Trabajo Aprendizaje Supervisado Automático

May 18, 2025

- Autor: Francisco Herrera Barajas

1 Introducción

En este trabajo vamos a utilizar un base de datos sobre un test para detectar la depresión. Este test recoge información autoreportada sobre la frecuencia de distintos síntomas relacionados con la depresión. Asímismo esta compuesto por 14 preguntas con escala likert de 6 puntos. Cada pregunta sirve para evaluar un síntoma. El número de sujetos es 491.

Nuestro objetivo sera ver cual es el síntoma que tiene mayor peso a la hora de predecir el nivel de depresión de un individuo. Para ello usaremos un algortimo de aprendizaje automático, los árboles de decisión.

Base de datos utilizada: https://www.kaggle.com/datasets/hamjashaikh/mental-health-detection-dataset

1.1 Definiciones de las variables

Number: Identificador único de cada paciente.

Sleep: Frecuencia de alteraciones del sueño.

Appetite: Cambios en el apetito.

Interest: Pérdida de interés en actividades.

Fatigue: Sensación de fatiga o baja energía.

Worthlessness: Sentimientos de inutilidad o culpa excesiva.

Concentration: Dificultad para concentrarse.

Agitation: Agitación física.

Suicidal Ideation: Pensamientos de autolesión o suicidio.

Sleep Disturbance: Problemas para dormir.

Aggression: Sentimientos de agresividad.

Panic Attacks: Experiencias de ataques de pánico.

Hopelessness: Sentimientos de desesperanza.

Restlessness: Sensación de inquietud.

Low Energy: Falta de energía.

Depression State: Estado general de depresión (categorías: 'Sin depresión', 'Leve', 'Moderada', 'Grave').

2 Presentación de los datos

Primero cojemos los paquetes que vamos a necesitar.

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪StratifiedKFold, cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

Cargamos la base de datos.

[ ]:
```
df = pd.read_csv("Deepression.csv")
df.head()
```

[ ]:

	Number	Sleep	Appetite	Interest	Fatigue	Worthlessness	Concentration	\
0	1	1.0	1.0	1.0	5.0	5.0	1.0	
1	2	2.0	5.0	5.0	1.0	1.0	5.0	
2	3	5.0	2.0	2.0	2.0	2.0	2.0	
3	4	1.0	1.0	1.0	5.0	5.0	1.0	
4	5	2.0	5.0	5.0	1.0	1.0	5.0	

	Agitation	Suicidal Ideation	Sleep Disturbance	Aggression	Panic Attacks	\
0	5.0	5.0	1.0	5.0	5.0	
1	1.0	1.0	5.0	1.0	1.0	
2	2.0	2.0	2.0	2.0	2.0	
3	5.0	5.0	1.0	5.0	5.0	
4	1.0	1.0	5.0	1.0	1.0	

	Hopelessness	Restlessness	Low Energy	Depression State
0	5.0	5.0	5.0	Mild
1	1.0	1.0	1.0	Moderate
2	2.0	2.0	2.0	Severe
3	5.0	5.0	5.0	No depression
4	1.0	1.0	1.0	Moderate

Arreglamos los datos:

1) Convertimos la variable Depression State en variables númerica.

2) Ponemos un orden lógico para el resto de variables:

Antes -> (1: Never, 2: Always, 3: Often, 4: Rarely, 5: Sometimes, 6: Not at all)

Ahora -> (0: Not at all, 1: Never, 2: Rarely, 3: Sometimes, 4: Often, 5: Always)

3) Eliminamos los valores faltantes.

