PREDICCIÓN DE NIVEL DE ESTRÉS

Por :
Ana María Montañez Becerra
Joanny torres Cardona

Materia:
Introducción a la Inteligencia Artificial

Profesor:
Raúl Ramos Pollan

UNIVERSIDAD DE ANTIOQUIA
1 8 0 3

UNIVERSIDAD DE ANTIOQUIA
FACULTAD DE INGENIERÍA
MEDELLÍN 2023

Contenido

INTRODUCCIÓN

La inteligencia artificial es una herramienta muy usada y con una gran variedad de aplicaciones. En el área de la salud se puede usar para predecir la cantidad de fallecimiento por enfermedades evitables, es decir, conociendo el estado de salud y los hábitos de algunas personas es posible saber su tendencia a sufrir alguna enfermedad adquirida por malos hábitos alimenticios o falta de ejercicio.

1. PLANTEAMIENTO DEL PROBLEMA

El sedentarismo y la inactividad física han aumentado los últimos años, tanto en adultos, jóvenes y niños. Esto trae consigo el aumento de obesidad y la aparición de algunas enfermedades crónicas como diabetes y enfermedades cardiovasculares hasta la muerte prematura [1]. En la actualidad existe tecnología que permite tener idea de alguna información que nos da idea del estado de salud, estos datos pueden ser: latidos por minuto, porcentaje de saturación de oxígeno en sangre, frecuencia respiratoria y cardiaca. Y al mismo tiempo permite tener un control sobre el tiempo invertido en ejercicio físico y el número de pasos dados durante el día [2].

Dada información de un grupo de personas obtenida mediante un reloj inteligente y encuestas, se quiere evaluar la salud mental, física y estado de ánimo, teniendo en cuenta valores como frecuencia cardiaca, el porcentaje de saturación de oxígeno, niveles de estrés y calidad del sueño, se propone el diseño de un algoritmo que permita predecir el nivel de estrés y al mismo tiempo correlacionar con otras variables que afecten la calidad de vida del individuo.

1.1 Dataset

A partir de dataset *"Lifesnaps Fitbit data set"* tomado de Lifesnaps Fitbit dataset | Kaggle se selecciona los datos tomados diariamente para usarlos en este proyecto. Para la construcción de este dataset se contó con la participación de 71 sujetos, el tamaño del dataset a trabajar es de 7410 filas y 63 columnas. Algunos de los datos tomados del usuario por medio del reloj inteligente son:

- **`nightly_temperature:`** temperatura durante la noche
- **`nremhr:`** Heart rate during NREM sleep
- **`rmssd:`** Medida más relevante y precisa de la actividad del sistema nervioso autónomo a corto plazo
- **`spo2:`** porcentaje de saturación de oxígeno
- **`calories:`** Calorías
- **`bpm:`** Latidos por minutos

Las que el usuario lleno mediante una encuesta son:

- **`stress_score:`** Clasificación al nivel de estrés
- **`sleep_points_percentage:`** Porcentaje de puntuación al dormir
- **`HAPPY:`** Clasificación de nivel de sentimiento de felicidad
- **`SAD:`** Clasificación de nivel de sentimiento de tristeza
- **`TENSE/ANXIOUS:`** Clasificación de nivel de sentimiento de ansiedad

- **`TIRED:`** Clasificación de nivel de sentimiento de cansancio
- **`GYM:`** Asistencia al gimnasio

Por cada uno de los pacientes existe más de un dato por día, por lo tanto se evaluará la posibilidad de hacer promedio de los datos para cada uno de los pacientes para así comparar los datos de diferentes pacientes y poder hacer estadística descriptiva teniendo en cuenta el género y la edad.

1.2 Métrica

- Error absoluto de media: Es una media del valor absoluto de los errores. Es la métrica más fácil de comprender ya que es el promedio de los errores.

