Autor: SHANIA ZHARICK GOMEZ CONEO - 2210688

Introducción

Motivación

Las encuestas longitudinales son herramientas cruciales para analizar cambios en temas específicos a lo largo del tiempo. Estas encuestas permiten mantener a los mismos sujetos en mediciones periódicas, lo que ayuda a controlar errores de medición y evaluar condiciones que afectan los cambios buscados.

Objetivos

En este cuaderno, se utilizará la base de datos de la Encuesta Longitudinal de Protección Social (ELPS) realizada en Colombia en 2012 para responder a preguntas específicas sobre la protección social de los menores de 5 años. Se analizarán aspectos relacionados con su alimentación, cuidado y asistencia a instituciones educativas.

Metodología

Modelos Usados

Para este análisis, se utilizarán varios modelos estadísticos y de aprendizaje automático, incluyendo:

1. Análisis descriptivo
2. Regresión logística
3. Árboles de decisión

Argumentos en las Funciones Implementadas

Las funciones implementadas se basarán en los datos proporcionados por la ELPS y estarán diseñadas para responder a las siguientes preguntas:

- ¿Cuáles son los factores que influyen en la nutrición de los menores de 5 años?
- ¿Qué variables afectan la probabilidad de que un menor asista a una institución educativa?

Cargar datos:

```
from google.colab import files

# Subir el archivo desde tu computadora
uploaded = files.upload()

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# !rm *
!ls
```

```
'D. MENORES.txt'   sample_data
```

Resultados

Descripción de los Resultados

A continuación se presentan los resultados obtenidos de los análisis realizados. Estos incluyen gráficos y tablas que ilustran las principales conclusiones del estudio.

1)

Utilice las variables de las columnas "P51", "P52", "P55", "P56", "P57", "P6159s8", "P6163s8", "P6161" y "P6161s1". En el análisis se deben eliminar las filas con datos faltantes de la columna "P52". También se debe tener en cuenta que los valores faltantes de las columnas "P6159s8" y "P6163s8" denotan que la madre y el padre sí realizan alguna actividad con el menor, respectivamente.

```
df = pd.read_csv('D. MENORES.txt', sep='\t', encoding='utf-8')
```

```
df.head()
```

	Directorio	Nro_encuesta	Secuencia_encuesta	Secuencia_p	Orden	P51	P52	P52s1	P53	P54	.
0	447	247	4	1	4	4	5.0	NaN	NaN	NaN	
1	17255	247	3	1	3	1	NaN	NaN	1.0	4.0	
2	373	247	5	1	5	1	NaN	NaN	4.0	4.0	
3	373	247	4	1	4	1	NaN	NaN	4.0	4.0	
4	226	247	3	1	3	2	5.0	NaN	NaN	NaN	

5 rows × 53 columns

```
variables_seleccionadas = ["P51", "P52", "P55", "P56", "P57", "P6159s8", "P6163s8", "P6161", "P6161s1"
df_selected = df[variables_seleccionadas]

df_clean = df_selected.dropna(subset=['P52'])

df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2724 entries, 0 to 4090
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   P51      2724 non-null   int64
 1   P52      2724 non-null   float64
 2   P55      2724 non-null   int64
 3   P56      2724 non-null   int64
 4   P57      2724 non-null   int64
 5   P6159s8  180 non-null    float64
 6   P6163s8  700 non-null    float64
 7   P6161    2724 non-null   int64
 8   P6161s1  2210 non-null   float64
dtypes: float64(4), int64(5)
memory usage: 212.8 KB
```

```
# Estadísticas descriptivas
summary_stats = df_clean.describe()
print(summary_stats)
```

```
                P51          P52          P55          P56          P57  \
count  2724.000000  2724.000000  2724.000000  2724.000000  2724.000000
mean      2.683921     4.284508     1.066814     1.116373     1.066814
std       1.349754     1.224746     0.249745     0.320731     0.249745
min       2.000000     1.000000     1.000000     1.000000     1.000000
25%       2.000000     4.000000     1.000000     1.000000     1.000000
50%       2.000000     5.000000     1.000000     1.000000     1.000000
75%       2.000000     5.000000     1.000000     1.000000     1.000000
max       8.000000     6.000000     2.000000     2.000000     2.000000

       P6159s8  P6163s8        P6161      P6161s1
count    180.0    700.0  2724.000000  2210.000000
mean       1.0      1.0     1.188693     2.895928
std        0.0      0.0     0.391336     2.097883
min        1.0      1.0     1.000000     1.000000
25%        1.0      1.0     1.000000     2.000000
50%        1.0      1.0     1.000000     2.000000
75%        1.0      1.0     1.000000     3.000000
max        1.0      1.0     2.000000    12.000000
```

2)

