UNIVERSIDAD DE ANTIOQUIA

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL PARA LAS CIENCIAS E INGENIERÍAS

PROYECTO DE CURSO – INFORME FINAL -

LEONARDO MARÍN RESTREPO, 98633850
WILLIAM ALEXANDER TORRES ZAMBRANO, 71784722

DOCENTE

RAÚL RAMOS POLLÁN

FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
MAYO 28 DE 2023

INFORME FINAL: Predict_students_dropout_and_academic_success

1. Introducción

El presente trabajo tiene como objetivo abordar el problema de clasificación en un contexto específico, con el propósito de predecir la categoría a la que pertenecen ciertos elementos de un conjunto de datos. Para ello, se analizará el desempeño de dos modelos de clasificación: Bosques Aleatorios y Regresión Logística.

El problema en estudio se enfoca en la clasificación de estudiantes en tres categorías: "abandono", "graduado" y "matriculado". Se cuenta con un conjunto de datos que incluye diversas características de los estudiantes, como su historial académico, información socioeconómica y otros atributos relevantes. El objetivo es desarrollar modelos de clasificación que puedan predecir con precisión la categoría a la que pertenece un estudiante en función de estas características.

En este trabajo, se explorarán dos enfoques de clasificación: Bosques Aleatorios y Regresión Logística. Estos modelos son ampliamente utilizados en problemas de clasificación debido a su eficacia y capacidad para manejar conjuntos de datos complejos. Se comparará el desempeño de ambos modelos en términos de precisión, recall, F1-score y otras métricas relevantes. La evaluación y comparación de estos modelos permitirá determinar cuál de ellos es más adecuado para abordar el problema de clasificación de estudiantes en este contexto específico. Asimismo, se analizarán las conclusiones derivadas de dicha comparación y se discutirán las implicaciones y posibles aplicaciones de los resultados obtenidos.

En resumen, este trabajo presenta un análisis detallado del problema de clasificación de estudiantes utilizando los modelos de Bosques Aleatorios y Regresión Logística. A través de la evaluación comparativa de ambos modelos, se busca proporcionar recomendaciones basadas en evidencia para abordar este tipo de problemas y contribuir al avance de la investigación en el campo de la clasificación de datos.

2. Exploración descriptiva del dataset

Antes de abordar cualquier análisis o modelo de clasificación, es fundamental realizar una exploración inicial del conjunto de datos disponible. Esta etapa permite comprender mejor la naturaleza de los datos, identificar posibles problemas o inconsistencias, y obtener una visión general de las características que conforman el dataset.

En el caso del presente estudio, se dispone de un conjunto de datos que contiene información sobre estudiantes y diversas variables relacionadas con su desempeño académico y características personales. Durante la exploración inicial, se llevan a cabo una serie de pasos para obtener una comprensión más completa del dataset:

- Análisis de la estructura: Se examina la estructura del dataset, incluyendo la cantidad de registros y columnas presentes. Esto proporciona una idea inicial de la magnitud y complejidad de los datos.
- Inspección de las variables: Se analizan las variables presentes en el dataset, revisando sus nombres, tipos de datos y posibles categorías. Esta etapa permite identificar las características relevantes que se utilizarán en el proceso de clasificación.
- Estadísticas descriptivas: Se calculan estadísticas descriptivas para cada variable, como la media, la mediana, el mínimo y el máximo. Esto proporciona información sobre la distribución de los datos y posibles valores atípicos.
- Manejo de datos faltantes: Se evalúa la presencia de datos faltantes en el dataset y se decide cómo abordarlos. Dependiendo de la cantidad y la naturaleza de los datos faltantes, se pueden aplicar técnicas como la imputación de valores o la eliminación de registros incompletos.
- Análisis de correlaciones: Se examina la correlación entre las diferentes variables del dataset. Esto permite identificar posibles relaciones o dependencias entre las características, lo cual puede ser útil en la selección de variables para el modelo de clasificación.