- Error Cuadrado Medio (MSE): El Error Cuadrado Medio (MSE) es la media del error cuadrático. Es más popular que el error de Media absoluto ya que hace foco en grandes errores. Esto se debe a que el término cuadrático tiene errores más grandes que van aumentando su valor.

- Error Cuadrático Medio (RMSE). R-cuadrática: Esta métrica es una medida popular para darle precisión al modelo. Representa cuán cerca están los datos de la linea de regresión ajustada. Mientras más alto el R-cuadrático, mejor se encontrará ajustado el modelo respecto de los datos. El puntaje mejor posible es 1.0 y puede tomar valores negativos .

1.3 Variable Objetivo

La variable objetivo es el nivel de estrés (**`stress_score`**), esto con el fin de relacionar el impacto de los niveles de estrés en la salud de cada uno de los individuos y predecir cómo se comportará este partiendo como base en los otros datos de cada uno de los sujetos.

2. EXPLORACIÓN DE LAS VARIABLES

Para iniciar con la exploración de variables lo primero que se hace es agrupar los datos ya que, como se mencionó anteriormente, existen varios datos de un mismo sujeto, dicho dataset queda con el nombre de newData, luego de esto se analizan algunas variables de interés.

2.1 Análisis de la variable objetivo

Para iniciar con la exploración de las variables, se analiza el comportamiento de la distribución que tiene la variable objetivo de los datos crudos, dicho comportamiento de muestra en la *Figura 1,* donde se puede observar que los datos se distribuyen aproximadamente entre 50 a 95, pero de igual manera se observa que existen varios datos faltantes.

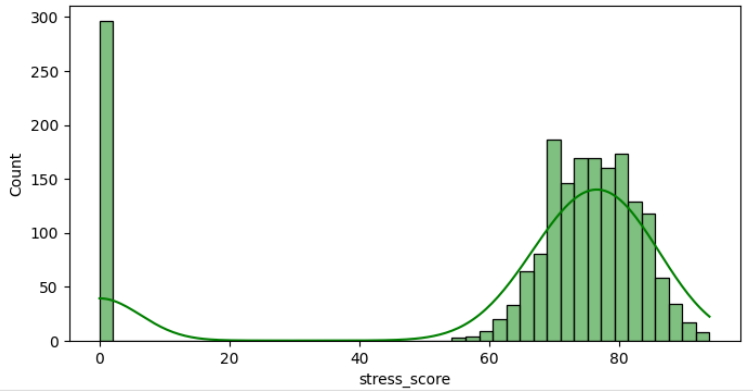

Figura 1. Variable objetivo datos crudos

En la *Figura 2* se cambian esos datos faltantes por la media de todos los datos después de haberlos agrupados según el id de cada uno de los participantes, y estos son los datos que se usarán en lo que resta del trabajo.

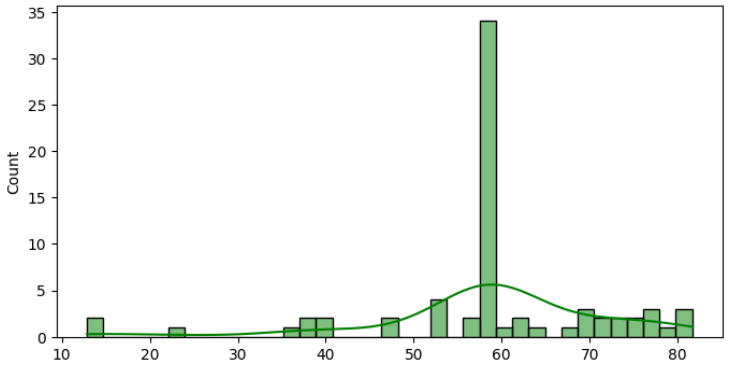

Figura 2. Variable objetivo

2.2 Clasificación de los datos

En el dataset se brinda información del porcentaje de descanso o sueño REM de cada uno de los participantes, esto permite ver la relación que existe entre el nivel de estrés y la calidad de sueño, por otra parte está la edad lo que permite ver qué grupo de personas sufre más de estrés debido a diferentes factores, estas se dividen en dos grupos, personas menores de 30 años y personas mayores a 30 años.

