INFORME FINAL

Personal Key Indicators of Heart Disease
(Dataset)

Presentado por:
Jaime Alexis Herrera Ruiz – 1040732222 (Ing. Sistemas)
Arley Fuentes Arenales – 1038439006 (Ing. Sistemas)

Presentado a:
Raúl Ramos Pollán

Ude@
Inteligencia Artificial para las Ciencias y las Ingenierías.

Fecha:
08/noviembre/2022

CONTENIDO

INTRODUCCIÓN

1. Problema predictivo

Según el Centro para el Control y la Prevención de Enfermedades (CDC), las enfermedades cardíacas son una de las principales causas de muerte en los EE.UU. Aproximadamente la mitad de todos los estadounidenses tienen al menos uno de los tres factores de más riesgo clave de enfermedad cardíaca: presión arterial alta, colesterol alto y el tabaquismo. Además, a estos factores se incluyen también, la diabetes, la obesidad, el sedentarismo o la ingesta de alcohol.

A partir de los datos obtenidos, se construye un modelo, a través de métodos de aprendizaje automático que permiten detectar patrones que pueden predecir la posibilidad de una persona de padecer enfermedades cardíacas.

2. Métricas de desempeño

El modelo es de clasificación binaria y se utilizarían las siguientes métricas:

- Accuracy
- Precisión
- Recall o sensibilidad o TPR (Tasa positiva real)
- F1-Score
- Cohen’s Kappa
- Área bajo la curva de funcionamiento del receptor (ROC) (AUC)
- Matriz de confusión

3. Primer criterio del desempeño deseable en producción

El porcentaje de acierto esperado de la predicción sobre el padecimiento de enfermedades cardíacas es de al menos de un 85% ya que el modelo se utilizará para dar sugerencias sobre estilos de vida saludable para disminuir riesgo de estas enfermedades.

EXPLORACIÓN DEL DATASET

Obtenido desde:
https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease

El conjunto de datos proviene del CDC y hacen parte del Sistema de Vigilancia de Factores de Riesgo del Comportamiento (BRFSS), que realiza encuestas telefónicas anuales para recopilar datos sobre el estado de salud de los residentes de EE. UU.
Consta de 319795 filas y 18 columnas. La gran mayoría de las columnas son preguntas que se hacen a los encuestados sobre su estado de salud:

- **HeartDisease:** encuestados que alguna vez informaron haber tenido una enfermedad cardíaca coronaria (CHD) o un infarto de miocardio (IM).
- **BMI:** Índice de Masa Corporal (IMC).
- **Smoking:** ¿Ha fumado al menos 100 cigarrillos en toda su vida? (La respuesta Sí o No).
- **AlcoholDrinking:** bebedores empedernidos (hombres adultos que toman más de 14 tragos por semana y mujeres adultas que toman más de 7 tragos por semana)
- **Stroke:** (Alguna vez le dijeron) (usted tuvo) un accidente cerebrovascular?
- **PhysicalHealth:** Pensando en su salud física ¿durante cuántos días durante los últimos 30 días su salud física no fue buena? (0-30 días).
- **MentalHealth:** Pensando en su salud mental, ¿durante cuántos días durante los últimos 30 días su salud mental no fue buena? (0-30 días).
- **DiffWalking:** ¿Tiene serias dificultades para caminar o subir escaleras?
- **Sex:** ¿Eres hombre o mujer?
- **AgeCategory:** Categoría de edad de catorce niveles.
- **Race:** Valor de raza/etnicidad
- **Diabetic:** (Alguna vez le dijeron) (usted tenía) diabetes?
- **PhysicalActivity:** Adultos que informaron haber realizado actividad física o ejercicio durante los últimos 30 días además de su trabajo habitual.
- **GenHealth:** ¿Diría usted que, en general, su salud es...
- **SleepTime:** en promedio, ¿cuántas horas duermes en un período de 24 horas?
- **Asthma:** (Alguna vez le dijeron) (usted tenía) asma?
- **KidneyDisease:** sin incluir cálculos renales, infección de la vejiga o incontinencia, ¿alguna vez le dijeron que tenía una enfermedad renal?
- **SkinCancer:** (Alguna vez le dijeron) (usted tenía) cáncer de piel?

Notebooks para modificar el dataset

Este primer notebook se utiliza para eliminar aleatoriamente un porcentaje de datos (entre 5% y 10%) en 4 columnas escogidas también de manera aleatoria por código. Entonces, los datos eliminados fueron:

- BMI 9%
- Diabetic 6%
- SleepTime 7%
- Asthma 9%

```python
# Eliminación de datos

# Elección aleatoria de 4 columnas
def elegir_columnas():
    lista_cols = []
    df_cols = list(df.columns)
    df_cols.remove('HeartDisease')
    n = 0
    while n < 4:
        col_alea = random.choice(df_cols)
        lista_cols.append(col_alea)
        df_cols.remove(col_alea)
        n += 1
    return lista_cols

# Eliminación aleatoria de valores
num_datos = int(df.shape[0])-1
for col in elegir_columnas():
    n = 0
    # Porcentaje aleatorio
    porc_datos = int(num_datos*random.randint(5,10)/100)

    while n < porc_datos:
        # Elección aleatoria de la fila
        index = random.randint(0, num_datos)
        # Cambio del valor a vacío
        if df.loc[index, col] == df.loc[index, col]:
            df.loc[index, col] = np.NaN
            n += 1
```

