UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROYECTO FINAL

"ANÁLISIS DE DATOS DE DATASET DE ENFERMEDADES CARDÍACAS DE LA UCI (Unidad de Cuidados Intensivos)"

MATERIA: INF - 354 INTELIGENCIA ARTIFICIAL
DOCENTE: Ph. D. MOISES SILVA
UNIVERSITARIA: LUZ MICAELA VAQUEROS APAZA

LA PAZ – BOLIVIA
2024

INDICE

Clasificación con Árbol de Decisión y Técnicas de Preprocesamiento de datos de UCI

1. Introducción

Este artículo detalla un proceso de análisis de datos, con enfoque en la clasificación usando técnicas como Árbol de Decisión y preprocesamiento avanzado, incluyendo la imputación de datos faltantes, balanceo de clases, normalización de datos y reducción de dimensionalidad mediante PCA (Análisis de Componentes Principales). Además, se describe a fondo el conjunto de datos utilizado y el objetivo de investigación que guía el proyecto. A continuación, se explican los pasos tomados en el proyecto, desde la carga del conjunto de datos hasta la evaluación del modelo.

2. Descripción del conjunto de datos

El conjunto de datos utilizado en este proyecto proviene de un archivo csv. ,nombrado (Examen) cargado en el entorno de trabajo de Colab . El conjunto de datos contiene varias columnas, que representan características de las observaciones, y una columna objetivo (num) que contiene las etiquetas que se intentan predecir. Las principales columnas del conjunto de datos incluyen tanto variables numéricas como categóricas. Es importante notar que algunas columnas presentan valores nulos, lo que requiere ser tratado durante el preprocesamiento.

A continuación, se detallan las principales características del conjunto de datos:

- **Columnas numéricas**: Estas columnas contienen datos cuantitativos que son fundamentales para las predicciones. Por ejemplo, pueden incluir medidas de rendimiento o características de las observaciones que son continuas en la naturaleza.
- **Columnas categóricas**: Contienen categorías o clases que deben ser transformadas a un formato numérico para que los modelos de aprendizaje automático puedan procesarlas. Este tipo de datos puede incluir características como el sexo, la región o el tipo de clase de una observación.
- **Valores nulos**: El conjunto de datos presenta algunas columnas con valores faltantes. La cantidad de valores nulos varía según la columna, por lo que se toma una decisión específica sobre cómo

tratarlos. Para las columnas numéricas, se imputa la media, mientras que para las categóricas se imputa la moda.

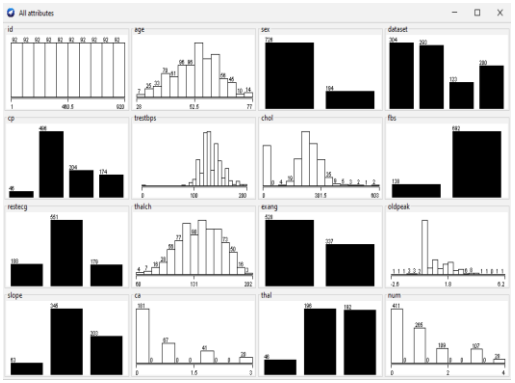


Fig: Weka procesamiento de Datos

Elaboracion propia

3. Objetivo de la investigación

El objetivo principal de este proyecto es desarrollar un modelo de clasificación que permita predecir la variable objetivo (num) en función de las demás características del conjunto de datos. A partir de este análisis, se busca lograr un modelo eficiente y preciso, capaz de generalizar bien a datos nuevos, utilizando técnicas de preprocesamiento, como la imputación de valores nulos, balanceo de clases, normalización y reducción de dimensionalidad.

En términos más específicos, la investigación tiene como fin:

1. **Clasificación y predicción**: Utilizar técnicas de aprendizaje automático, particularmente el Árbol de Decisión, para predecir la variable num. Esta variable objetivo podría representar, por ejemplo, categorías como el nivel de riesgo o el tipo de grupo al que pertenece cada observación.
2. **Mejora del rendimiento**: Emplear técnicas de preprocesamiento como SMOTE para balancear las clases, PCA para reducir la dimensionalidad, y la normalización de los datos para mejorar la precisión y el tiempo de ejecución del modelo.
3. **Análisis de impacto**: Evaluar cómo diferentes configuraciones del modelo (como el número de componentes principales en PCA) impactan en la precisión del modelo, y determinar el mejor enfoque para lograr la mejor predicción posible.

El conjunto de datos elegido y el objetivo de investigación se centran en un problema de clasificación supervisada, donde el modelo aprende de un conjunto de datos etiquetados para hacer predicciones sobre nuevas observaciones. En este caso, el propósito es

entender cómo las características del conjunto de datos interactúan entre sí y cómo se pueden optimizar los procesos de predicción.