2.3 Datos faltantes

En el dataset crudo se observaron que para varias variables hacían falta datos, por lo que se decide reemplazar estos datos faltantes por la media de los datos, y para variables a las cuales les faltaban el 50% de los datos totales o más se decide eliminarlas debido a que no brindarán mucha información. Este es el caso de `'scl_avg'` y `'spo2'`.

2.4 Correlación de las variables

En la *Figura 3* se observa la correlación que hay entre los datos que componen el dataset, es posible observar que en la mayoría de los datos no existe una correlación

entre estos o es muy débil, partiendo de acá se decide cuales se van a usar en el entrenamiento supervisado del algoritmo.

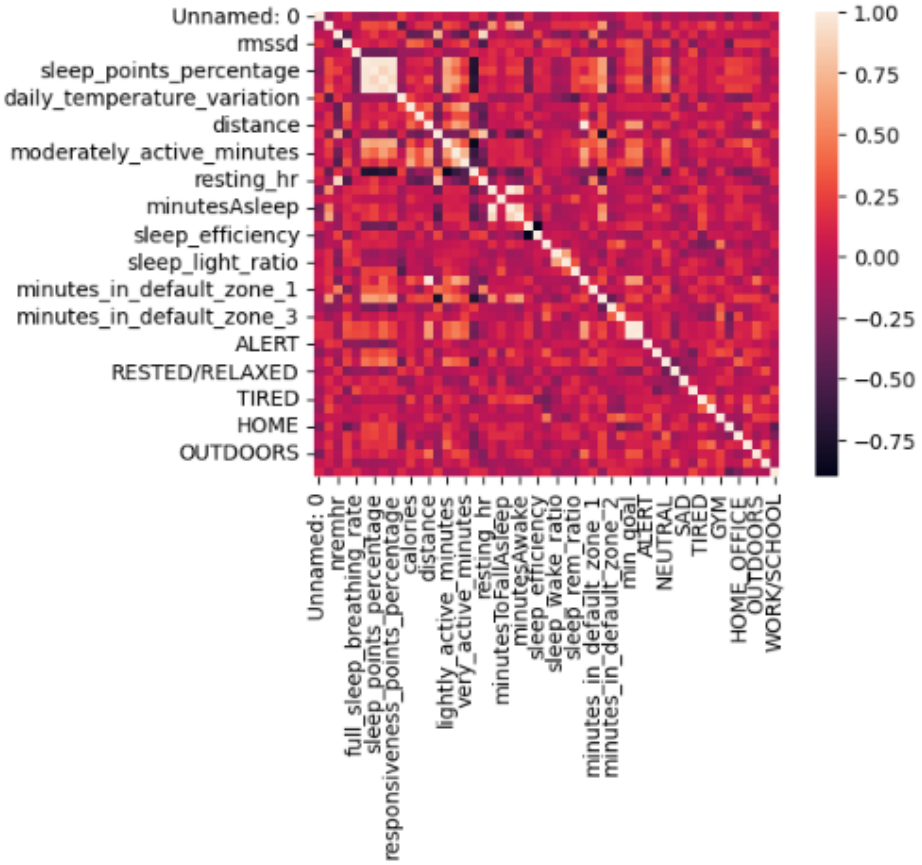

Figura 3. Correlación entre los datos

3. TRATAMIENTO DE LOS DATOS

- **Eliminación de las columnas con muchos datos faltantes:**
  Como antes se mencionó, se elimina las columnas donde más del 50% de los datos no están, ya que no serán de relevancia y no aportan información al modelo. de esta manera las variables eliminadas son `'scl_avg'` y `'spo2'`. La *Figura 4* muestra el código usado para la eliminación de dichos valores.

```
[ ]  newData1 = newData.drop(['scl_avg', 'spo2'], axis=1)
```