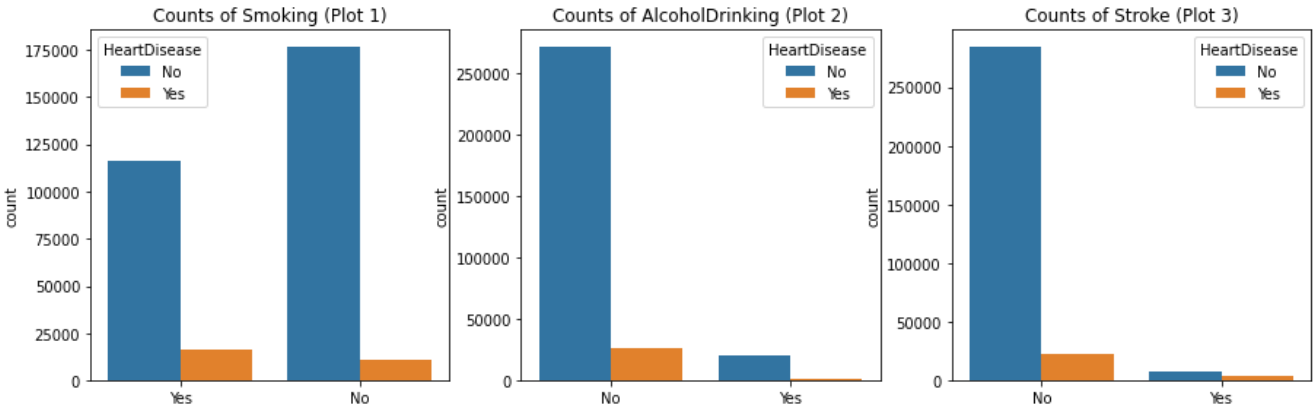
ITERACIONES

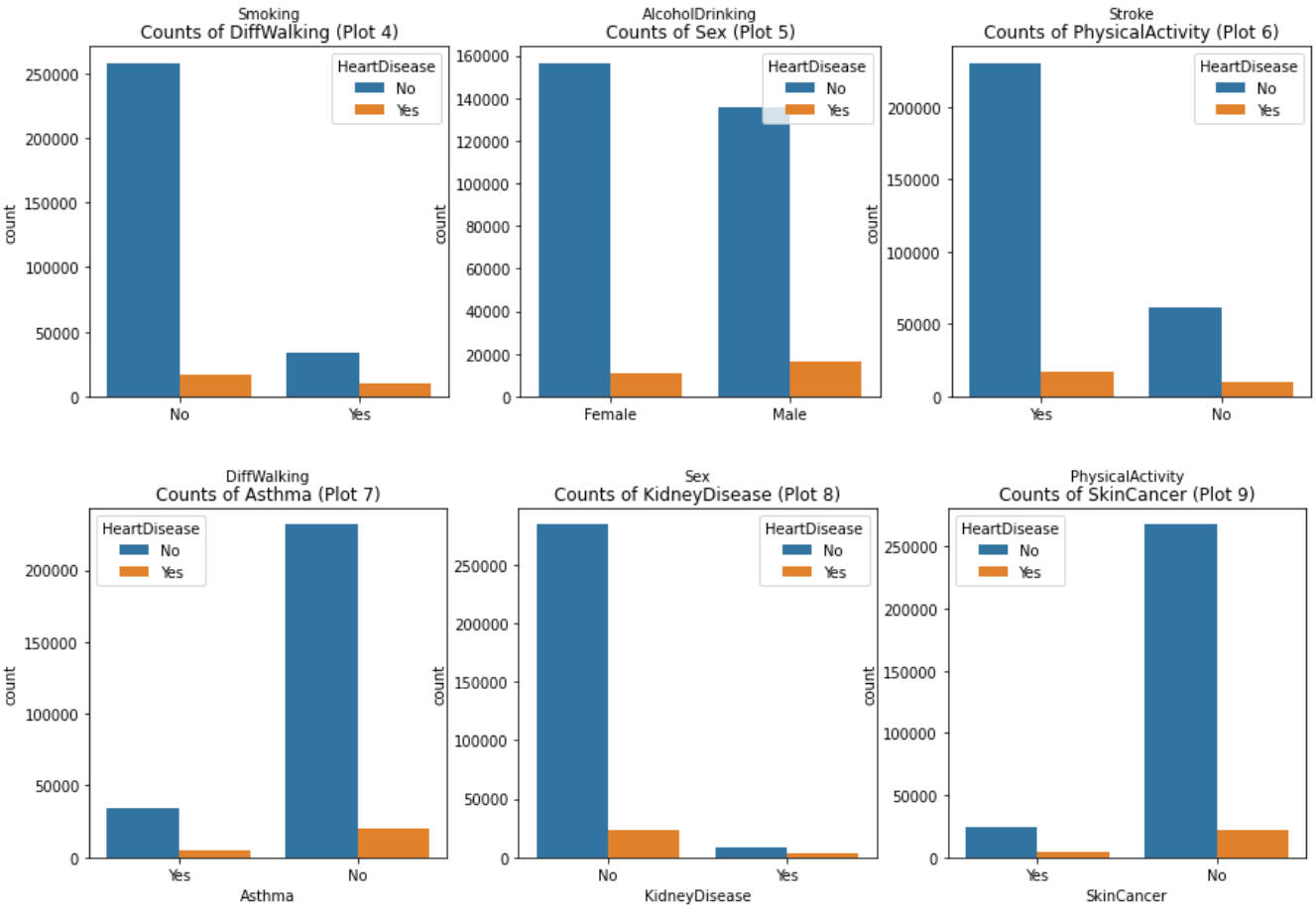
Además del notebook para modificar el dataset, se hicieron otros cuatro para la exploración, análisis y creación de modelos:

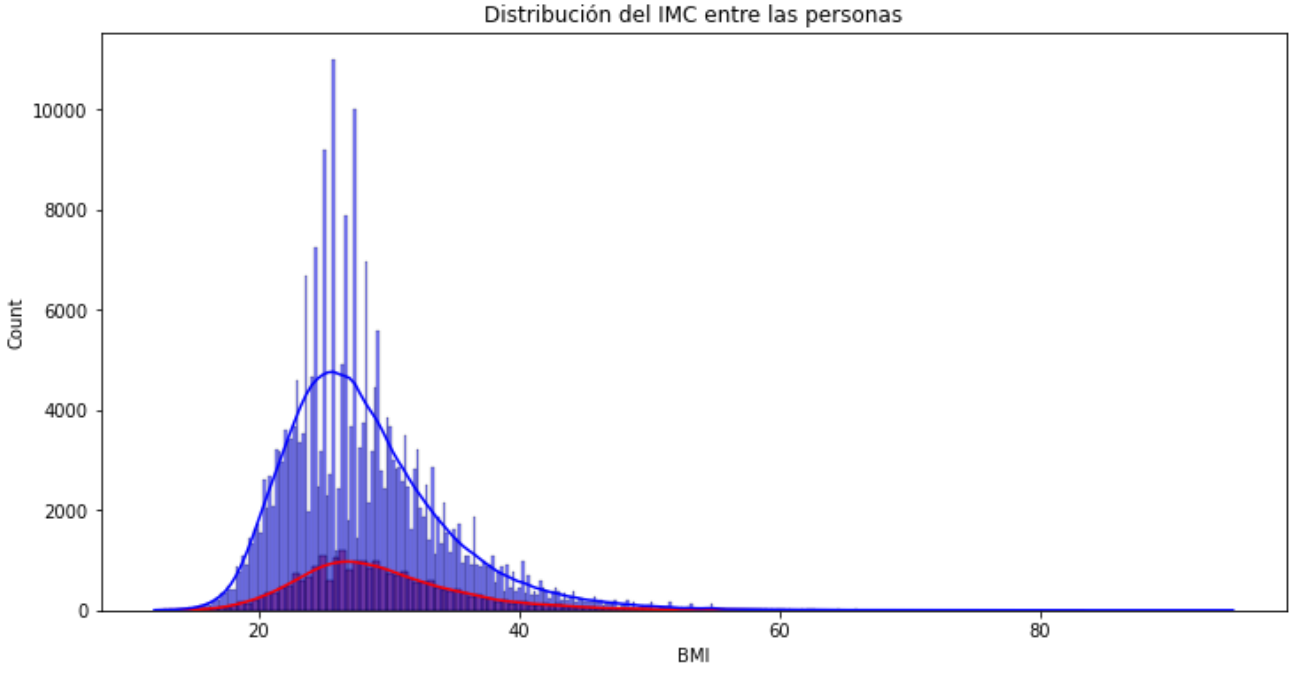
1. **Exploración de los datos**
   En este notebook mediante el uso de las librerías para graficar, se visualizan los datos para tratar de entender más fácil cómo se está comportando la información, en este caso todas las columnas se comparan con la variable respuesta que es la de enfermedades cardíacas (HeartDisease).