[ ]:
```
mapping = {'No depression': 0, 'Mild': 1, 'Moderate': 2, 'Severe': 3}
df['depression_level'] = df['Depression State'].map(mapping)
```

[ ]:
```
orden_logico = {
    6: 0,  # Not at all
    1: 1,  # Never
    4: 2,  # Rarely
    5: 3,  # Sometimes
    3: 4,  # Often
    2: 5   # Always
}

columnas_ordinales = [
    'Sleep', 'Appetite', 'Interest', 'Fatigue', 'Worthlessness',
    'Concentration', 'Agitation', 'Suicidal Ideation', 'Sleep Disturbance',
    'Aggression', 'Panic Attacks', 'Hopelessness', 'Restlessness', 'Low Energy'
]

for col in columnas_ordinales:
    df[col] = df[col].map(orden_logico)

df.head()
```

[ ]:

	Number	Sleep	Appetite	Interest	Fatigue	Worthlessness	Concentration	\
0	1	1.0	1.0	1.0	3.0	3.0	1.0	
1	2	5.0	3.0	3.0	1.0	1.0	3.0	
2	3	3.0	5.0	5.0	5.0	5.0	5.0	
3	4	1.0	1.0	1.0	3.0	3.0	1.0	
4	5	5.0	3.0	3.0	1.0	1.0	3.0	

	Agitation	Suicidal Ideation	Sleep Disturbance	Aggression	Panic Attacks	\
0	3.0	3.0	1.0	3.0	3.0	
1	1.0	1.0	3.0	1.0	1.0	
2	5.0	5.0	5.0	5.0	5.0	
3	3.0	3.0	1.0	3.0	3.0	
4	1.0	1.0	3.0	1.0	1.0	

	Hopelessness	Restlessness	Low Energy	Depression State	depression_level
0	3.0	3.0	3.0	Mild	1.0
1	1.0	1.0	1.0	Moderate	2.0
2	5.0	5.0	5.0	Severe	3.0

```
3           3.0           3.0           3.0    No depression                0.0
4           1.0           1.0           1.0         Moderate                2.0
```

```
[ ]: df = df.dropna()
```

```
[ ]: df.shape
```

```
[ ]: (491, 17)
```

Separamos las variables predictoras (X) y variable objetivo (y)

```
[ ]: X = df.drop(columns=['Depression State', 'depression_level','Number '])
     y = df['depression_level']
```

3 Elección del algoritmo y justificación

La elección de los árboles de decisión esta justificada por varias razones:

Primero por la facilidad de interpretación que se tiene respecto a otros algoritmos de aprendizaje automático. Ya que un árbol de decisión aparte de tener un enfoque predictivo, permite de forma clara crear una serie de reglas lógicas derivadas de los datos. Además tiene un formato muy visual que ayuda enormemente a su interpretación y comunicación de resultados.

Otra ventaja es su capacidad de manejar variables ordinales sin tener que realizar niguna transformación.

Finalmente que decir que los árboles de decisión son capaces de capturar relaciones complejas y no lineales que pueden ayudar a ver los problemas de manera diferente.

4 Validación Cruzada

Definimos el modelo

```
[ ]: clf = DecisionTreeClassifier(max_depth=5, random_state=42)
```

Realizamos la validación cruzada con 5 k-folds

```
[ ]: cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

5 Ajuste de hiperpárametros

Los parámetros principales de los árboles de decisión son:

1) La función para medir la ganáncia de información: Entrópia o Gini.
2) La máxima profundidad del árbol.
3) Mínimo número de muestras para dividir un nodo.
4) Muestras mínimas que debe tener una hoja.

Para decidir los párametros utilizamos GridSearchCV

[ ]:
```
param_grid = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 3, 5],
    'criterion': ['gini', 'entropy']
}

grid = GridSearchCV(
    estimator=DecisionTreeClassifier(random_state=42),
    param_grid=param_grid,
    cv=cv,
    scoring='accuracy',
    n_jobs=-1
)
grid.fit(X, y)

print("Mejores hiperparámetros encontrados:", grid.best_params_)
print("Mejor precisión media en CV:", grid.best_score_)
```

