartDisease

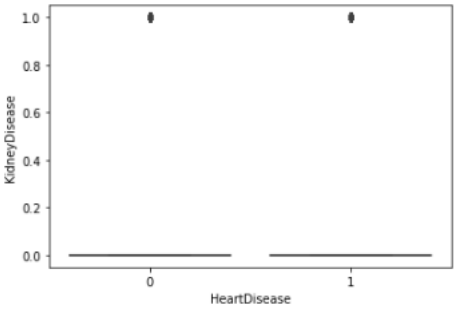

Diabetic
HeartDisease
0
1
0.0
0.5
1.0
1.5
2.0
2.5
3.0

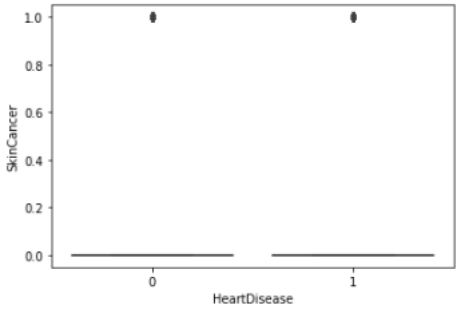

PhysicalActivity
HeartDisease
0
1
0.0
0.2
0.4
0.6
0.8
1.0


GenHealth
HeartDisease
0
1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


SleepTime
HeartDisease
0
1
0
5
10
15
20
25


Asthma
HeartDisease
0
1
0.0
0.2
0.4
0.6
0.8
1.0


KidneyDisease
HeartDisease
0
1
0.0
0.2
0.4
0.6
0.8
1.0


SkinCancer
HeartDisease
0
1
0.0
0.2
0.4
0.6
0.8
1.0

En el caso de los atributos BMI, MentalHealth y SleepTime, se pudo observar que estas columnas cuentan grandes cantidades de valores atípicos o outliers. Sin embargo, cada uno de estos valores atípicos no fueron originados por algún tipo de error humano, por lo cual, se preservará cada uno de ellos.

Por otro lado, con respecto al balance de clase se puede decir que existe un desbalance de clase con un 91.44% para la clase 'No' y un 8.5% para la segunda clase 'Yes', siendo el atributo de decisión: *HeartDisease*.

```
1 Balance_clases(df)
```

```
¿Cuál es tu atributo de decisión?:HeartDisease
No     91.440454
Yes     8.559546
Name: HeartDisease, dtype: float64
El dataset cuenta con 2 clases y son las siguientes: ['No', 'Yes']
```

Ilustración 11.Balance de clases.

Ahora bien, antes de mostrar los resultados obtenidos al implementar el algoritmo de PCA es importante mencionar que, por cuestiones de tiempo y costo computacional solo se llevó a cabo la etapa de entrenamiento y prueba con 10,000 y 20,000 datos de los 319,795 datos que presenta originalmente la base de datos en cuestión. Dado que, el utilizar todos los datos tomaría días incluso semanas.

```
error_rate = []

n = int(len(train_x)/20)
for k in range(2,n):
    pred_i = predict_classification(train_x.values.tolist(),train_y,k)
    #print(pred_i)
    error_rate.append(np.mean(pred_i != train_y))
```

```
11%|          | 21672/191877 [9:48:47<66:51:41,  1.41s/it]
```

Ilustración 12.Tiempo de ejecución utilizando la totalidad de los datos.

A continuación, se muestra en la siguiente tabla los resultados obtenidos al implementar el algoritmo de PCA, en donde se pueden observar cada uno de los porcentajes de covarianza de cada uno de los atributos de la base de datos, en donde se obtuvo que los únicos atributos que aportan mayor información son 10 de 17 atributos: BMI, Smoking, AlcoholDrinking, Stroke,

PhysicalHealth, MentalHealth, AgeCategory, Diabetic, PhysicalActivity, GenHealth, SleepTime sumando una ganancia por encima del 90%.

Tabla 3. Porcentaje de covarianza de cada uno de los atributos.

Atributo	Porcentaje
BMI	**18.967619338273803**
Smoking	**17.469894312255377**
AlcoholDrinking	**12.538267815519616**
Stroke	**8.341922886205534**
PhysicalHealth	**8.181814868924251**
MentalHealth	**6.398673105506219**
AgeCategory	**4.854727259823212**
Diabetic	**4.081222739746422**
PhysicalActivity	**3.788742891015641**
GenHealth	**3.552435740903239**
SleepTime	**2.8745883946506128**
Asthma	2.3593203249524053
KidneyDisease	2.228498248851065
SkinCancer	2.138640449193989
Race	1.6256240537731736
DiffWalking	0.34915183715675346
Sex	0.24885573324868074

Enseguida se muestran los resultados obtenidos utilizando todos los atributos que contiene el conjunto de datos, posteriormente se muestran los resultados al implementar el algoritmo de PCA.

