UVG
UNIVERSIDAD DEL VALLE DE GUATEMALA

Inteligencia Artificial
- Laboratorio 2 -

Semestre I - 2024

Instrucciones:

- Esta es una actividad en grupos de no más de 5 integrantes.
    - Recuerden **unirse al grupo de canvas**
- No se permitirá ni se aceptará cualquier indicio de copia. De presentarse, se procederá según el reglamento correspondiente.
- Tendrán hasta el día indicado en Canvas.
    - No se confíen, aprovechen el tiempo en clase para entender todos los ejercicios y avanzar lo más posible.

Task 1 - Preguntas Teóricas

Responda a cada de las siguientes preguntas de forma clara y lo más completamente posible.

1. ¿Por qué el modelo de Naive Bayes se le considera “naive”?
2. Explique la formulación matemática que se busca optimizar en Support Vector Machine, además responda ¿cómo funciona el truco del Kernel para este modelo? (Lo que se espera de esta pregunta es que puedan explicar en sus propias palabras la fórmula a la que llegamos que debemos optimizar de SVM en clase)
3. Investigue sobre Random Forest y responda
    a. ¿Qué tipo de ensemble learning es este modelo?
    b. ¿Cuál es la idea general detrás de Random Forest?
    c. ¿Por qué se busca baja correlación entre los árboles de Random Forest?

Task 2 - Naive Bayes: Clasificador de Mensajes Ham/Spam

Deberá construir un programa que reciba como entrada un archivo llamado “entrenamiento.txt” que será su dataset para entrenar un modelo basado en Bayes/Laplace Smoothing para clasificar mensajes como ham o spam. De dicho modelo, deberá reportar alguna métrica de desempeño. Además, con dicho modelo deberá ser capaz de interpretar mensajes futuros como spam o ham. Asimismo, deberá considerar las siguientes restricciones:

- Solamente se podrá entrenar un modelo por dataset de entrenamiento (No se pueden cargar más de un archivo para entrenar el modelo)
- Deberá limpiar el dataset de caracteres especiales y de combinaciones de mayúsculas/minúsculas.
- Cada línea representa una observación (mensaje con su respectiva categoría)
- Del dataset dado deberá dividirlo en training y test.
    - Para este punto, podrá usar librerías externas como las dadas en sklearn.
    - **NO** se aceptará el uso de librerías para la construcción del modelo principal

Task 2.1 - Lectura y limpieza del dataset

Reciba como entrada un archivo llamado “entrenamiento.txt” que tendrá una estructura como la que se muestra abajo. En dicho archivo, cada línea representa un mensaje, y la primera palabra de la línea indica si es un mensaje Ham o Spam (etiqueta/categoría del mensaje), luego encuentra una tabulación (\t) para separar la etiqueta del verdadero mensaje, finalmente está el mensaje que termina hasta que encuentre un cambio de línea (\n). En base a este archivo, usted deberá entrenar un modelo basado en Bayes con Laplace Smoothing para clasificar mensajes como spam o ham.

```
ham     Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham     Ok lar... Joking wif u oni...
spam    Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham     U dun say so early hor... U c already then say...
ham     Nah I don't think he goes to usf, he lives around here though
spam    FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
ham     Even my brother is not like to speak with me. They treat me like aids patent.
```

Además, deberá limpiar el archivo de texto, teniendo en cuenta que pueden haber caracteres especiales que solamente agreguen ruido al mensaje y otros que pueden llegar a ser de utilidad para la clasificación. De igual modo, se recomienda cambiar los mensajes a que todos sigan la misma nomenclatura, es decir, todo en mayúsculas o bien, todo en minúscula, o similar, de modo tal que pueda clasificar de mejor manera la participación de cada palabra para determinar la categoría del mensaje.

También, deberá separar el dataset con un 80% para training, 20% para testing. Si llegan a necesitar una parte para validation, pueden subdividir el 20% de testing en 10% para validation y 10% para testing. Recuerden que esta división siempre deberá ser realizada de forma aleatoria, para reproducir sus resultados usen una *seed*.