Figura 4. Código para eliminar datos faltantes

- **Relleno de datos faltantes:**
  Para el relleno de los datos faltantes se opta por reemplazarlos por la media de los datos, se tiene presente que se concentran todos los datos en ese punto y esto puede llegar a afectar el entrenamiento del modelo. En la *Figura 5* se muestra el código utilizado para esto.

```
[ ]  newData1.fillna(newData.mean(),inplace=True)
```

Figura 5. Código para llenar datos faltantes

4. MÉTODOS SUPERVISADOS

4.1 Selección de modelos

4.1.1 Aplicación métrica

La métrica que se usa para medir el desempeño del modelo es el error medio absoluto, MSE y R2-score en la *Figura 6* se muestra que código usado para esto.

```
test_x = np.asanyarray(validation_examples[['sleep_points_percentage']])
test_y = np.asanyarray(validation_targets[['stress_score']])
test_y_predict = regression.predict(test_x)


print("Error medio absoluto: %.2f" % np.mean(np.absolute(test_y_predict - test_y)))
print("Suma residual de los cuadrados (MSE): %.2f" % np.mean((test_y_predict - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_predict , test_y) )
```

Figura 6. métricas

4.1.2 Partición de los datos

Para entrenar los modelos se usará el dataset que se generó, llamado `train_x`. De todo el conjunto de estos datos es necesario hacer una particion para entrenar y otra para prueba, los datos que se tienen para la prueba no se usan en ningún momento durante el entrenamiento, posteriormente, con los datos que se tiene para el entrenamiento se hace una partición nuevamente para tener datos para validar el modelo y probar el algoritmo verificando así su desempeño, en la *Figura 7* se muestra el código usado para hacer tales particiones.

```
# Se separán los datos en train, test y validación
train_size = 0.8

# Se separan los datos de entrenamiento
z_train, z_rem, w_train, w_rem = train_test_split(z, w, train_size=0.8)

# Se separán los datos de validación y de test
test_size = 0.5
z_valid, z_test, w_valid, w_test = train_test_split(z_rem, w_rem, test_size=0.5)
```

Figura 7. Partición de los datos

5. MÉTODOS SUPERVISADOS Y NO SUPERVISADOS

Luego de hacer el análisis con métodos supervisados se empieza el análisis con los métodos no supervisados. En este caso los métodos supervisados que se usaron fueron regresión lineal y árbol de decisión.

5.1 Regresión lineal y árbol de decisión

La regresión lineal busca encontrar una relación lineal entre una variable dependiente (también llamada variable objetivo o variable de respuesta) y una o más variables independientes (también llamadas variables predictoras o características).

En la regresión lineal, se supone que la relación entre las variables puede ser aproximada por una línea recta en un espacio de características multidimensionales. El objetivo del modelo de regresión lineal es encontrar los coeficientes que mejor describen esta línea recta y permita predecir los valores de la variable dependiente a partir de los valores de las variables independientes [3][4]. En la *Figura 8* se muestra el código usado para este modelo.

```
#creacion del modelo
regression = linear_model.LinearRegression()
#train of data x and train of label y
train_x = np.asanyarray(training_examples[['sleep_points_percentage']])
train_y = np.asanyarray(training_targets[['stress_score']])

#entrenamiento del modelo
regression.fit(train_x, train_y)
# Coeficientes
print ('Coefficients: ', regression.coef_)
print ('Intercept: ',regression.intercept_)
```

```
Coefficients:  [[90.04484971]]
Intercept:  [5.31877789]
```

Figura 8. Regresión lineal

Por otra parte está el método supervisado del árbol de decisión es un algoritmo de aprendizaje automático utilizado para resolver problemas de clasificación y regresión. Los árboles de decisión son estructuras de flujo de control que utilizan una serie de decisiones en forma de nodos para llegar a una conclusión o tomar una acción.