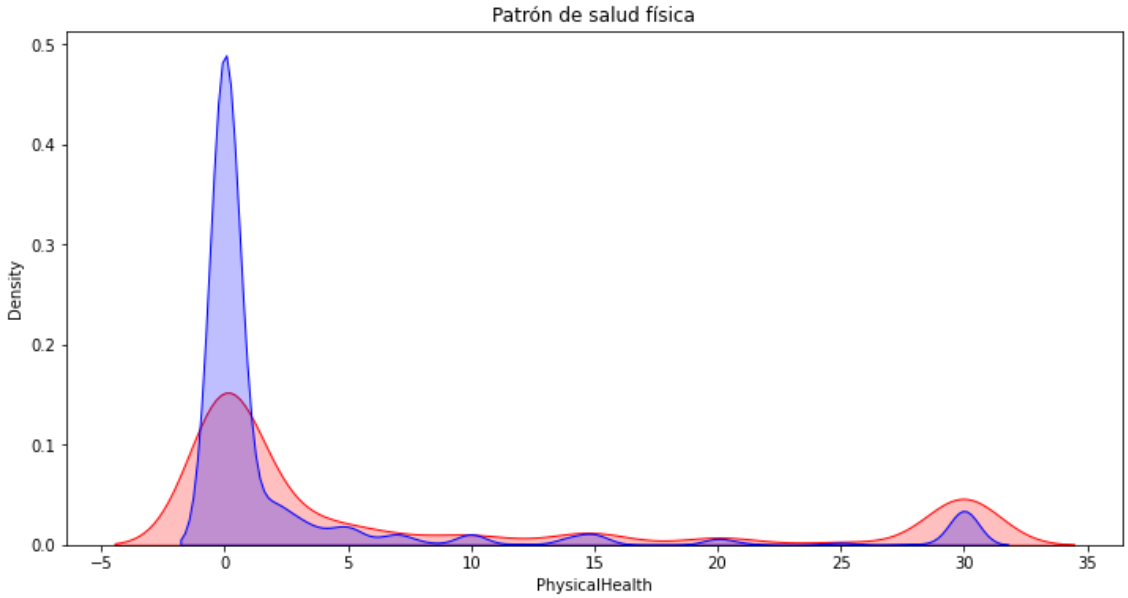
```
# Esta lista de listas contiene todas las columnas que tienen valores categóricos
binarios
colRange =
[['Smoking','AlcoholDrinking','Stroke'],['DiffWalking','Sex','PhysicalActivity'],['A
sthma','KidneyDisease','SkinCancer']]
# Esta función imprime los gráficos de conteo contando el número de personas en cada
categoría
def printCount(cols):
    fig, axes = plt.subplots(3, 3, figsize=(15, 15))
    row=0
    col=0
    p_count=1
    for row in range(3):
        for col in range(3):
            # reads column name from the list
            column = colRange[row][col]
            # plots the counts of the particular column
            sns.countplot(ax=axes[row,col],x=df[column],hue=df['HeartDisease'])
            # sets the title of the corresponding plot along with plot number
            axes[row,col].set_title("Counts of {} (Plot {})".format(column,p_count))
            p_count += 1
# Calling the function
printCount(colRange)
```

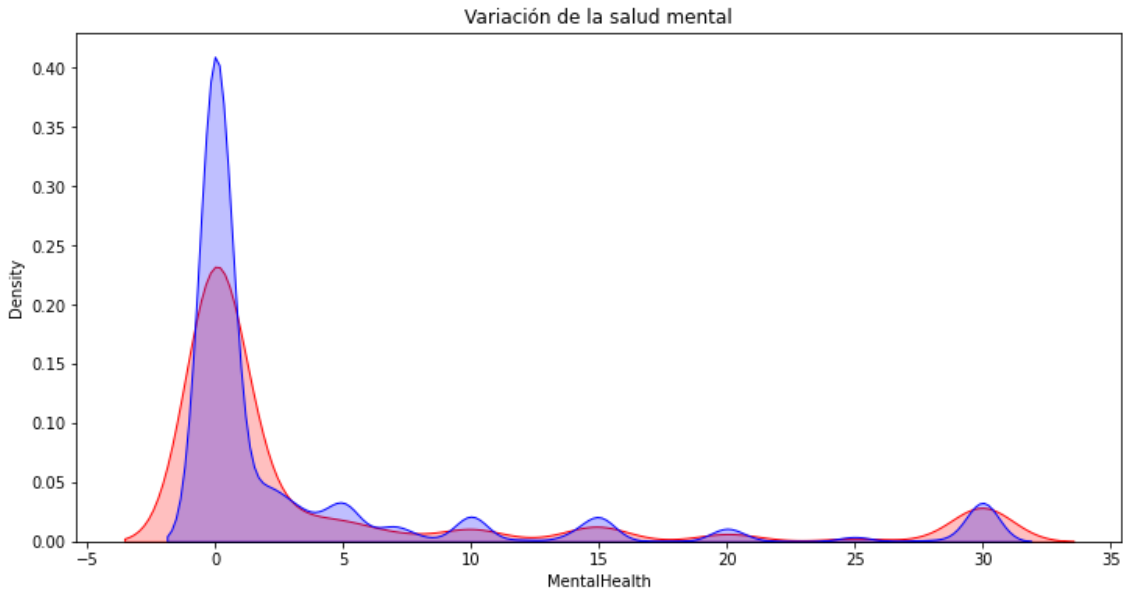
Resultado:

