ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CARRERA TSDS

ASIGNATURA:	Desarrollo de Inteligencia Artificial
PROFESOR:	Ing. Mayra Alvarez
PERÍODO ACADÉMICO:	2022-B

TAREA #2

TÍTULO:

Algoritmos de ML supervisado

NOMBRE DE ESTUDIANTE(S):

- Males Maldonado Paulina

FECHA DE ENTREGA: 13/02/2023

PROPÓSITO DE LA PRÁCTICA

- Implemente los algoritmos bayesianos, arboles de decisión, SVM (diferentes kernels), k-NN y Redes neuronales.
- Aplicar los algoritmos al dataset que usted elija (acorde a implementación de los mismos).
- Conclusión de resultados obtenidos (¿Qué algoritmo tuvo mejores resultados? En que se basa para decir “mejores resultados”).

OBJETIVO(S)

Aplicar distintos tipos de algoritmos de aprendizaje a un conjunto de datos para determinar y analizar distintas predicciones, al probar distintos métodos podremos ver cual es el de mejor resultados.

DESARROLLO

Para la implementación de todos los algoritmos se utilizo el mismo data set, el archivo csv es sobre el rendimiento que estudiantes académicamente, influenciados por distintos factores y antecedentes como alimentación, nivel de estudio de sus padres, preparación para sus exámenes, género y etnia.

```
In [4]: data = leer_datos("StudentsPerformance.csv")
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

```
In [5]: data.head()
```

Out[5]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

Ilustración 1Obtención de datos

El archivo cuenta con mil datos y 8 columnas en total, algunos tipo string, y otros tipo entero para realizar el respectivo analizas se mapeo los datos de cada columna que haya sido necesaria.

```
In [6]: # Dimensiones del dataset
        data.shape
        #cantidad de datos (filas x columnas)
```

```
Out[6]: (1000, 8)
```

Ilustración 2 Dimensiones del Dataset

En la mayoría de los algoritmos se implementó la misma librería para el análisis y mapeo de datos, dependiendo de cada algoritmo se incluía el que se necesitaría para su modelo. A continuación, se muestra los principales usados en todos los algoritmos:

Librerías