En el caso de un árbol de decisión para clasificación, cada nodo interno representa una característica o atributo, y las ramas salientes representan los posibles valores que puede tomar esa característica. Las hojas del árbol representan las clases o categorías de salida. El proceso de construcción de un árbol de decisión implica dividir recursivamente el conjunto de datos en función de las características más informativas para clasificar correctamente las instancias [5][6] , en la *Figura 9* se observa el código usado para implementar este modelo.

```
from sklearn.tree import DecisionTreeRegressor
decisionTree = DecisionTreeRegressor(max_depth=8)
decisionTree.fit(z_train,w_train)
```

Figura 9. Árbol de decisión

5.3 Redes neuronales

Para el primer método no supervisado se opta por una red neuronal definido como se muestra en la *Figura 10*, y para la implementación se hace para 100

épocas y el rendimiento de este modelo se observa mediante la función de pérdida

```
class PimaClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.hidden1 = nn.Linear(45, 12)
        self.act1 = nn.ReLU()
        self.hidden2 = nn.Linear(12, 8)
        self.act2 = nn.ReLU()
        self.output = nn.Linear(8, 1)
        self.act_output = nn.Sigmoid()

    def forward(self, x):
        x = self.act1(self.hidden1(x))
        x = self.act2(self.hidden2(x))
        x = self.act_output(self.output(x))
        return x

model = PimaClassifier()
print(model)
```

Figura 10. estructura red neuronal

5.3 PCA

El análisis de componentes principales o PCA por sus siglas en inglés (*Principal component analysis*) se trata de un método no supervisado, mayormente utilizado en el ámbito del aprendizaje automático y la estadística con el fin de reducir de manera significativa la dimensión del conjunto de datos respectivo. Ahora bien, el principal objetivo de dicho método se basa en hallar una representación más compacta, en donde se conserve de igual forma, la mayor cantidad de información posible.

De manera general, el PCA tiene como función la transformación de un conjunto de variables correlacionadas en un nuevo conjunto de variables no correlacionadas, las cuales se llamaran componentes principales, los cuales se organizarán de manera descendente a partir de la varianza de los datos originales.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
pca.fit(X)
print ("sklearn PCA components")
print (pca.components_)
print ("brute force components")
print (v1)
print (v2)
```