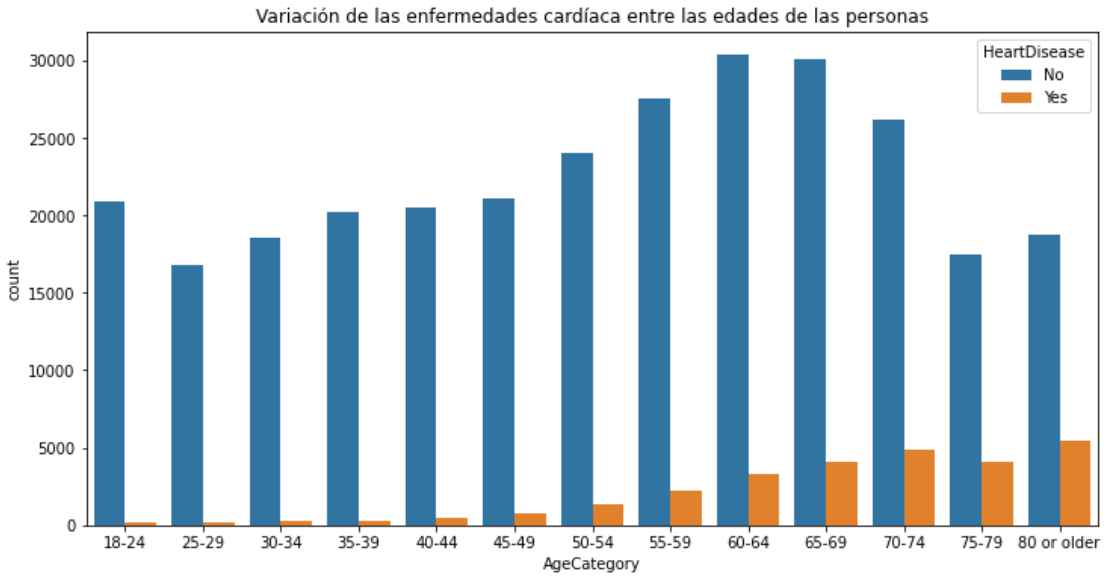


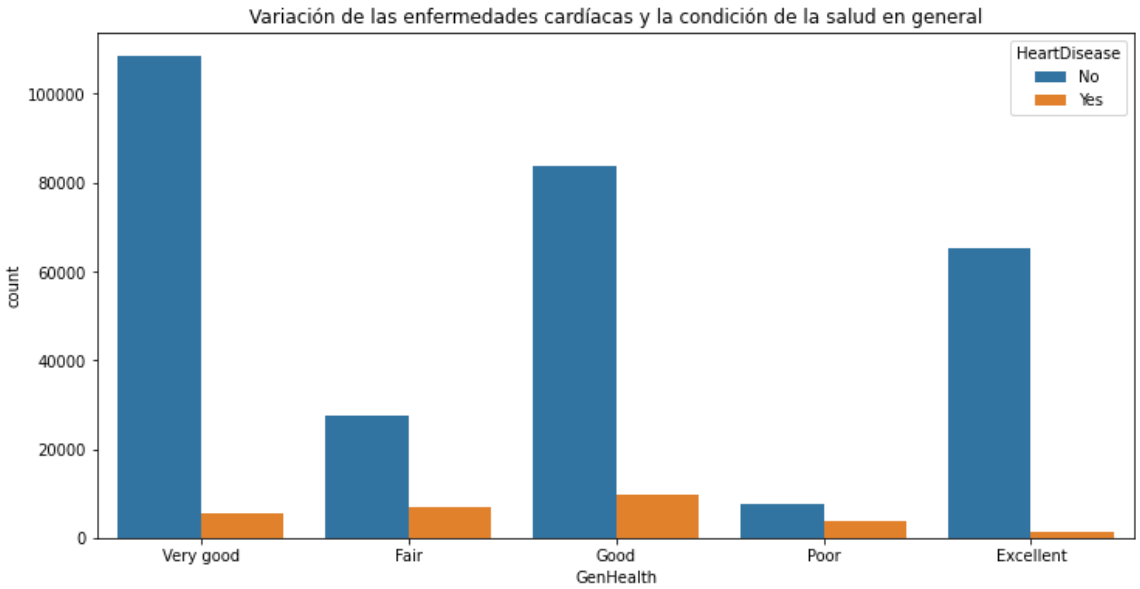

Smoking
Counts of DiffWalking (Plot 4)
AlcoholDrinking
Counts of Sex (Plot 5)
Stroke
Counts of PhysicalActivity (Plot 6)
DiffWalking
Counts of Asthma (Plot 7)
Sex
Counts of KidneyDisease (Plot 8)
PhysicalActivity
Counts of SkinCancer (Plot 9)
HeartDisease
No
Yes
count
Asthma
KidneyDisease
SkinCancer