Tome un conjunto de datos de entrenamiento y de validación (del 20%), teniendo en cuenta que las entradas corresponden a las variables de las columnas "P51", "P52", "P55", "P56", "P57", "P6159s8" y "P6163s8", y, la salida es la variable "P6161".

```
from sklearn.model_selection import train_test_split

variables_entrada = ["P51", "P52", "P55", "P56", "P57", "P6159s8", "P6163s8"]
variable_salida = "P6161"

X = df[variables_entrada]
y = df[variable_salida]


data_clean = pd.concat([X, y], axis=1).dropna()

#(80% entrenamiento, 20% validación)
X_train, X_val, y_train, y_val = train_test_split(data_clean[variables_entrada], data_clean[variable_sa


print("Tamaño del conjunto de entrenamiento:", X_train.shape)
print("Tamaño del conjunto de validación:", X_val.shape)
```

```
Tamaño del conjunto de entrenamiento: (107, 7)
Tamaño del conjunto de validación: (27, 7)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


modelo = LinearRegression()


modelo.fit(X_train, y_train)


y_pred = modelo.predict(X_val)


mse = mean_squared_error(y_val, y_pred)
print("Error cuadrático medio en conjunto de validación:", mse)
```

```
Error cuadrático medio en conjunto de validación: 0.20512473150800142
```

```
X_dummies = pd.get_dummies(X, drop_first=True)

print("Dimensiones de X_dummies:", X_dummies.shape)
```

```
Dimensiones de X_dummies: (4091, 7)
```

```
from sklearn.model_selection import train_test_split

#(80% entrenamiento, 20% validación)
X_train, X_val, y_train, y_val = train_test_split(X_dummies, y, test_size=0.2, random_state=42)

print("Tamaño del conjunto de entrenamiento:", X_train.shape)
print("Tamaño del conjunto de validación:", X_val.shape)
```

```
Tamaño del conjunto de entrenamiento: (3272, 7)
Tamaño del conjunto de validación: (819, 7)
```

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')

X_train_imputed = imputer.fit_transform(X_train)

X_val_imputed = imputer.transform(X_val)
```

3)

Implemente una red neuronal multicapa. Recuerde que todas las variables de entrada son cualitativas, por lo tanto, pueden transformarlas a variables tipo dummie. Además, se trata de una red de clasificación ya que la variable de la columna "P6161" es cualitativa: debe usar la función MLPClassifier.

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score


modelo_mlp = MLPClassifier(hidden_layer_sizes=(50, 50), max_iter=500, random_state=42)
modelo_mlp.fit(X_train_imputed, y_train)

y_pred = modelo_mlp.predict(X_val_imputed)


accuracy = accuracy_score(y_val, y_pred)
print("Exactitud en conjunto de validación después de imputación:", accuracy)
```

```
Exactitud en conjunto de validación después de imputación: 0.8485958485958486
```

```
import pandas as pd
from sklearn.model_selection import train_test_split


variables_entrada = ["P51", "P52", "P55", "P56", "P57", "P6159s8", "P6163s8"]
variable_salida = "P6161"

X = df[variables_entrada]
y = df[variable_salida]


X_dummies = pd.get_dummies(X, drop_first=True)


X_train, X_val, y_train, y_val = train_test_split(X_dummies, y, test_size=0.2, random_state=42)


print("Tamaño del conjunto de entrenamiento:", X_train.shape)
print("Tamaño del conjunto de validación:", X_val.shape)
```

```
Tamaño del conjunto de entrenamiento: (3272, 7)
Tamaño del conjunto de validación: (819, 7)
```

```
from sklearn.impute import SimpleImputer


imputer = SimpleImputer(strategy='mean')


X_train_imputed = imputer.fit_transform(X_train)


X_val_imputed = imputer.transform(X_val)
```

4)

A. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


modelo_rf = RandomForestClassifier(n_estimators=100, random_state=42)
modelo_rf.fit(X_train_imputed, y_train)


y_pred_rf = modelo_rf.predict(X_val_imputed)


accuracy_rf = accuracy_score(y_val, y_pred_rf)
print("Exactitud del Random Forest en conjunto de validación después de imputación:", accuracy_rf)
```

```
Exactitud del Random Forest en conjunto de validación después de imputación: 0.8376068376068376
```

B. Naive Bayes Classifier (GaussianNB)

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.impute import SimpleImputer


imputer = SimpleImputer(strategy='mean')


X_train_imputed = imputer.fit_transform(X_train)


X_val_imputed = imputer.transform(X_val)


modelo_nb = GaussianNB()
modelo_nb.fit(X_train_imputed, y_train)


y_pred_nb = modelo_nb.predict(X_val_imputed)


accuracy_nb = accuracy_score(y_val, y_pred_nb)
print("Exactitud del Naive Bayes en conjunto de validación después de imputación:", accuracy_nb)
```

```
Exactitud del Naive Bayes en conjunto de validación después de imputación: 0.7997557997557998
```

Support Vector Machine (SVM)

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.impute import SimpleImputer
```

5)

```
X_train_imputed = imputer.fit_transform(X_train)
```

Encuentre, de acuerdo con las predicciones del mejor modelo de acuerdo con el ítem anterior, los valores de las variables de las columnas "P51", "P52", "P55", "P56", "P57", "P6159s8" y "P6163s8" qué menores de edad no ha recibido citas de control de crecimiento y desarrollo.

El mejor modelo de acuerdo a las predicciones es SVM maquina de soporte vectorial con 0.8437118437118437 de exactitud.

```
# [redacted]
```

Bibliografía y Referencias