```
Mejores hiperparámetros encontrados: {'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 1, 'min_samples_split': 2}
Mejor precisión media en CV: 0.4603174603174603
```

Volvemos a instanciar el modelo con estos párametros

[ ]:
```
best_model = grid.best_estimator_
y_pred_cv = cross_val_predict(best_model, X, y, cv=cv)
```

6 Métricas utilizadas y justificación

6.1 Métricas globales

[ ]:
```
cv_scores = cross_val_score(best_model, X, y, cv=5, scoring='accuracy')

print(f"Precisión media: {cv_scores.mean():.2f}")
print(f"Desviación estándar: {cv_scores.std():.2f}")
```

```
Precisión media: 0.44
Desviación estándar: 0.17
```

La precisión de la media significa que el modelo acierta un 44% cuando se evalúa en datos no vistos dentro del propio conjunto. Aunque no es muy alta, al menos es casi el doble que si se acertase por azar, $1/4 = 0,25$.

La desviación estándar no esta indicando que el modelo funciona bastante parecido en cada división de los datos.

6.2 Matriz de confusión

[ ]:
```
print("Evaluación del modelo:")
print("Precisión global:", accuracy_score(y, y_pred_cv))
print("Reporte de clasificación:\n", classification_report(y, y_pred_cv))
print("Matriz de confusión:\n", confusion_matrix(y, y_pred_cv))
```

```
Evaluación del modelo:
Precisión global: 0.46028513238289204
Reporte de clasificación:
              precision    recall  f1-score   support

         0.0       0.47      0.66      0.55       157
         1.0       0.59      0.33      0.43       117
         2.0       0.58      0.28      0.38       109
         3.0       0.34      0.48      0.40       108

    accuracy                           0.46       491
   macro avg       0.50      0.44      0.44       491
weighted avg       0.50      0.46      0.45       491

Matriz de confusión:
 [[104  12   8  33]
 [ 40  39   6  32]
 [ 36   8  31  34]
 [ 41   7   8  52]]
```

Esto resultados son entre los valores predecidos en el modelo y los valores de los datos reales.

La primera tabla que vemos nos indica la precisión, entendida casos positivos correctos sobre los predichos como positivos. La mayoría de valores muestra valores cercanos al 50% o algo por debajo.

El recall o sensibilidad casos positivos detectados sobre todos los positivos reales. Según estos datos vemos que para el nivel 0, no tener depresión, la sensibilidad es alta. En cambio para el nivel 1 y 2 es baja.

El f1-score es una medida que mide un compromiso entre precisión y recall. Vemos un valor superior al 50% en el nivel 0, cercano en el 1 y 3. Mientras que el 2 son bajos.

Globalmente estos resultas nos muestran que el modelo es aceptable.

6.3 Importancia de las variables

[ ]:
```
importances = best_model.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns

print("Importancia de características:")
for idx in indices:
    print(f"{feature_names[idx]}: {importances[idx]:.3f}")
```