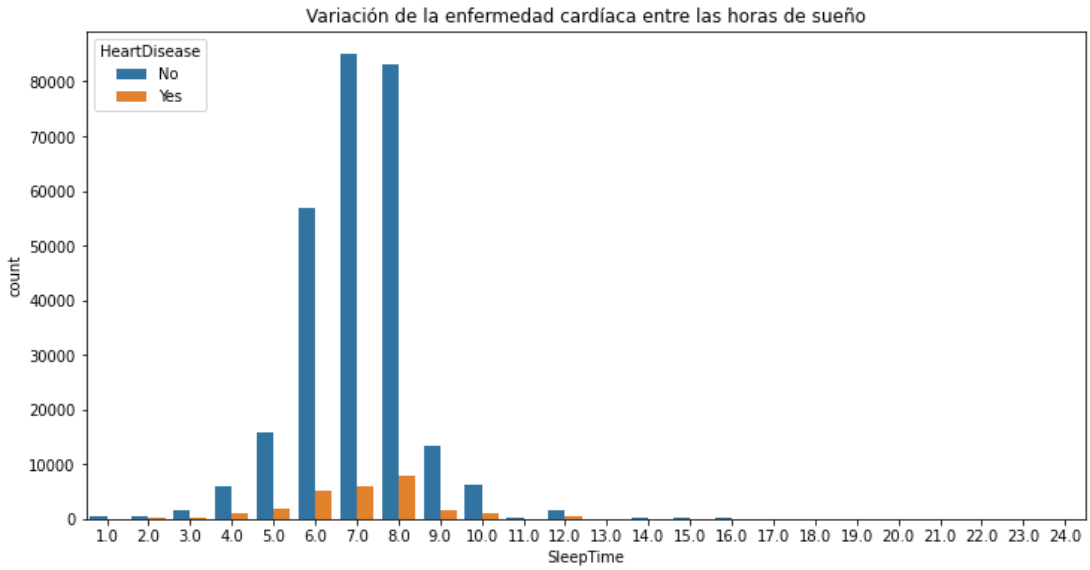

Distribución del IMC entre las personas
Count
BMI

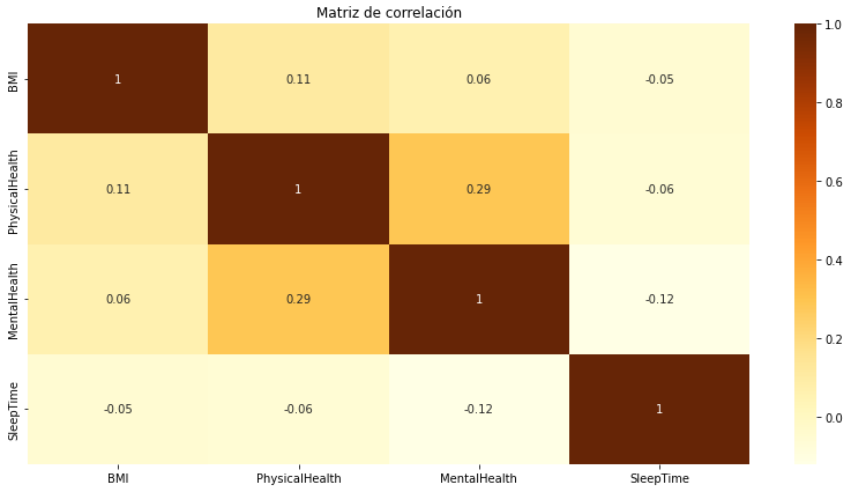

Patrón de salud física
Density
PhysicalHealth


Variación de la salud mental
Density
MentalHealth


Variación de las enfermedades cardíaca entre las edades de las personas
HeartDisease
No
Yes
count
AgeCategory
18-24
25-29
30-34
35-39
40-44
45-49
50-54
55-59
60-64
65-69
70-74
75-79
80 or older







La matriz de correlación hasta este punto no es muy útil ya que no se han tratados los datos que son de tipo categórico.

2. Limpieza de datos

En este notebook se desarrolla la limpieza de datos que consta de rellenar los datos faltantes utilizando algunos métodos como sacar la media o rellenar con datos aleatorios.

- Media:

```
df['BMI'] = df['BMI'].replace(np.nan, round(df['BMI'].mean(), 2))
```

- Rellenar datos aleatoriamente:

```
i = 0
for cn in df['Diabetic']:
    if df.loc[i, 'Diabetic'] != df.loc[i, 'Diabetic']:
        df.loc[i, 'Diabetic'] = random.choice(['Yes', 'No'])
    i += 1
```

- Fusionando respuestas:

```
df['Diabetic'] = df['Diabetic'].replace({'No, borderline diabetes':'No','Yes (during pregnancy)':'Yes' })
```

También se transforman los valores de texto a valores numéricos ya que los algoritmos de Machine Learning trabajan mejor de esta forma.

```
df =  df[df.columns].replace({'Yes':1, 'No':0, 'Male':1,'Female':0})
df['Diabetic'] = df['Diabetic'].astype(int)
```

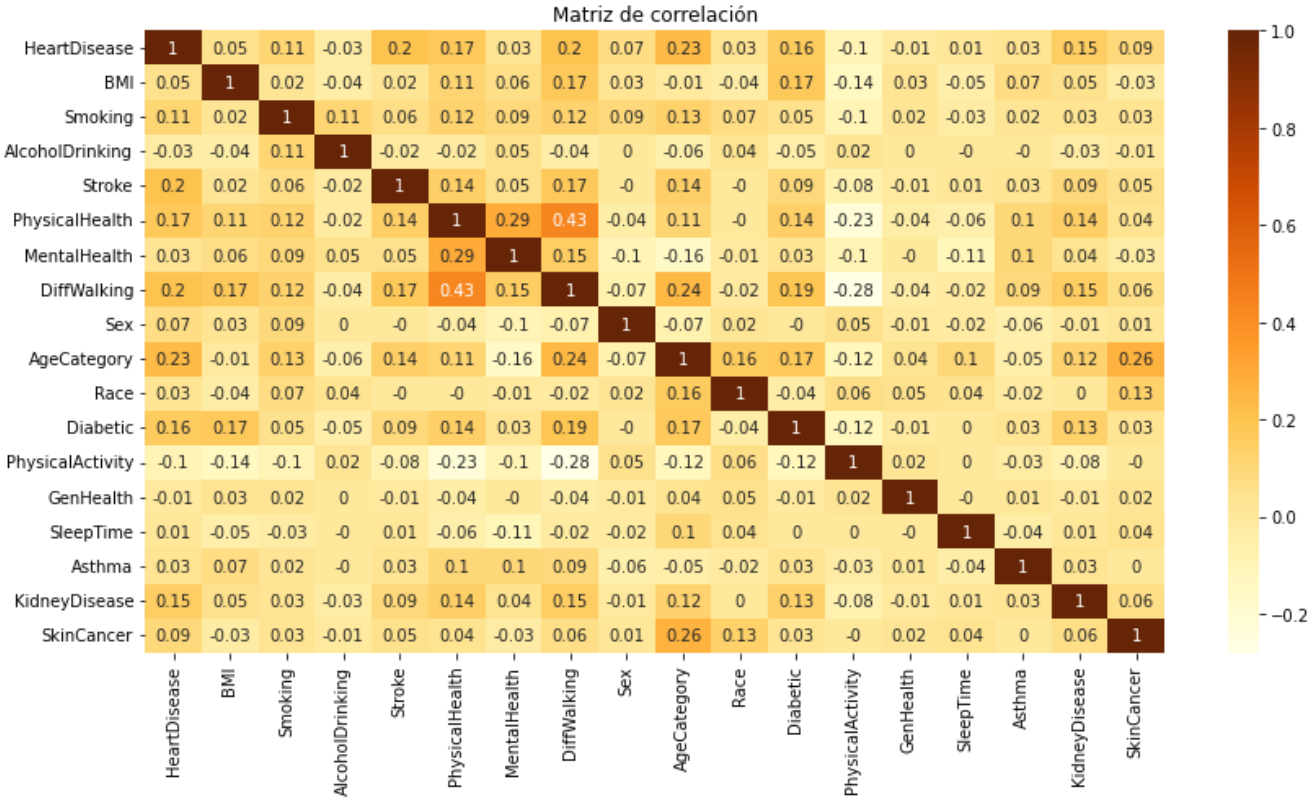
```
le = preprocessing.LabelEncoder()

cols = ['AgeCategory', 'Race', 'GenHealth']
age_name = []
race_name = []
gh_name = []

for col in cols:
    df[col] = le.fit_transform(df[col])
    if df[col].name == 'AgeCategory':
        age_name = le.classes_
        print(le.classes_)
    if df[col].name == 'Race':
        race_name = le.classes_
        print(le.classes_)
    if df[col].name == 'GenHealth':
        gh_name = le.classes_
        print(le.classes_)
```

Se crea la matriz de correlación ya habiendo organizado los datos:

```
correlation = df.corr().round(2)
plt.figure(figsize = (14,7))
sns.heatmap(correlation, annot = True, cmap = 'YlOrBr')
plt.title('Matriz de correlación')
```



3. Creación de modelos

Ya habiendo hecho la limpieza de datos, procedimos a realizar algunos modelos de Machine Learning:

a. Muestra y estandarización

Muestra:

```
# Se separa la columna con la información de los sobrevivientes
y = df['HeartDisease']
X = df.drop('HeartDisease', axis=1)
# Se separan los datos de "train" en entrenamiento y prueba para probar los
algoritmos
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Estandarización:

```
scaler = StandardScaler()

# Scale trainint data
X_train = scaler.fit_transform(X_train)

# Scale test data
X_test = scaler.fit_transform(X_test)
```

b. Utilización de métricas

Se utilizan las métricas: Accuracy, Precision, Recall, F1 Score, Cohen's Kappa, Área bajo la curva y la matriz de confusión.