```
sklearn PCA components
[[-0.68114194 -0.73215139]]
brute force components
[0.67850941 0.73459171]
[ 0.72373404 -0.69007901]
```

Figura 11. PCA

6. CURVAS DE APRENDIZAJE

6.1 Métodos supervisados

Para obtener la curva de aprendizaje de los algoritmos se usa el módulo learning_curves de la librería de sk.learns.model_selection, la cual implementa una metodología de cross-validation para evaluar diferentes desempeños usando diferentes tamaños para el entrenamiento del modelo. La *Figura 10* muestra la gráfica de aprendizaje para la regresión lineal. Se puede observar que se tiene un desempeño parecido en entrenamiento y test y es un desempeño bueno. Por otra parte en la *Figura 11* se observa la curva de aprendizaje de modelo de árbol de decisión en la cual el desempeño es parecido pero, a diferencia de la curva de regresión lineal, el desempeño no es tan bueno.

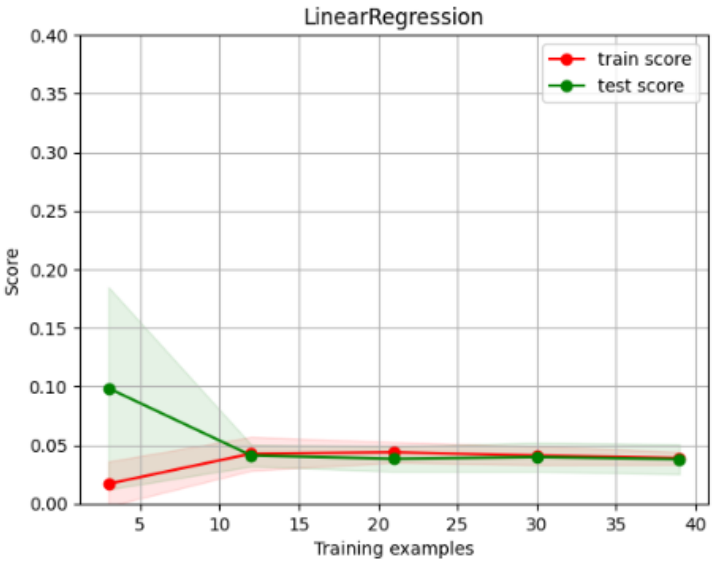

Figura 11. Curva de aprendizaje regresión lineal

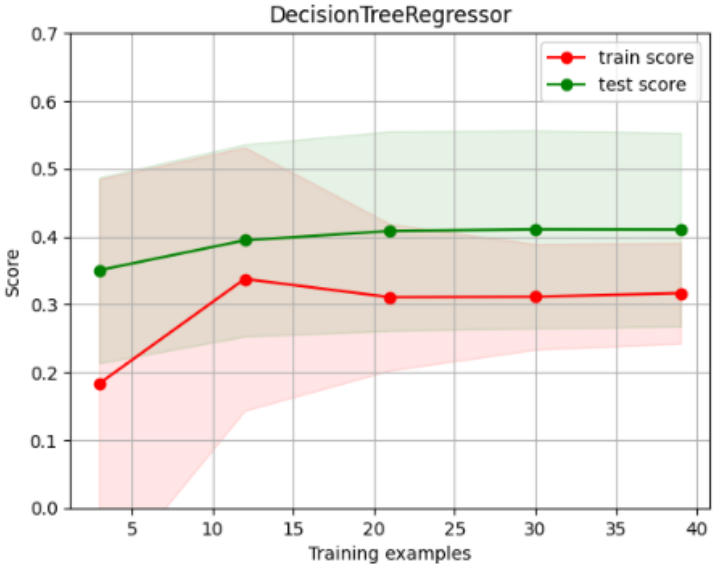

Figura 12. Curva de aprendizaje árbol de decisión

7. RETOS Y MEJORAS DEL MODELO

Para poder evaluar el comportamiento de los modelos creados es necesario tener más datos ya que al agrupar los datos solo se tenían 71 sujetos, de igual manera es posible ver que la correlación entre los datos es muy débil lo que afectará a la precisión de las predicciones de los modelos generados.

8. CONCLUSIONES

- Los modelos de inteligencia artificial son de gran ayuda a la hora de querer predecir alguna variable dependiente de interés, por lo tanto la creación de estos modelos se convierte en una herramienta de apoyo en las diferentes áreas del conocimiento.
- La cantidad de datos permite mayor aprendizaje y puede evitar el sobreajuste del modelo.
- El uso de métodos supervisados en la inteligencia artificial son de gran importancia para el aprendizaje de patrones y la toma de decisiones informadas, es decir, decisiones basadas en ejemplos que se le dan, permitiendo así, la predicción y generalización con base a dichos ejemplos para usarse en ámbitos como el reconocimiento de imágenes y recomendaciones en diferentes temas.
- Los métodos no supervisados representan una gran labor en la IA, dado que con estos se permite el descubrimiento de patrones ocultos, e incluso, la exploración y comprensión de datos.

BIBLIOGRAFÍA

[1] J. Ildefonso (2019). ”Sedentarismo, la enfermedad del siglo XXI” [En línea]. Dinsponible en: https://www.sciencedirect.com/science/article/abs/pii/S0214916819300543

[2] S. Yfantidou, C. Karagianni. et al (2023). “Lifesnaps Fitbit dataset” [En línea]. Disponible en: Lifesnaps Fitbit dataset | Kaggle

[3] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to Linear Regression Analysis. Wiley.

[4] Wooldridge, J. M. (2016). Introductory Econometrics: A Modern Approach. Cengage Learning.

[5] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th edition). Morgan Kaufmann.

[6] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd edition). Morgan Kaufmann.