Clasificación sin PCA

Como resultado de un barrido de valores de k para encontrar el valor, tal que, disminuya la tasa de error en la clasificación del algoritmo KNN, se obtuvo la siguiente gráfica, en donde se puede observar el incremento del error cuando aumenta el número de k vecinos.

Utilizando 10,000 datos

Se obtuvieron los siguientes resultados:

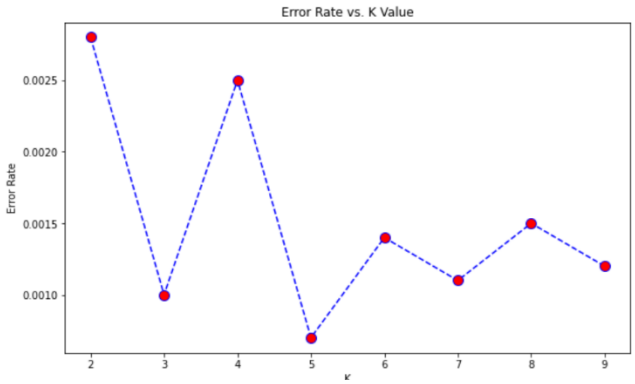


Ilustración 13. Error Rate vs Valores K vecinos.

Una vez seleccionado el valor de k = 3, se continuó con el entrenamiento y prueba del modelo, obteniendo los resultados mostrados en la Tabla 4.

Tabla 4.Clasificación sin utilizar los componentes principales.

	Entrenamiento	**Pruebas**
Exactitud	0.999	0.995
Precisión	1.0	0.9989
Sensitividad	0.9898	0.9956
F1 Score	0.9949	0.9973

Utilizando 20,000 datos

Se obtuvo el desempeño mostrado a continuación:

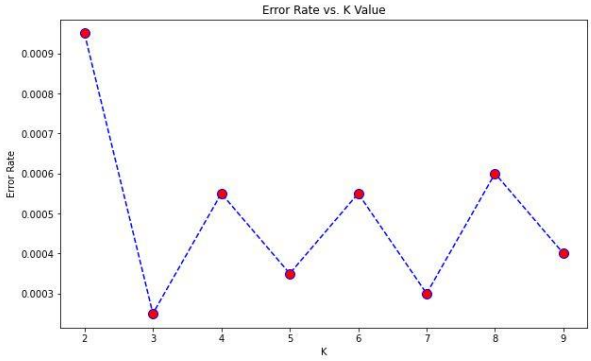


Ilustración 14.Error Rate vs Valores K vecinos.

Una vez seleccionado el valor de k = 3, se continuó con el entrenamiento y prueba del modelo, obteniendo los resultados mostrados en la Tabla 5.

Tabla 5. Clasificación sin utilizar los componentes principales.

	Entrenamiento	**Pruebas**
Exactitud	0.9997	0.9997
Precisión	1.0	1.0
Sensitividad	0.9975	0.9969
F1 Score	0.9987	0.9984

Comparando las Tablas 4 y 5, se puede observar que no existe una gran diferencia entre las métricas de valuación cuando se utilizan 10000 datos y cuando se trabaja con 20000 datos.

Clasificación con PCA

Como resultado de un barrido de valores de k para encontrar el valor, tal que, disminuya la tasa de error en la clasificación del algoritmo KNN después de un análisis de componentes principales, se obtuvo la siguiente gráfica, en donde se puede observar el incremento del error cuando aumenta el número de k vecinos.

Utilizando 10,000 datos

Se obtuvieron los siguientes resultados:

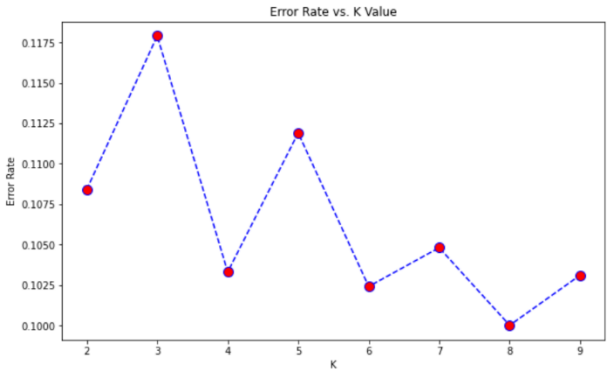


Ilustración 15.Error Rate vs Valores K vecinos.

Una vez seleccionado el valor de k = 4, se continuó con el entrenamiento y prueba del modelo, obteniendo los resultados mostrados en la Tabla 6.

Tabla 6.Clasificación sin utilizar los componentes principales.