La exploración inicial del dataset es un paso crucial en cualquier proyecto de análisis y modelado de datos. Proporciona una base sólida para comprender la naturaleza de los datos y tomar decisiones informadas sobre las técnicas de preprocesamiento y modelado que se aplicarán. Además, ayuda a detectar posibles problemas o limitaciones del dataset, lo que contribuye a la calidad y fiabilidad de los resultados obtenidos en el proceso de clasificación.

En nuestro proyecto tenemos una exploración inicial:

```
[10] import pandas as pd

     # Cargar el archivo CSV
     data = pd.read_csv('dataset.csv')

     # Mostrar las primeras filas del dataset
     print("Primeras filas del dataset:")
     print(data.head())

     # Obtener información general del dataset
     print("\nInformación del dataset:")
     print(data.info())

     # Calcular estadísticas descriptivas del dataset
     print("\nEstadísticas descriptivas del dataset:")
     print(data.describe())

     # Contar los valores únicos en cada columna
     print("\nValores únicos en cada columna:")
     for column in data.columns:
         unique_values = data[column].nunique()
         print(f"{column}: {unique_values} valores únicos")

     # Mostrar la cantidad de valores faltantes en cada columna
     print("\nValores faltantes en cada columna:")
     print(data.isnull().sum())
```

```
[13] #Exploración gráfica del dataset original de Kaggle
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Gráfico de barras para mostrar la distribución de variables categóricas
     categorical_columns = ['Marital status', 'Application mode', 'Course', 'Daytime/evening attendance', 'Previous qualification', 'Nacionality'
     for column in categorical_columns:
         plt.figure(figsize=(10, 6))
         sns.countplot(data=data, x=column)
         plt.title(f"Distribución de {column}")
         plt.xticks(rotation=90)
         plt.show()

     # Gráfico de histograma para variables numéricas
     numeric_columns = ['Application order', 'Age at enrollment', 'Curricular units 1st sem (credited)', 'Curricular units 1st sem (enrolled)', '
     for column in numeric_columns:
         plt.figure(figsize=(10, 6))
         sns.histplot(data=data, x=column, kde=True)
         plt.title(f"Histograma de {column}")
         plt.show()
```

PRE-PROCESAMIENTO DE LOS DATOS:

Teniendo en cuenta las instrucciones del proyecto, el dataset seleccionado ha de cumplir con los siguientes requisitos:

- al menos 5000 instancias (filas, imágenes, etc.)
- al menos 30 columnas
- al menos el 10% de las columnas han de ser categóricas
- al menos ha de tener un 5% de datos faltantes en al menos el 3 columnas.

Hacemos lo siguiente con el dataset cargado desde kaggle:

[14]
```
import os

# Obtener la ruta actual del notebook
current_path = os.getcwd()

# Concatenar la ruta actual con el nombre del archivo
file_path = os.path.join(current_path, 'dataset.csv')

# Imprimir la ruta del archivo
print(file_path)
```

```
/content/kaggle-cli/dataset.csv
```

[15]
```
import pandas as pd
import numpy as np

# cargar el archivo CSV
df = pd.read_csv('/content/kaggle-cli/dataset.csv')

# eliminar datos al azar de las columnas indicadas
columnas = ['Marital status', 'Previous qualification', 'Gender']
porcentaje_faltante = 0.05

for columna in columnas:
    n_filas = len(df)
    n_eliminar = int(n_filas * porcentaje_faltante)
    indices_eliminar = np.random.choice(n_filas, n_eliminar, replace=False)
    df.loc[indices_eliminar, columna] = np.nan

# guardar el archivo modificado
df.to_csv('newdataset.csv', index=False)
```

[16]
```
import pandas as pd

# cargar el archivo modificado
df_new = pd.read_csv('/content/kaggle-cli/newdataset.csv')

# revisar cantidad de valores faltantes en las columnas de interés
columnas = ['Marital status', 'Previous qualification', 'Gender']
for col in columnas:
    n_missing = df_new[col].isna().sum()
    print(f"La columna {col} tiene {n_missing} valores faltantes.")
```

```
La columna Marital status tiene 221 valores faltantes.
La columna Previous qualification tiene 221 valores faltantes.
La columna Gender tiene 221 valores faltantes.
```