4. Carga y Exploración de Datos

El primer paso en cualquier proyecto de análisis de datos es la carga del conjunto de datos y la exploración inicial de su estructura. En este caso, utilizamos pandas para cargar el archivo CSV y visualizamos las primeras filas para verificar las columnas disponibles y el tipo de datos:

código python

```
data = pd.read_csv
("/content/Drive/MyDrive/datos/examen.csv")
print(data.head())
print(data.columns)
print(data.info())
```

Con esta exploración, obtenemos una comprensión general de las columnas presentes y podemos detectar posibles problemas, como valores nulos o columnas no relevantes.

5. Manejo de Valores Nulos

El manejo de valores nulos es crucial para evitar que los modelos de aprendizaje automático produzcan resultados erróneos. En este caso, se implementó una estrategia diferenciada para las columnas numéricas y categóricas:

- **Columnas numéricas**: Si los valores nulos son menores al 50% de la columna, se imputa la media de esa columna.
- **Columnas categóricas**: Los valores nulos se imputan con la moda, es decir, el valor más frecuente de la columna.

código python

```
for col in numerical_cols:
    if data[col].isnull().mean() < 0.5:
        data[col]                                =
data[col].fillna(data[col].mean())
    else:
        data = data.drop(columns=[col])

for col in categorical_cols:
    if data[col].isnull().mean() < 0.5:
        data[col]                                =
data[col].fillna(data[col].mode()[0])
    else:
        data = data.drop(columns=[col])
```

De esta forma, se garantizó que las columnas con datos nulos no afectarán el proceso de análisis y clasificación.

6. Codificación de Variables Categóricas

Las variables categóricas fueron codificadas utilizando LabelEncoder, que asignan valores numéricos a las categorías de las columnas, permitiendo que puedan ser procesadas por los modelos de aprendizaje automático:

código python

```
encoder = LabelEncoder()
for col in categorical_cols:
    data[col] = encoder.fit_transform(data[col])
```

Este paso es fundamental ya que la mayoría de los algoritmos de aprendizaje automático requieren que los datos sean numéricos.

7. División en Variables Predictoras y Objetivo

A continuación, se definieron las variables predictoras (X) y la variable objetivo (y). En este caso, la columna numfue seleccionada como la columna objetivo:

código python

```
X = data.drop(column_objetivo, axis=1)
y = data[column_objetivo]
```

Con esto, preparamos los datos para la fase de entrenamiento y prueba.

8. Balanceo de Clases con SMOTE

El conjunto de datos puede estar desequilibrado, lo que afectará negativamente el rendimiento del modelo. Para abordar este problema, se utilizó SMOTE (Synthetic Minority Over-sampling Technique), que genera muestras sintéticas de la clase minoritaria:

código python

```
smote = SMOTE()
X_balanced, y_balanced = smote.fit_resample(X, y)
```

Este paso es crucial para mejorar la capacidad de predicción del modelo, especialmente cuando se enfrenta a conjuntos de datos desequilibrados donde una clase es mucho más representada que la otra.

9. Normalización de datos

La normalización de los datos es otro paso importante en el preprocesamiento, ya que muchas técnicas de aprendizaje automático, como los árboles de decisión, pueden beneficiar de tener los datos escalados. Se utiliza StandardScalerpara normalizar las características, asegurando que todas tengan media cero y desviación estándar uno:

código python

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)
```

Esto ayuda a evitar que características con rangos muy grandes o pequeños afecten de manera desproporcionada al modelo.

10. Reducción de Dimensionalidad con PCA

La reducción de dimensionalidad se implementó utilizando PCA (Análisis de Componentes Principales), lo que permite reducir el número de características en los datos sin perder demasiada información. Esto es útil para mejorar el rendimiento del modelo y la interpretabilidad de los datos:

código python

```
pca = PCA(n_components=10)  # Probar con diferentes valores de n_components
X_pca = pca.fit_transform(X_scaled)
```

En este caso, se probaron varios valores de componentes principales (10, 12, 9, 5, 3) para determinar cuántos componentes principales eran suficientes para explicar la mayor parte de la variabilidad de los datos.

11. División en Conjuntos de Entrenamiento y Prueba

La división del conjunto de datos en conjuntos de entrenamiento y prueba es una etapa fundamental para la validación del modelo. En este proyecto, se utilizó una división de 80% para entrenamiento y 20% para prueba:

código python

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y_balanced, test_size=0.2, random_state=42)
```

Esto asegura que el modelo se entrene con una gran parte de los datos, pero se evalúa con datos que no ha visto antes, lo que

proporciona una evaluación más realista de su desempeño.