	Entrenamiento	**Pruebas**
Exactitud	0.8967	0.9026
Precisión	0.3982	0.4081
Sensitividad	0.0933	0.1076
F1 Score	0.1511	0.1703

Utilizando 20,000 datos

Se obtuvieron los siguientes resultados:

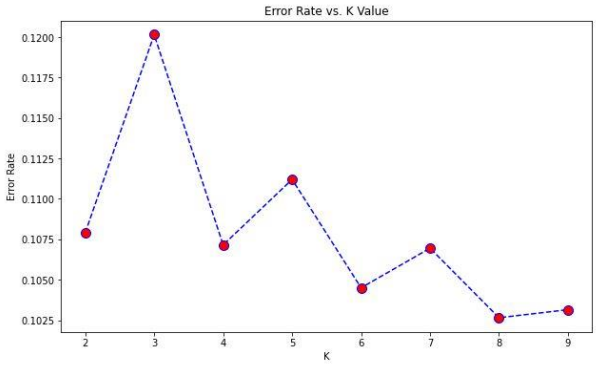


Ilustración 16.Error Rate vs Valores K vecinos.

Una vez seleccionado el valor de k = 8, se continuó con el entrenamiento y prueba del modelo, obteniendo los resultados mostrados en la Tabla 7.

Tabla 7. Clasificación utilizando los componentes principales.

	Entrenamiento	**Pruebas**
Exactitud	0.8955	0.9186
Precisión	0.4282	0.4911
Sensitividad	0.0942	0.1023
F1 Score	0.1545	0.1693

Comparando las Tablas 6 y 7, se puede observar de igual manera que existe una muy pequeña diferencia entre los valores resultantes al trabajar con 10000 datos y 20000 datos.

Así bien, durante la etapa de entrenamiento y prueba del algoritmo tanto para la clasificación con y sin PCA se obtuvieron resultados cercanos y superiores al 90%. Sin embargo, al realizar la comparación entre estas dos opciones se pudo observar que el comportamiento del modelo fue mejor cuando se utilizaron todos los atributos en cuestión. Lo anterior puede deberse tanto a la

cantidad de datos seleccionados como a los conjuntos de entrenamiento y prueba propuestos debido a que como se comentó anteriormente únicamente se utilizó menos del 7 % de la base de datos original. Otro factor importante es que 3 de los 10 atributos seleccionados por el método de PCA cuentan con un gran porcentaje de outliers, lo que posiblemente afectó los resultados.

Aun así, sería interesante probar en un futuro el comportamiento de la base de datos al utilizar toda la base de datos, con lo cual posiblemente se pueda comprobar la efectividad del algoritmo de PCA.

Así mismo, en cuestión al tiempo de entrenamiento y prueba este fue menor al utilizar el algoritmo de PCA, lo cual tiene mucha lógica ya que por medio de este algoritmo se realizó un filtrado de los atributos con mayor peso.

Conclusiones

Como se ha podido comprobar en la realización de este trabajo, el análisis de componentes principales es una técnica no supervisada de extracción de características que hace posible la reducción de las variables de un conjunto de datos de entrada con una pérdida mínima de información.

En este programa generalizado inicialmente se generaron funciones enfocadas a preparar, analizar y evaluar el conjunto de datos en cuestión a fin de poder conocer la efectividad del algoritmo de análisis de componentes principales (PCA), en donde se compararon los resultados obtenidos en la clasificación con y sin la utilización del algoritmo PCA por medio de diferentes métricas estadísticas.

Afortunadamente, al reducir la cantidad de información de la base de datos se requirió un menor tiempo y costo computacional al implementar las etapas de entrenamiento y prueba. Sin embargo, esto pudo traer ciertas deficiencias al utilizar el algoritmo de PCA con una base de datos con clases no balanceadas y con un gran número de outliers.

Referencias

[1] Lever, J., Krzywinski, M. & Altman, N. Análisis de componentes principales. Métodos nacionales 14 , 641–642 (2017). https://doi.org/10.1038/nmeth.4346

[2] Keboola. (2022, Noviembre 12). Una guía para el análisis de componentes principales (PCA) para el aprendizaje automático. Recuperado de https://www.keboola.com/blog/pca-machine-learning

[3] M. Antonio and A. Fernández, *Inteligencia artificial para programadores con prisa by Marco Antonio Aceves Fernández - Books on Google Play*. Universo de Letras. [Online]. Available: https://play.google.com/store/books/details/Inteligencia_artificial_para_programadores_con_pri?id=ieFYEAAAQBAJ&hl=en_US&gl=US

[4] "Personal Key Indicators of Heart Disease | Kaggle." https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease/code (accessed Aug. 25, 2022).