ANÁLISIS EXPLORATORIO DE LOS DATOS

```
[17] # Importar las bibliotecas necesarias:
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[18] # Cargar los datos desde el archivo CSV:
     df = pd.read_csv('/content/kaggle-cli/newdataset.csv')
```

```
[19] # Verificar la estructura de los datos:
     df.head() # muestra las primeras filas del dataframe
     df.info() # muestra información sobre las columnas y tipos de datos
     df.describe() # muestra estadísticas descriptivas de las variables numéricas
```

```
# Visualizar la distribución de los datos:
sns.histplot(df['Age at enrollment'], kde=False) # histograma de la variable Age at enrollment
sns.boxplot(x='Gender', y='Previous qualification', data=df) # diagrama de caja de la variable Calificación por género
```

```
#Explorar relaciones entre variables:
sns.pairplot(df[['Gender', 'Previous qualification', 'Debtor']]) # gráfico de dispersión de las variables Gender, Previous qualification e Debtor
sns.heatmap(df.corr(), annot=True) # mapa de calor de la matriz de correlación entre variables numéricas
```

```
[22] import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[23] # cargar el archivo CSV
     df = pd.read_csv('/content/kaggle-cli/newdataset.csv')

     # mostrar las primeras filas del dataset
     print(df.head())

     # obtener información general del dataset
     print(df.info())

     # obtener estadísticas descriptivas del dataset
     print(df.describe())

     # obtener la distribución de cada variable
     for col in df.columns:
         sns.displot(df[col])
         plt.title(f"Distribución de {col}")
         plt.show()
```

3. Iteraciones de desarrollo

MODELO DE PREDICCIÓN USANDO CLASIFICACIÓN RANDOMFORESTCLASSIFIER

Se utiliza el modelo de clasificación RandomForestClassifier para realizar la predicción. Es un algoritmo de aprendizaje automático basado en el ensamblaje de árboles de decisión aleatorios.

Los datos se dividieron en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca scikit-learn. Esta función permite realizar una división aleatoria de los datos en subconjuntos de entrenamiento y prueba.

Mediante la línea X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42),

X_imputed representa las características del dataset con los datos faltantes completados utilizando la técnica del promedio, y y representa la variable objetivo "Target". Se utiliza un test_size de 0.2, lo que implica que el 20% de los datos se utilizarán como conjunto de prueba y el 80% restante se utilizará como conjunto de entrenamiento. El parámetro random_state se establece en 42 para garantizar la reproducibilidad de la división de los datos.

Después de dividir los datos, se utilizan X_train y y_train como el conjunto de entrenamiento para entrenar el modelo, y X_test se utiliza como el conjunto de prueba para evaluar el rendimiento del modelo

```
import pandas as pd

# Cargar el dataset original desde el archivo CSV
dataset = pd.read_csv("/content/kaggle-cli/newdataset.csv")

# Crear la columna "Target" en el dataset
dataset["Target"] = ""

# Aplicar la lógica para asignar la categoría a cada estudiante
# Puedes utilizar condiciones y reglas según los atributos y características disponibles en tu dataset
# Por ejemplo:
# Si el estudiante es un deudor y tiene tasas de matrícula al día, se considera "matriculado"
# Si el estudiante tiene una cualificación previa y unidades curriculares aprobadas en al menos un semestre, se considera "graduado"
# Si ninguna de las condiciones anteriores se cumple, se considera "abandono"
dataset.loc[(dataset["Debtor"] == "Yes") & (dataset["Tuition fees up to date"] == "Yes"), "Target"] = "matriculado"
dataset.loc[(dataset["Previous qualification"].notnull()) & ((dataset["Curricular units 1st sem (approved)"] > 0) | (dataset["Curricular units 2nd sem (approved)"] > 0)), "Target"] = "graduado"
dataset.loc[dataset["Target"] == "", "Target"] = "abandono"

# Guardar el dataset modificado en un nuevo archivo CSV
dataset.to_csv("/content/kaggle-cli/newdataset_modified.csv", index=False)
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.preprocessing import LabelEncoder

# Cargar el dataset modificado desde el archivo CSV
dataset = pd.read_csv("/content/kaggle-cli/newdataset_modified.csv")

# Separar los datos en características (X) y variable objetivo (y)
X = dataset.drop("Target", axis=1)
y = dataset["Target"]

# Codificar las etiquetas de manera numérica
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Completar los datos faltantes con el promedio de cada columna
imputer = SimpleImputer(strategy="mean")
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Crear y entrenar el modelo de clasificación
model = RandomForestClassifier(random_state=42)
model.fit(X_train_imputed, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test_imputed)

# Calcular la precisión del modelo en porcentaje
accuracy = accuracy_score(y_test, y_pred) * 100

# Calcular el área bajo la curva (AUC)
auc = roc_auc_score(y_test, y_pred)

print("Precisión del modelo: %.2f%%" % accuracy)
print("Área bajo la curva (AUC): %.2f" % auc)
```

```
Precisión del modelo: 99.10%
Área bajo la curva (AUC): 0.98
```