12. Entrenamiento del Modelo con Árbol de Decisión

El modelo seleccionado para este análisis fue un Árbol de Decisión, que es un algoritmo de clasificación fácil de entender e interpretar. Se entrenó con los datos de entrenamiento utilizando el siguiente código :

código python

```
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

Este modelo crea una estructura de árbol para tomar decisiones basadas en las características de los datos de entrada, y se utiliza combinadas en problemas de clasificación.

13. Predicciones y Evaluación del Modelo

Una vez entrenado el modelo, se realizaron predicciones sobre el conjunto de prueba y se evaluó el rendimiento del modelo utilizando dos métricas principales: **precisión** y **matriz de confusión** .

código python

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Matriz de Confusión:")
print(conf_matrix)
```

- **Precisión (Accuracy)** : Mide el porcentaje de predicciones correctas sobre el total de predicciones realizadas.
- **Matriz de Confusión** : Muestra el desempeño del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Además, se visualizó la matriz de confusión para una interpretación más clara de los resultados:

código python

```
plt.figure(figsize=(6,6))
plt.imshow(conf_matrix,
cmap='Blues', interpolation='nearest')
plt.title("Matriz de Confusión")
plt.colorbar()
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.xticks([0, 1], ["No", "Sí"])
plt.yticks([0, 1], ["No", "Sí"])
plt.show()
```

14. Evaluación con Diferentes Componentes Principales

Finalmente, se probó el modelo con diferentes cantidades de componentes principales en PCA para evaluar su impacto en la precisión del modelo. Esto es importante, ya que la reducción de dimensionalidad puede mejorar el rendimiento del modelo al eliminar ruido en los datos.

código python

```
component_counts = [12, 10, 9, 5, 3]
for n in component_counts:
    pca = PCA(n_components=n)
    X_pca = pca.fit_transform(X_scaled)
    X_train, X_test, y_train, y_test =
train_test_split(X_pca, y_balanced,
test_size=0.2, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Accuracy con {n} componentes
principales: {accuracy_score(y_test, y_pred)}")
```

Este análisis permite ajustar el modelo y encontrar la cantidad óptima de componentes principales para maximizar su desempeño.

Conclusión

Este proyecto ha demostrado cómo se puede abordar un problema de clasificación utilizando técnicas de preprocesamiento avanzadas y modelos de aprendizaje automático como el Árbol de Decisión. A través de la imputación de valores nulos, la normalización de los datos, el balanceo de clases con SMOTE y la reducción de dimensionalidad con PCA, se optimizó el proceso de clasificación y se logró un modelo con un rendimiento sólido.

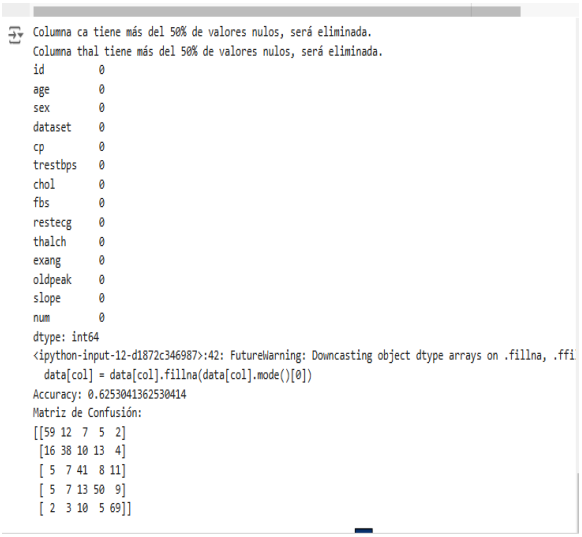
Además, se explorará cómo las diferentes configuraciones del modelo, como el número de componentes principales en PCA, afectan la precisión del modelo, lo que permite tomar decisiones informadas sobre cómo ajustar el modelo en futuros trabajos.

Este enfoque puede ser aplicado a otros problemas de clasificación con características similares, y se destaca la importancia del preprocesamiento de datos y la evaluación exhaustiva de modelos en el desarrollo de soluciones efectivas de aprendizaje automático.

Referencias

- Redwankar, RS (2021). *Datos sobre enfermedades cardíacas* . Kaggle. Recuperadohttps ://www.kaggl.com/datos/redwankarimsony / datos sobre enfermedades del corazón
- Colab: https://colab.research.google.com/drive/13SY2nGnsc9AA6qmojxYRG895vK9rIOZb?usp=sharing&authuser=2#scrollTo=YagvScVtronM
- **Github:** https://github.com/MicoVaqueros/DAT354_P1

Anexos



FigResultados de los preprocesamiento de Datos UCI

Elaboracion propia

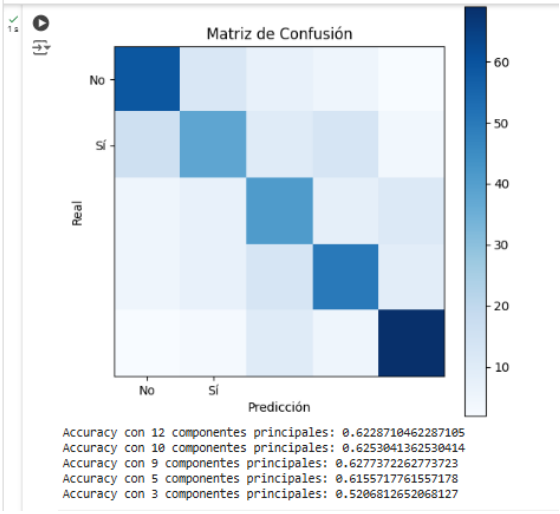


Fig: Matriz de Confusion de Preprocesamiento de Datos

Elaboracion propia