```
def evaluate_model(model, x_test, y_test):
    # Predict Test Data
    y_pred = model.predict(x_test)

    # Calculate accuracy, precision, recall, f1-score, and kappa score
    acc = metrics.accuracy_score(y_test, y_pred)
    prec = metrics.precision_score(y_test, y_pred)
    rec = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    kappa = metrics.cohen_kappa_score(y_test, y_pred)

    # Calculate area under curve (AUC)
    y_pred_proba = model.predict_proba(x_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)

    # Display confussion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)

    return {'acc': acc, 'prec': prec, 'rec': rec, 'f1': f1, 'kappa': kappa,
            'fpr': fpr, 'tpr': tpr, 'auc': auc, 'cm': cm}
```

```
def imprimir_resultados(modelo_eval):
    print('Accuracy:', modelo_eval['acc'])
    print('Precision:', modelo_eval['prec'])
    print('Recall:', modelo_eval['rec'])
    print('F1 Score:', modelo_eval['f1'])
    print('Cohens Kappa Score:', modelo_eval['kappa'])
    print('Area Under Curve:', modelo_eval['auc'])
    print('Confusion Matrix:\n', modelo_eval['cm'])
```

c. Modelos Supervisados.

- Regresión logística:

```
logreg = LogisticRegression(random_state=0, max_iter=1000)
logreg.fit(X_train, y_train)

# Evaluar modelo
logreg_eval = evaluate_model(logreg, X_test, y_test)

# Resultados
imprimir_resultados(logreg_eval)
```

Impresión:

- Accuracy: 0.912881689832549
- Precision: 0.5108225108225108
- Recall: 0.0844062947067239
- F1 Score: 0.1448741559238797
- Cohens Kappa Score: 0.12313107569833648
- Area Under Curve: 0.8255755666508892
- Confusion Matrix:
  [[57915 452]
  [5120 472]]

- Vecinos más cercanos

- Accuracy: 0.9040010006410356
- Precision: 0.36244979919678716
- Recall: 0.12911301859799715
- F1 Score: 0.19040084388185657
- Cohens Kappa Score: 0.151426767548428
- Area Under Curve: 0.701154579504121
- Confusion Matrix:
  [[57097 1270]
  [4870 722]]

- Árbol de decisión

- Accuracy: 0.8606451007676793
- Precision: 0.2264865755229781
- Recall: 0.24588698140200285
- F1 Score: 0.23578839063705734
- Cohens Kappa Score: 0.15926298910113368
- Area Under Curve: 0.5846675678877964
- Confusion Matrix:
  [[53671 4696]
  [4217 1375]]

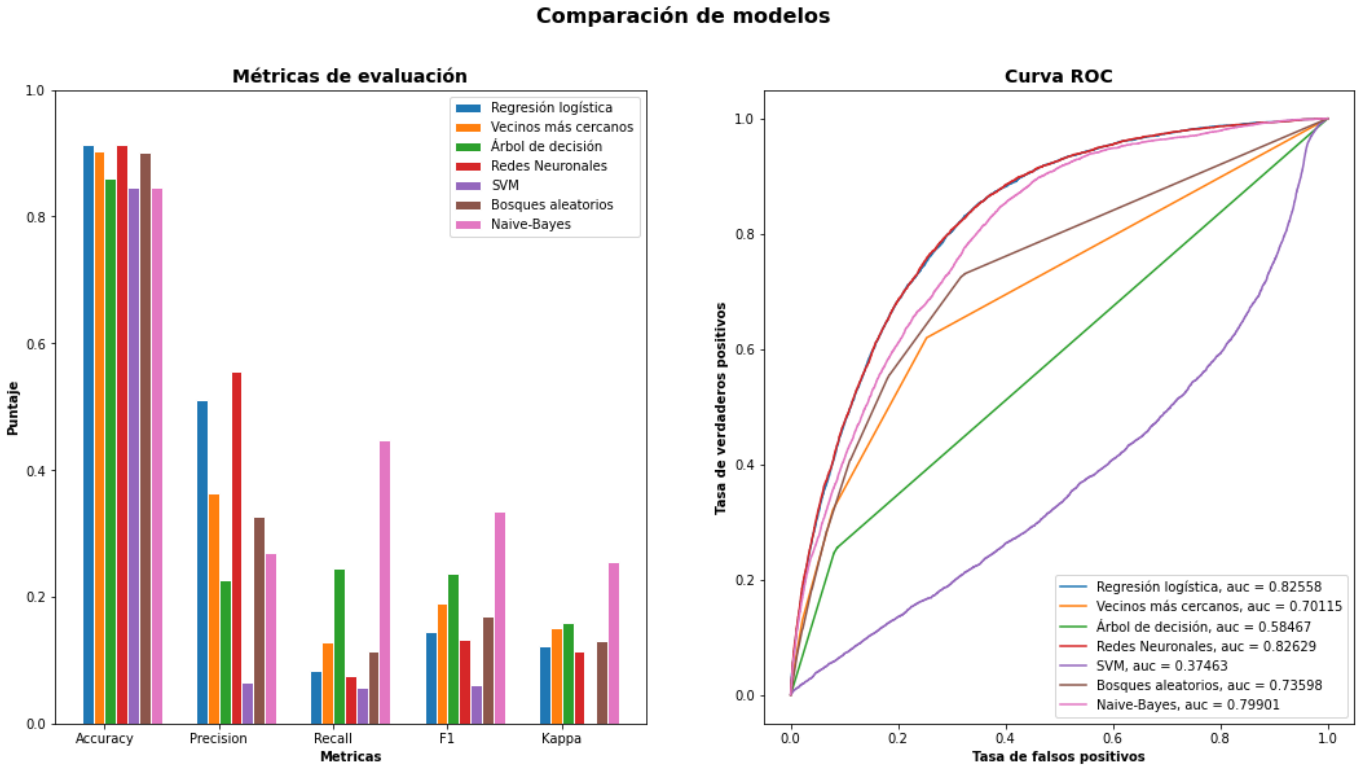
- Redes neuronales
  - Accuracy: 0.9138823308682125
  - Precision: 0.5549738219895288
  - Recall: 0.07582260371959942
  - F1 Score: 0.13341724354940213
  - Cohens Kappa Score: 0.11481177586662983
  - Area Under Curve: 0.8262913858937035
  - Confusion Matrix:
    [[58027 340]
    [5168 424]]

- Máquina de vectores de soporte
  - Accuracy: 0.845557310151816
  - Precision: 0.06513798701298701
  - Recall: 0.05740343347639485
  - F1 Score: 0.061026615969581754
  - Cohens Kappa Score: -0.022749396403559663
  - Area Under Curve: 0.3746328697651947
  - Confusion Matrix:
    [[53760 4607]
    [5271 321]]

- Bosques aleatorios
  - Accuracy: 0.9019684485373443
  - Precision: 0.327217125382263
  - Recall: 0.1148068669527897
  - F1 Score: 0.16997617156473394
  - Cohens Kappa Score: 0.13048551662722574
  - Area Under Curve: 0.735979623029583
  - Confusion Matrix:
    [[57047 1320]
    [4950 642]]

- Naive-Bayes
  - Accuracy: 0.8452133397958067
  - Precision: 0.26868556701030927
  - Recall: 0.44742489270386265
  - F1 Score: 0.3357487922705314
  - Cohens Kappa Score: 0.25427576023808596
  - Area Under Curve: 0.7990070041243885
  - Confusion Matrix:
    [[51557 6810]
    [3090 2502]]

- Comparación
  Se comparan gráficamente los modelos supervisados utilizados:



Los modelos de regresión logística, vecinos más cercanos, redes neuronales y bosques aleatorios son los que más muestran una gran cantidad de predicciones positivas que fueron correctas. Además la regresión logística y las redes neuronales según la curva ROC son los que tienen mejor rendimiento frente a los otros modelos.
El modelo Naive-bayes es el que mejor garantiza que haya predicciones de verdaderos positivos y verdaderos negativos. Esto no quiere decir que sea un buen modelo, ya que los valores de las métricas siguen siendo muy bajos, es decir, las predicciones utilizando estos modelos no son muy confiables.

d. Modelos no supervisados

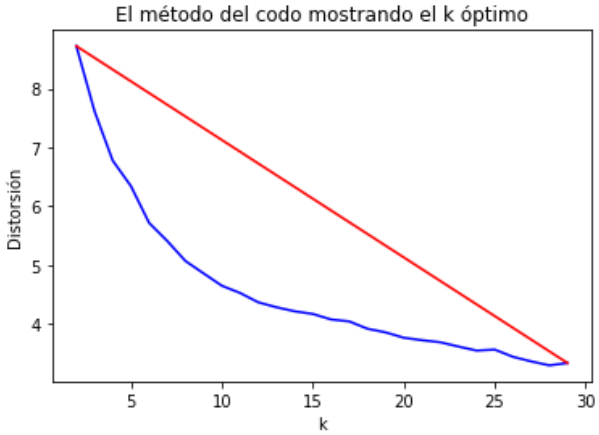
- PCA y Clustering