```
Importancia de características:
Concentration: 0.355
Appetite: 0.165
Suicidal Ideation: 0.112
Panic Attacks: 0.087
Aggression: 0.076
Low Energy: 0.071
Worthlessness: 0.058
Sleep: 0.037
Interest: 0.028
Sleep Disturbance: 0.013
Hopelessness: 0.000
Restlessness: 0.000
Agitation: 0.000
Fatigue: 0.000
```

Esta lista nos muestra cuales son las variables más importantes en el modelo para reducir la entropía en cada división. Es decir que variables son más útiles para predecir los valores finales de la variable dependiente en el modelo.

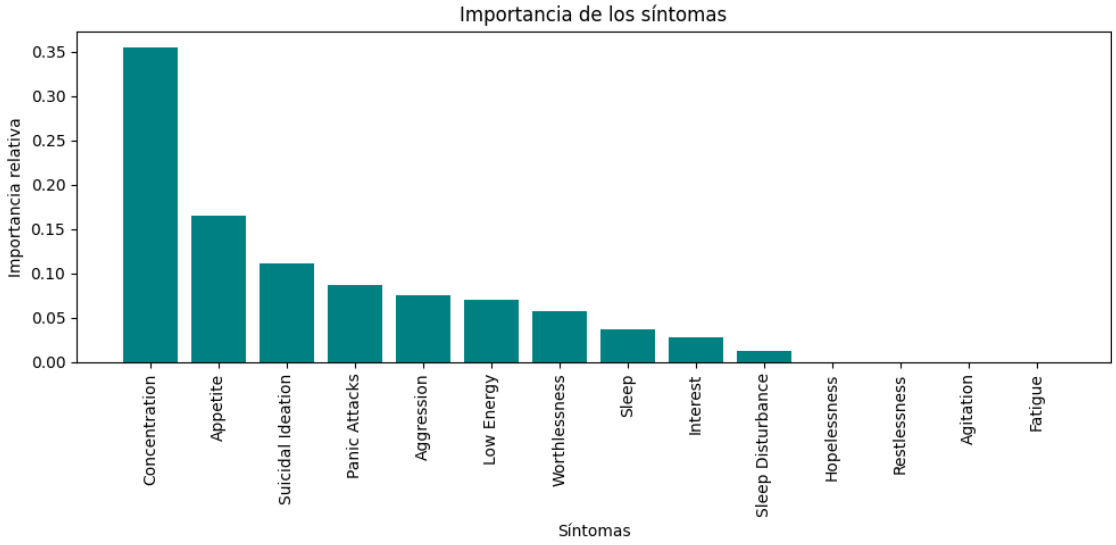
En nuestro caso sería la variable concentration sería la que tiene el mayo peso. Esa variable mide las dificultades en concentración, es decir a mayor valor más dificultad a la hora de concentrarse. Appetite y Suicidal Ideation completan los primeros puesto.

7 Principales resultados

7.1 Importancia de las variables

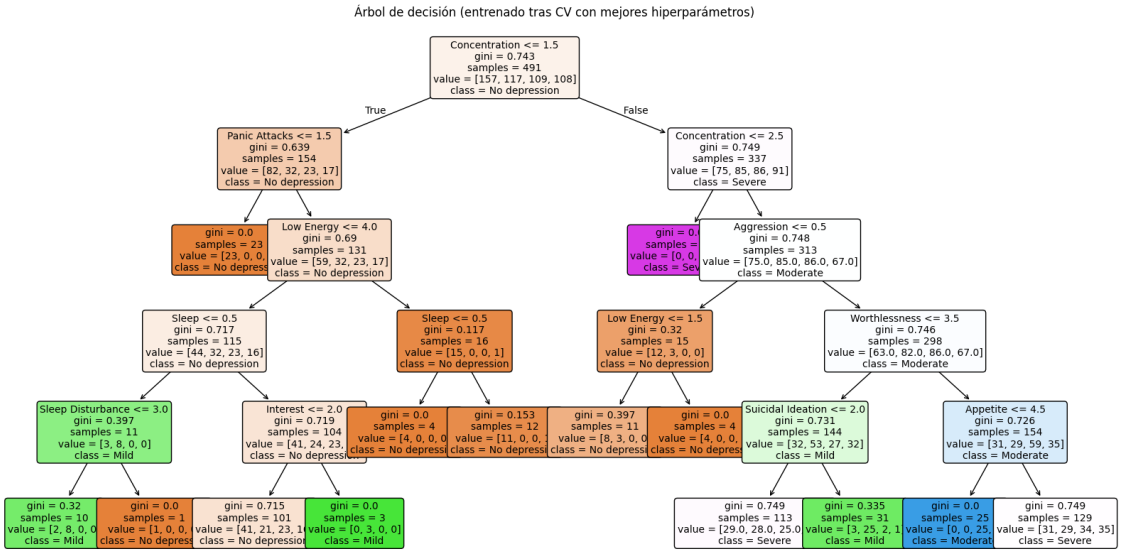
Este gráfico es la lista que expusimos antes

[ ]:
```
plt.figure(figsize=(10, 5))
plt.bar(feature_names[indices], importances[indices], color='teal')
plt.xticks(rotation=90)
plt.xlabel("Síntomas")
plt.ylabel("Importancia relativa")
plt.title("Importancia de los síntomas")
plt.tight_layout()
plt.show()
```



7.2 Árbol de decisión

[ ]:
```
plt.figure(figsize=(20, 10))
plot_tree(
    best_model,
    feature_names=feature_names,
    class_names=["No depression", "Mild", "Moderate", "Severe"],
    filled=True,
    rounded=True,
    fontsize=10
)
plt.title("Árbol de decisión (entrenado tras CV con mejores hiperparámetros)")
plt.show()
```



Este es el árbol de decisión para los síntomas que ha generado el modelo.