MODELO DE REGRESIÓN LOGÍSTICA (LOGISTICREGRESSION)

Se utiliza el modelo de Regresión Logística (LogisticRegression) en lugar del modelo de Bosques Aleatorios. El resto del código es similar al anterior, realizando el preprocesamiento de datos, entrenando el modelo, haciendo predicciones y calculando la precisión y el AUC del modelo de Regresión Logística.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.preprocessing import LabelEncoder

# Cargar el dataset modificado desde el archivo CSV
dataset = pd.read_csv("/content/kaggle-cli/newdataset_modified.csv")

# Separar los datos en características (X) y variable objetivo (y)
X = dataset.drop("Target", axis=1)
y = dataset["Target"]

# Codificar las etiquetas de manera numérica
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Completar los datos faltantes con el promedio de cada columna
imputer = SimpleImputer(strategy="mean")
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Crear y entrenar el modelo de Regresión Logística
model = LogisticRegression(random_state=42)
model.fit(X_train_imputed, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test_imputed)

# Calcular la precisión del modelo en porcentaje
accuracy = accuracy_score(y_test, y_pred) * 100

# Calcular el área bajo la curva (AUC)
auc = roc_auc_score(y_test, y_pred)

print("Precisión del modelo de Regresión Logística: %.2f%%" % accuracy)
print("Área bajo la curva (AUC) del modelo de Regresión Logística: %.2f" % auc)
```

```
Precisión del modelo de Regresión Logística: 94.80%
Área bajo la curva (AUC) del modelo de Regresión Logística: 0.88
```

Comparación entre los dos modelos:

Se utilizan los modelos de Bosques Aleatorios (RandomForestClassifier) y Regresión Logística (LogisticRegression) para hacer predicciones en el conjunto de prueba (X_test_imputed). Luego, se genera un informe de clasificación utilizando la función classification_report para cada modelo, comparando las predicciones con las etiquetas verdaderas (y_test). Este informe proporciona métricas como precisión, recall, F1-score y soporte para cada clase de la variable objetivo. Puedes utilizar este informe para comparar el desempeño de los dos modelos en términos de sus predicciones.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import classification_report

# Cargar el dataset modificado desde el archivo CSV
dataset = pd.read_csv("/content/kaggle-cli/newdataset_modified.csv")

# Separar los datos en características (X) y variable objetivo (y)
X = dataset.drop("Target", axis=1)
y = dataset["Target"]

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Completar los datos faltantes con el promedio de cada columna
imputer = SimpleImputer(strategy="mean")
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Crear y entrenar el modelo de Bosques Aleatorios
random_forest_model = RandomForestClassifier(random_state=42)
random_forest_model.fit(X_train_imputed, y_train)

# Realizar predicciones con el modelo de Bosques Aleatorios
y_pred_random_forest = random_forest_model.predict(X_test_imputed)

# Crear y entrenar el modelo de Regresión Logística
logistic_regression_model = LogisticRegression(random_state=42)
logistic_regression_model.fit(X_train_imputed, y_train)

# Realizar predicciones con el modelo de Regresión Logística
y_pred_logistic_regression = logistic_regression_model.predict(X_test_imputed)

# Comparar las predicciones
print("Informe de clasificación para el modelo de Bosques Aleatorios:")
print(classification_report(y_test, y_pred_random_forest))

print("Informe de clasificación para el modelo de Regresión Logística:")
print(classification_report(y_test, y_pred_logistic_regression))
```