```
In [34]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion matrix
```

Ilustración 3 Librerías implementadas

Algoritmo Bayesiano

1. **Objetivo:** Este algoritmo tiene como objetivo hacer predicciones precisas y tomar decisiones informadas sobre un evento dado cierta información o evidencia.

2. **Librerías implementadas:**

- Scikit-learn: importamos la clase “GaussianNB”e importamos la clase “SelectKBest”

3. **Proceso:**

- Separaremos los datos como un conjunto de entrada y un valor de K(etiqueta).
- Estableceremos como **k** a la columna de preparación de exámenes, y las demás características como el conjunto de entrada.
- En la preparación de los datos utilizaremos la técnica Split donde lo distribuiremos aleatoriamente en un 80/20.
- Creamos el modelo y lo entrenamos con los datos de test y train.

4. **Resultados:**

   La exactitud que nos da el modelo sobre el Test es de: 0.64 lo que significaría que el algoritmo fue capaz de predecir correctamente si un estudiante se preparó o no para los exámenes en el 64% de los casos.

   Después de realizar una predicción para la evaluación del modelo nos dio una exactitud de 0.62, en este algoritmo se podría decir que es bajo el nivel de aprendizaje, ya que debajo de 0.65 el modelo solo estaría memorizando, pero muchas veces depende de las características y dispersión de datos, por lo que para este conjunto de datos de podría considerar que es un **modelo regular.**

Algoritmo de Árbol de Decisión

1. **Objetivo:** Este algoritmo busca construir un modelo de clasificación o regresión que pueda predecir una variable objetivo a partir de un conjunto de características.

2. **Librerías implementadas:**

   - Scikit-learn: importamos la clase “tree”.

3. **Proceso:**

- En la preparación de los datos utilizaremos la técnica Split donde lo distribuiremos aleatoriamente en un 70/30.
- Creamos un árbol de decisión con profundidad de 4, un parámetro “criterion” utilizado para especificar el criterio de selección utilizado para dividir los nodos del árbol, en este caso “entropy”.
- Entrenamos el modelo.

4. **Resultados:**

La exactitud de este algoritmo nos da una precisión de 65%, en este caso se puede evidenciar que el modelo no esta funcionando muy bien, y que podría haber un margen de mejora, teniendo en cuenta que el rendimiento del modelo depende de varios factores como la calidad de datos, selección de características, etc. Cabe mencionar que la configuración sobre la profundidad y criterio es la más común utilizada.

Despúes de poner a prueba al modelo con una evaluación de precisión nos arroja un resultado de exactitud 0.71, lo que es notablemente es mucho mejor, por lo que se concluye que es un modelo “bueno” dependiendo de los datos que se le brinde.

Algoritmo de Redes Neuronales

1. **Objetivo:** Este algoritmo busca crear un modelo computacional, entrenando una red neuronal que pueda realizar una tarea específica o predicciones, este modelo se inspira en la estructura y función del cerebro humano.

2. **Librerías implementadas:**

    - Keras: importamos la clase “Sequential” e importamos la clase “Dense”.

3. **Proceso:**

    - Separamos los datos en datos entrada(características), y datos de salida(etiqueta).

    - Definimos el modelo y la arquitectura de la red, en este caso redes neuronales secuenciales, creamos una capa de 64 neuronas, que requerirá una entrada de 7 características y para su activación usamos “relu” , después una capa de 32 neuronas con una activación de “relu”, por ultimo una capa de 1 neurona con activación “sigmoid”.

    - Establecemos los parámetros de la red.

    - Entrenamos el modelo con una cantidad de épocas de entrenamiento de 30, lo evaluamos.

4. **Resultados:**

El algoritmo nos da una exactitud de 0.73 siendo su valor más alto, bastante bueno considerando que no pusimos un número de épocas alto para que pueda aprender, la evaluación del modelo nos muestra una pérdida de aprendizaje de 0.52% que siendo un valor alto se sigue obteniendo resultados positivos, también cabe recalcar que al crear las capas se utilizaron distintos tipos de activación lo que puede ayudar a mejor el modelo.

Algoritmo de k-NN

1. **Objetivo:** Este algoritmo busca clasificar nuevos puntos de dato según las clases a las que pertenece sus vecinos mas cercanos en un conjunto de entrenamiento.

2. **Librerías implementadas:**

    - Matplotlib.pyplot
    - Sklearn.neighbors: Importamos la calse KNeighborsClassifier.

3. **Proceso:**

    - Creamos arrays de entrenamiento y la etiqueta de test.
    - Determinamos el mejor valor de **k** para una mejor precisión con un gráfico y en el, el rango de las k dándonos 15 el mejor resultado.
    - Creamos el modelo con el valor de k y entrenamos el modelo.

4. **Resultados:**

    El algoritmo nos da una precisión 0.72 siendo bueno, sin embargo, después de una evaluación del modelo con una predicción baja a 0.68 dándonos a entender que después de todo el modelo no esta dando una predicción muy acertada.

    Al crear una matriz de confusión notamos que:

    - 39= Número de veces que el algoritmo clasifico correctamente como "si"
    - 68=Número de veces que el algoritmo clasifico como "si" pero fue "no"
    - 22= Número de veces que el algoritmo clasifico como "no" pero fue "si"
    - 171= Número de veces que el algoritmo clasifico correctamente como "no"

Algoritmo de SVM

1. **Objetivo:** Este algoritmo busca encontrar un hiperplano que mejor separa las clases en el espacio de características. El modelo utiliza técnicas de kernel para proyectar los datos.

2. **Librerías implementadas:**

    - Skleanr.svm: importamos la clase “SVC”
    - Sklearn.metrics: importamos la clase “Accuracy_score

3. **Proceso:**

    - Separamos el conjunto de datos con la técnica de Split en un 80/20%.
    - Creamos el modelo con distintos kernels en el polinómico establecemos una función polinómica de tercer grado para mapear los datos “degree=3”, ajustamos la importancia de los términos con “coef0=1”, y un parámetro de regularización que controla la penalización por errores de clasificación, “C=5”.

En el kernel radial establecemos un parámetro que controla el ancho de banda de la función. Un valor bajo de gamma significaría una distribución amplia, mientras un valor alto significaría que la distribución es más estrecha, “gamma=0.7”. Otro parámetro es el “C=5” es de regularización que controla la penalización de errores.

- Entrenamos todos los modelos y realizamos las pruebas

4. **Resultados:**

El algoritmo nos arrojo distintos valores de precisión dependiendo el kernel utilizado, inicialmente utilizando le kernel lineal nos da 0.69% de exactitud, utilizando le kernel polinómico nos da un 0.70% de precisión y utilizando el kernel radial nos da un 0.66% de precisión.

Después de evaluar todos los modelos y realizar una predicción el modelo con kernel lineal y polinómico nos arrojan valores simirales de 0.69% por otro lado utilizando el kernel radial nos da un 0.60% siendo realmente bajo, determinando que para este conjunto de datos no es un modelo realmente favorecedor aun habiendo distintos métodos de kernerls para ser creados.

CONCLUSIONES

- Implementar distintos algoritmo de aprendizaje nos permitió comparar cada algoritmo para determinar el mejor en cuestión a la necesidad que implementemos, tomando en cuenta que no existe un algoritmo que sea le mejor para todos los problemas.
- Algunos algoritmos son más adecuados para ciertos tipos de problemas o conjunto de datos ya que dependen también de la calidad, dispersión y cantidad de datos para su análisis, algunos algoritmos son más rápidos que otros por lo que depende de lo que estamos buscando tener como resultado.
- Después de implementar todos los algoritmos al mismo conjunto de datos con la misma clasificación de características y etiquetas, datos de entrada y de salida, el mejor resultado se obtuvo con el algoritmo de **redes neuronales,** dando a entender que podría ser por la estructura del modelo y las funciones de activación que se necesitan.

BIBLIOGRAFIA:

[1] Find Open Datasets and Machine Learning Projects | Kaggle. (s.f.). Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets?tags=11105-Education&page=3

[2] Principales Algoritmos usados en Machine Learning. (s.f.). Aprende Machine Learning. https://www.aprendemachinelearning.com/principales-algoritmos-usados-en-machine-learning/

[3] D, C. (2020, 17 de diciembre). *Un Recorrido Por Los Algoritmos De Machine Learning*. DataSource.ai. https://www.datasource.ai/es/data-science-articles/un-recorrido-por-los-algoritmos-de-machine-learning