En el podemos ver diferentes síntomas pueden ir diferenciando los niveles de depresión. A medida que avanzamos por el árbol podemos ver diferentes combinaciones de síntomas que conducen a una clasificación concreta. Por ejemplo si tenemos unas dificultades para concentrarnos baja <= 1.5 y pocas experiencias de ataques de pánico <=1.5 podíamos descartar la depresión en la muestra.

8 Conclusiones

Como conclusión estos resultados no pueden ser considerados concluyentes por diversas razones. En primer lugar, existen numerosas incógnitas sobre los datos analizados. Se desconoce su origen, el cuestionario utilizado para su obtención, los ítems que lo componían y las propiedades psicométricas del instrumento. Además, no se dispone de información sobre la composición de la muestra, lo que plantea dudas sobre su representatividad, e incluso cabe la posibilidad de que se trate de datos ficticios.

Por otro lado, el tamaño muestral resulta claramente insuficiente para extraer conclusiones válidas o generalizables. Así como, el modelo muestra malas propiedades como una precisión no muy alta.

Por todas estas razones, no podemos sacar ningún tipo de conclusión acerca de estos. Aún así me ha parecido interesante, como ejercicio a la hora de ir utilizando algoritmos de aprendizaje automático para explorar datos, además de ver las diferentes posibilidades que podemos encontrar con estos.

9 Bonus Random Forest

Me pareció interesante probar un Random forest y una regresión regularización con LASSO con estos datos para comparar. A diferencia del árbol de decisión no lo he realizado con

[ ]:
```
rf_param_grid = {
    'n_estimators': [100],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 3],
    'criterion': ['gini', 'entropy']
}