Luego de correr los dos modelos se obtuvo:

```
Informe de clasificación para el modelo de Bosques Aleatorios:
              precision    recall  f1-score   support

    abandono       1.00      0.95      0.98       174
    graduado       0.99      1.00      0.99       711

    accuracy                           0.99       885
   macro avg       0.99      0.98      0.99       885
weighted avg       0.99      0.99      0.99       885

Informe de clasificación para el modelo de Regresión Logística:
              precision    recall  f1-score   support

    abandono       0.96      0.76      0.85       174
    graduado       0.95      0.99      0.97       711

    accuracy                           0.95       885
   macro avg       0.95      0.88      0.91       885
weighted avg       0.95      0.95      0.95       885
```

4. Retos y consideraciones de despliegue

Consideramos los siguiente Retos/Consideraciones:

1. Diversidad de datos
2. Limpieza y preprocesamiento de datos
3. Selección de características relevantes
4. Manejo de desequilibrio de clases
5. Evaluación de modelos y selección de métricas adecuadas
6. Interpretación y explicabilidad de los modelos

Estos retos y consideraciones son solo algunos ejemplos relevantes al analizar el problema en cuestión. Dependiendo del contexto específico y las características del dataset, pueden surgir otros desafíos adicionales que deben abordarse durante el despliegue de los modelos de clasificación. Es fundamental abordar estos aspectos para garantizar la confiabilidad y eficacia de las soluciones propuestas.

5. Conclusiones

Al comparar los informes de clasificación de los dos modelos, puedes extraer las siguientes conclusiones:

- Precisión general: El modelo de Bosques Aleatorios tiene una precisión general del 99%, mientras que el modelo de Regresión Logística tiene una precisión del 95%. Esto indica que el modelo de Bosques Aleatorios tiende a clasificar correctamente un mayor porcentaje de instancias en general.
- Precisión en la clase "abandono": El modelo de Bosques Aleatorios tiene una precisión del 100% en la clasificación de la clase "abandono", mientras que el modelo de Regresión Logística tiene una precisión del 96%. Esto sugiere que el modelo de Bosques Aleatorios tiene una menor tasa de falsos positivos en la clase "abandono".
- Recall en la clase "graduado": Ambos modelos tienen un recall alto en la clasificación de la clase "graduado", con un 100% para el modelo de Bosques Aleatorios y un 99% para el modelo de Regresión Logística. Esto indica que ambos modelos tienen una alta capacidad para identificar correctamente los casos de "graduado".
- F1-score: El modelo de Bosques Aleatorios muestra un F1-score del 99% en general, mientras que el modelo de Regresión Logística tiene un F1-score del 95%. Esto sugiere que el modelo de Bosques Aleatorios tiene un mejor equilibrio entre precisión y recall en comparación con el modelo de Regresión Logística.
- Soporte: El soporte indica el número de instancias en cada clase. En este caso, el soporte es similar en ambas clases para ambos modelos. Sin embargo, el modelo de Bosques Aleatorios muestra un mejor rendimiento en términos de métricas de evaluación para ambas clases, lo que indica una mayor capacidad de generalización.

En general, el modelo de Bosques Aleatorios parece tener un mejor desempeño en términos de precisión, recall, F1-score y capacidad de clasificación en comparación con el modelo de Regresión Logística. Sin embargo, es importante tener en cuenta otros factores, como la interpretación del modelo y los requisitos específicos del problema, al tomar una decisión final sobre cuál modelo elegir.

Referencias

Curso de Introducción a la Inteligencia Artificial para las Ciencias e Ingeniería
https://rramosp.github.io/ai4eng.v1/content/M00_intro_udea.html

Predict students' dropout and academic success

https://www.kaggle.com/datasets/thedevastator/higher-education-predictors-of-student-retention