```
pca = PCA(n_components=0.95, random_state=42)
X_reduced= pca.fit_transform(X)
X_reduced.shape
```

Se evalúa el método kmeans con diferente cantidad de clúster para luego elegir el optimo.

```
# run kmeans with many different k
distortions = []
K = range(2, 30)
for k in K:
    k_means = KMeans(n_clusters=k, random_state=42).fit(X_reduced)
    k_means.fit(X_reduced)
    distortions.append(sum(np.min(cdist(X_reduced, k_means.cluster_centers_,
'euclidean'), axis=1)) / X.shape[0])
```

Utilizamos el método del codo a partir de la evaluación anterior, para definir el mejor número de clúster, en este caso 9.

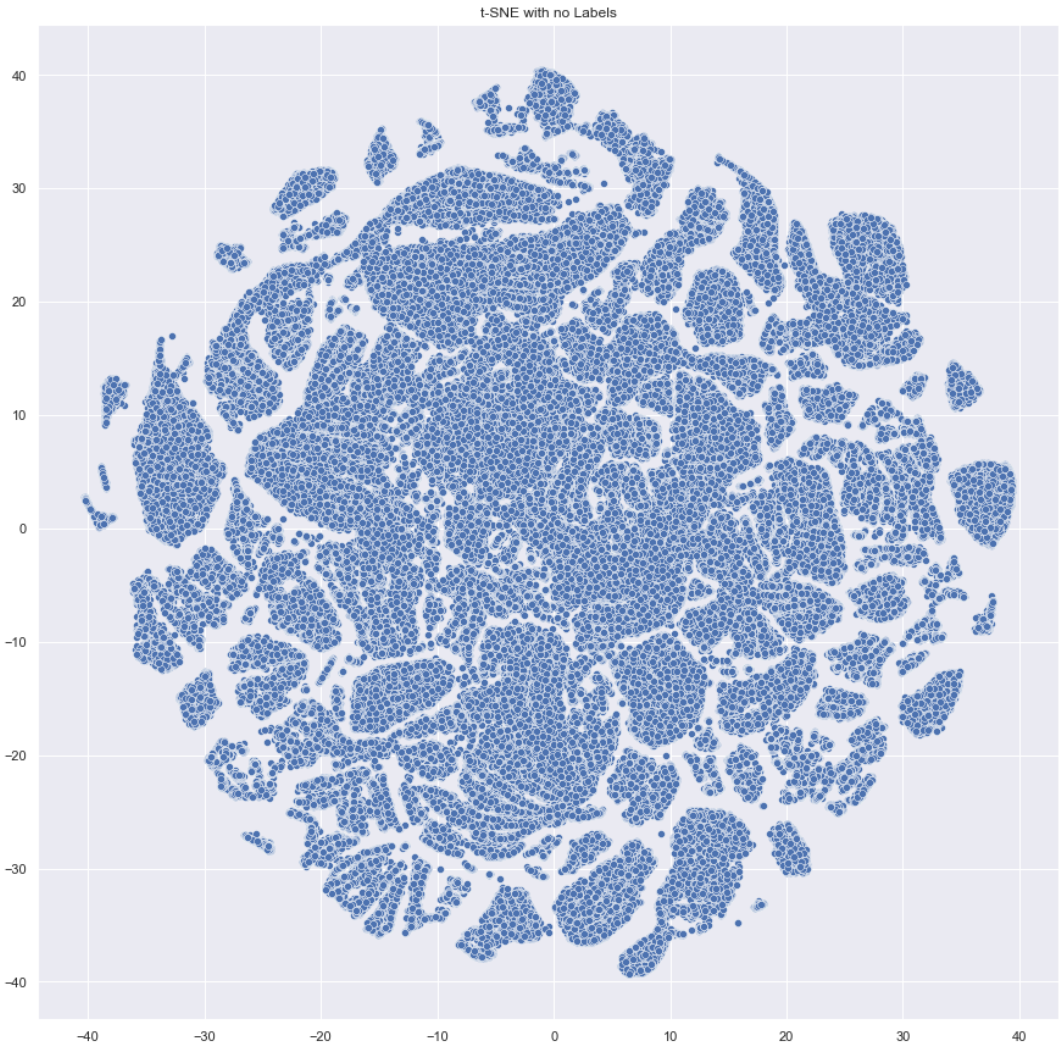


Se reduce la dimensionalidad utilizando t-SNE

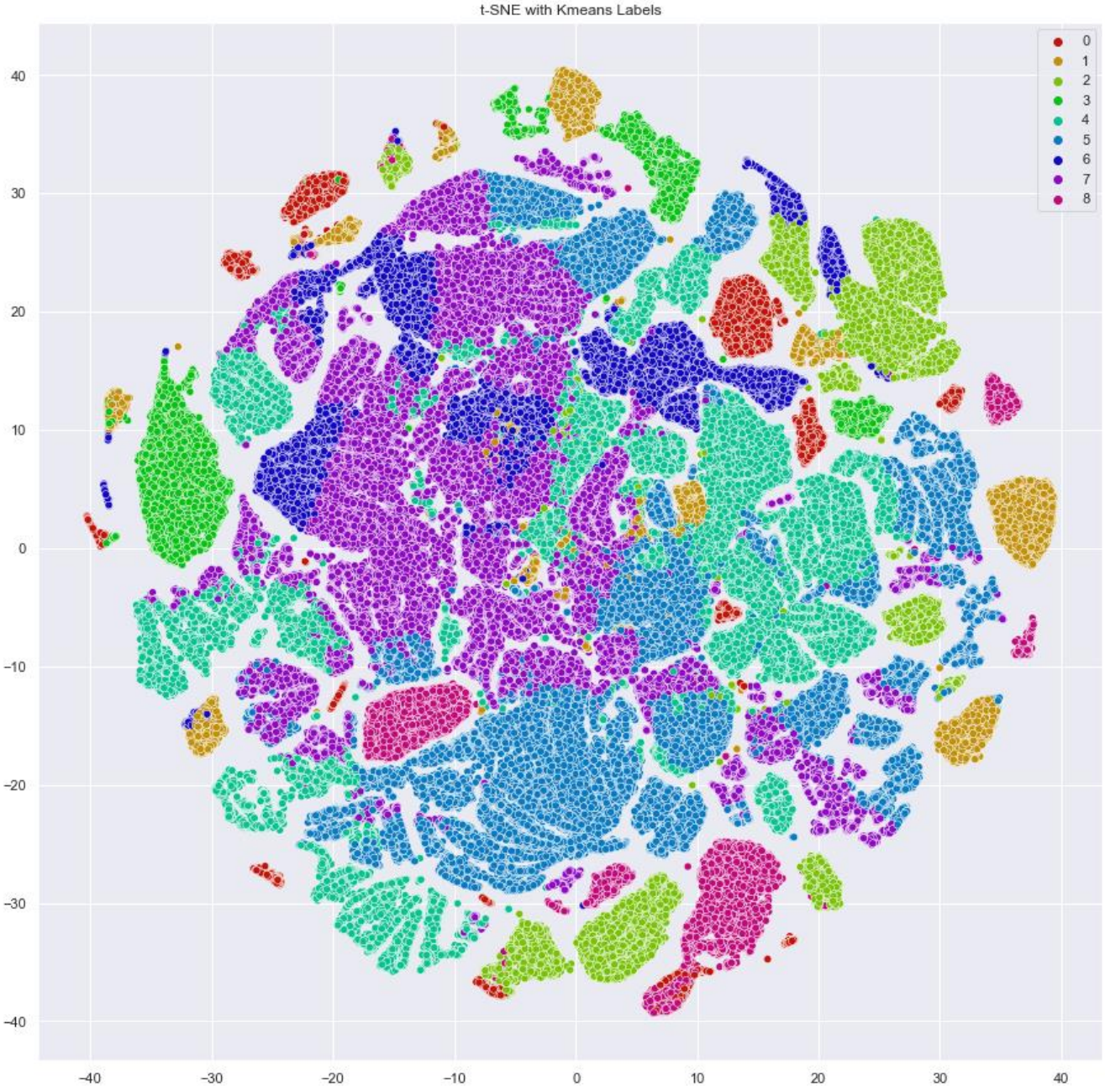
```
tsne = TSNE(verbose=1, perplexity=50)
X_embedded = tsne.fit_transform(X)
```

Esto para representar las similitudes entre vecinos en un espacio de menor dimensión.

Ahora graficamos los datos en dos dimensiones:



Para mejor visualización, coloreamos los nueve clústeres definidos anteriormente:



De esta forma agrupamos los datos en k (9) grupos de acuerdo con sus características. Este agrupamiento minimiza la suma de distancias entre cada objeto y el centroide de su grupo o clúster.

RETOS Y CONSIDERACIONES

- La exploración y limpieza de datos es un paso fundamental ya que una mala organización de la información conlleva posiblemente a obviar información importante o, por el contrario, priorizar información redundante.
- Algunos modelos como las redes neuronales son demorados en procesar, debido a la cantidad de iteraciones (500 en este caso) y por el gran número de datos. Esto depende también de la capacidad del equipo donde se ejecuta.