grid_rf = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=rf_param_grid,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    scoring='accuracy',
    n_jobs=-1
)
grid_rf.fit(X, y)

print("RANDOM FOREST")
print("Mejores hiperparámetros encontrados:", grid_rf.best_params_)
print(f"Precisión media con CV: {grid_rf.best_score_:.2f}")
```

```
RANDOM FOREST
Mejores hiperparámetros encontrados: {'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Precisión media con CV: 0.45
```

[ ]:
```
best_rf = grid_rf.best_estimator_
y_pred_cv = cross_val_predict(best_rf, X, y, cv=cv)
```

[ ]:
```
cv_scores = cross_val_score(best_rf, X, y, cv=cv, scoring='accuracy')
print(f"Precisión: {cv_scores.mean():.2f}")
print(f"Desviación estándar: {cv_scores.std():.2f}")
```

```
Precisión: 0.45
Desviación estándar: 0.02
```

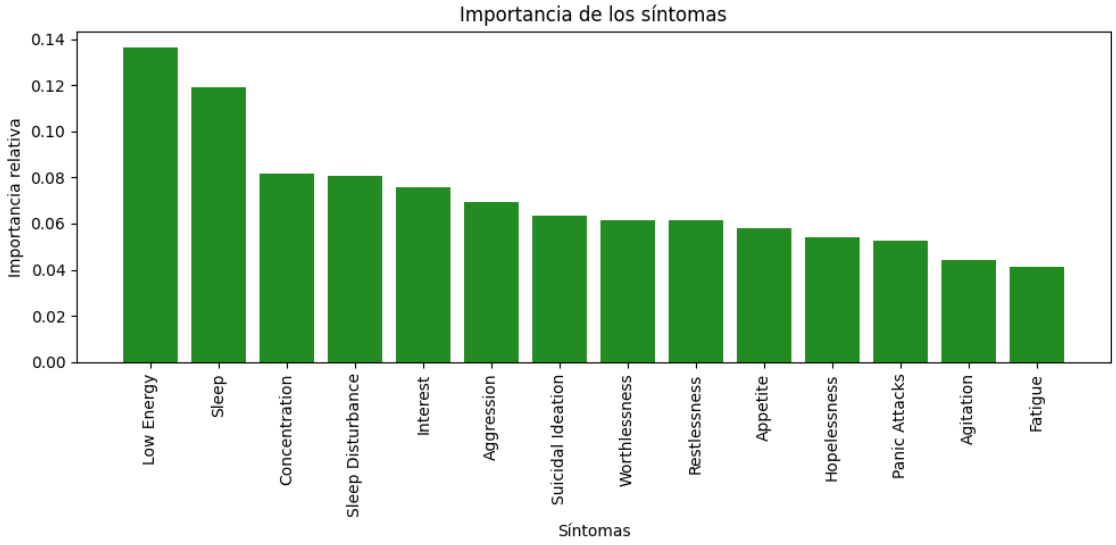
[ ]:
```
print("Evaluación basada 100% en validación cruzada:")
print(f"Precisión: {accuracy_score(y, y_pred_cv):.2f}")
print("Reporte de clasificación:\n", classification_report(y, y_pred_cv))
print("Matriz de confusión:\n", confusion_matrix(y, y_pred_cv))
```

```
Evaluación basada 100% en validación cruzada:
Precisión: 0.45
Reporte de clasificación:
              precision    recall  f1-score   support

         0.0       0.45      0.68      0.54       157
         1.0       0.68      0.29      0.41       117
```

```
           2.0       0.35      0.45      0.40       109
           3.0       0.50      0.30      0.37       108

      accuracy                           0.45       491
     macro avg       0.49      0.43      0.43       491
  weighted avg       0.49      0.45      0.44       491

Matriz de confusión:
 [[106   9  32  10]
 [ 42  34  29  12]
 [ 48   2  49  10]
 [ 42   5  29  32]]
```

[ ]:
```python
feature_names_rf = X.columns
importances_rf = best_rf.feature_importances_
indices_rf = np.argsort(importances_rf)[::-1]

print("Importancia de características:")
for idx in indices_rf:
    print(f"{feature_names_rf[idx]}: {importances_rf[idx]:.3f}")

# 6. Visualización de importancia
plt.figure(figsize=(10, 5))
plt.bar(feature_names_rf[indices_rf], importances_rf[indices_rf],␣
 ↪color='forestgreen')
plt.xticks(rotation=90)
plt.xlabel("Síntomas")
plt.ylabel("Importancia relativa")
plt.title("Importancia de los síntomas")
plt.tight_layout()
plt.show()
```

```
Importancia de características:
Low Energy: 0.137
Sleep: 0.119
Concentration: 0.082
Sleep Disturbance: 0.081
Interest: 0.076
Aggression: 0.069
Suicidal Ideation: 0.063
Worthlessness: 0.062
Restlessness: 0.062
Appetite: 0.058
Hopelessness: 0.054
Panic Attacks: 0.052
Agitation: 0.044
Fatigue: 0.041
```



La precisión es parecida pero la importancia de los síntomas a variado sensiblemente.