Ejercicio 4 – Implementación en Python

Parte 1: carga de datos y preparación

1. Descargar de Webasignatura el dataset de ejemplo “cardiac-training.csv”

2. Crear un archivo Python y agregar las dependencias necesarias

```
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
```

3. Leer el dataset desde el archivo CSV utilizando la librería Pandas. Y ver como esta compuesto.

```
input_file = " cardiac-training.csv"
df = pd.read_csv(input_file, header=0)
print(df.values)
```

4. Obtener a partir del dataset los datos y las clases.

```
X = df.loc[:, df.columns != '2do_Ataque_Corazon']
y = df['2do_Ataque_Corazon'].values
```

Parte 2: entrenamiento y testing

5. Dividir el conjunto de datos en 2, uno para entrenamiento y otro para prueba.

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.30, random_state=
0, shuffle=True)
```

6. Analizar cómo se utiliza el método de Logistic Regression y documentarlo (alternativas, parámetros). Entregar el documento.

7. Crear el un modelo de LR y entrenarlo.

```
lr = LogisticRegression()
lr = lr.fit(train_X, train_y)
```

Parte 3: evaluación

8. Predecir las clases para los datos del conjunto de prueba y ver los resultados.

```
y_pred = lr.predict(test_X)
print("Predicted vs Expected")
print(y_pred)
print(test_y)
```

9. Probar el modelo y ver el reporte. Observar las columnas y que significan.

```
print(classification_report(test_y, y_pred, digits=3))
```

10. Ver la matriz de confusión y analizar los resultados.

```
print(confusion_matrix(test_y, y_pred))
```

Comparaciones y extensiones:

11. Comparar con los resultados análogos obtenidos en RapidMiner
12. Entrenar un nuevo modelo utilizando todo el conjunto “cardiac-training” y aplicar el modelo a los ejemplos del conjunto “cardiac-scoring”. Comparar los resultados con los obtenidos en RapidMiner
13. Investigar cómo realizar un entrenamiento y test con cross-validation

Ejercicio 5 – Campaña de marketing directo en una institución bancaria Portuguesa

Los datos están relacionados con campañas de marketing directo de una entidad bancaria portuguesa. Las campañas de marketing se basaron en llamadas telefónicas. A menudo, se requería más de un contacto con el mismo cliente, para poder si querría subscribir al nuevo producto (depósito bancario a plazo).

Parte 1: Contexto, carga de datos y preparación

- Descarga de UCI el dataset "Bank Marketing" https://archive.ics.uci.edu/ml/datasets/bank+marketing
- Estudia el problema y los atributos, así como los trabajos de referencia
- Analiza qué atributos pueden suprimirse o combinarse, para eliminar aquéllos que no sean relevantes para el algoritmo de ML o que puedan combinarse
- Importa el dataset en RapidMiner
- Utiliza el "TurboPrep" de RapidMiner para realizar las tareas de preparación de datos previamente identificadas
- Algunos "tips":
  - Reducir la cantidad de clases diferentes en "Education" (los "basic…" podrían agruparse)
  - ¿Cómo es la distribución de la variable de salida? Tip: analizar opciones de "upsampling"[1][2][3][4]
  - Utiliza visualizaciones para analizar las relaciones entre las variables y la salida (por ejemplo, ¿cómo es la relación de la media de edad de los clientes que compraron el producto versus la media de edad de los que no lo compraron? ¿frecuencias de compra versus Job Title? ¿Estado Civil versus compra? ¿Educación versus compra? ¿día de la semana? ¿Edad?

Parte 2: Modelado y evaluación

- Utilizando cross-validation, entrena un modelo de regresión logística y evalúalo.

Parte 3: Implementación en Python

- Implementa todo el proceso utilizando Python y compara los resultados con los obtenidos en RapidMiner
- Investiga formas de visualización apropiadas.

[1] SMOTE en RapidMiner
[2] SMOTE for Imbalanced Classification with Python
[3] Upsampling with SMOTE for Classification Projects
[4] Overcoming Class Imbalance using SMOTE Techniques