- La alta Accuracy en los modelos se debe a la gran cantidad de personas sanas que hay respecto a las que no. Esto supone en este caso, que es una métrica para no tomar en cuenta y es mejor basarse en la precisión y en el Recall o utilizar la métrica F1-Score

CONCLUSIÓN

Se nos hizo confusa la interpretación de los modelos en cuanto a las métricas, por eso suponemos que estos resultados salen a partir de un conjunto de datos muy desequilibrado (Alta cantidad de personas con posibilidades de NO sufrir enfermedades cardíacas contra las que sí). El modelo de regresión logística es el que utilizaríamos para predecir, aunque según la información obtenida se considera mejor utilizarlo para predecir la ausencia de enfermedades cardíacas (Verdaderos negativos).
Sobre los modelos no supervisados, no comprendemos cómo utilizarlos para tomar decisiones o hacer predicciones.

FUENTES

HeartDiseases. (2022, enero 28). https://www.cdc.gov/nchs/fastats/heart-disease.htm

Personal Key Indicators of Heart Disease. (s. f.). Recuperado 4 de julio de 2022, de https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease

Preprocesamiento de datasets con Python, Scikit-learn y Pandas (Parte #4)—Aprende con ejemplos. (2021, mayo 12). https://aprendeconejemplos.org/python/preprocesamiento-de-datasets-con-scikit-learn-y-pandas

¿Clasificación o Regresión? - IArtificial.net. (2018, diciembre 15). https://www.iartificial.net/clasificacion-o-regresion/

Prediction of Heart Disease (Easy). (s. f.). Recuperado 22 de agosto de 2022, de https://kaggle.com/code/arkalodh/prediction-of-heart-disease-easy

Heart Disease Prediction. (s. f.). Recuperado 22 de agosto de 2022, de https://kaggle.com/code/andls555/heart-disease-prediction