Task 2.2 - Construcción del modelo

Una vez tenga una forma de leer y limpiar el dataset, deberá crear un modelo basado en Bayes con Laplace Smoothing para que clasifique los mensajes en spam o ham. Esto deberá basarse en la probabilidad de cada palabra en pertenecer a cada uno de los posibles grupos. Cuando tenga la probabilidad del mensaje que sea spam o ham, deberá clasificarlo con la probabilidad de la categoría que resulte mayor.

Es importante tomar en cuenta que el entrenamiento y construcción del modelo deberá hacerse usando solamente la parte de training del dataset.

Cuando ya tengan entrenado su modelo, deberán probar el modelo usando la parte de testing del dataset. La métrica a utilizar deberá proponerla considerando la distribución de clases/categorías en el dataset. Recuerde dejar justificada su respuesta en los comentarios de su código.

Presente al final del entrenamiento, la métrica de desempeño sobre el subset de training y sobre el subset de testing.

Task 2.3 - Clasificación de mensajes futuros

Al finalizar el entrenamiento de su modelo, permita que se ingresen nuevos mensajes de forma individual desde su interfaz (ya sea en consola o en una interfaz gráfica). Y que estos se clasifiquen usando su modelo. Su programa deberá poder recibir cualquier tipo de mensaje y devolver la probabilidad de que sea spam y ham, así mismo como cuál es la decisión de clasificación de su modelo.

Task 2.4 - Comparación con Librerías

Entrene un modelo de Bayes usando alguna librería ya existente como sklearn.naive_bayes.MultinomialNB en Python. Este deberá usar el mismo dataset dado, y deberá además dividirlo en training y testing como en el Task 1.2. Compare la métrica de desempeño tanto en training como en testing con la misma implementación que usted realizó. Responda como un comentario en su código:

- ¿Cuál implementación lo hizo mejor? ¿Su implementación o la de la librería?
- ¿Por qué cree que se debe esta diferencia?

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Inteligencia Artificial - Laboratorio 2 -

Task 3 - Clasificación de Partidas de League of Legends

League of Legends es un juego online multijugador, que suele ser categorizado como un MOBA (multiplayer online battle arena). Este se desarrolla en base a dos equipos (azul y rojo), en el cual 5 jugadores pertenecen a cada equipo y cada uno tiene un rol diferente, habiendo entonces 5 roles diferentes. En el campo de juego hay 3 líneas y una jungla. El objetivo del juego es derribar el Nexus enemigo.

Deberá construir un modelo para un problema de clasificación en el cual usará un set de datos del juego League of Legends y deberá usar como su variable objetivo si el equipo azul gana. Recuerden que:

- Deben hacer una breve exploración con los datos. Esto implica, pero no está limitado a:
  - Hacer encoding de las variables que se necesiten
  - Revisar si el dataset está balanceado, caso no estarlo, aplicar alguna técnica para balancearlo lo más y mejor posible
  - Escalar las variables si considera necesario
  - Selección de variables
- Recuerden hacer el split para training, testing y si consideran necesario para validation
  - 80% training
  - 20% testing
    - 10% validation si lo necesitan
- Recuerde definir de forma clara y razonada (es decir, diga el por qué de su elección) de una métrica de desempeño principal

Task 3.2 - Support Vector Machines: Clasificación de Partidas de League of Legends

Usando el dataset de este enlace, implemente desde cero un modelo de support vector machines para clasificar los mismos elementos que en el task dado. Para ello considere lo siguiente

- Divida el dataset en 80% para entrenamiento, 10% para validación (tuning) y 10% para test
- Recuerde que su variable objetivo es “blueWins”
- Provea una métrica de desempeño, justificando su elección
- Grafique los grupos encontrados
  - Puede usar solamente dos variables para mostrarlos en un plano cartesiano, similar a como lo hicieron en el laboratorio 1
- Mencione, como comentario que variables tuvieron que hacer tunning y cualquier otra consideración extra que tuvieron que tomar en cuenta

Para este task **no usen librerías**, sino implementen el algoritmo por ustedes mismos. Además, **evite el uso de herramientas de AI generativas (ChatGPT)**.

Luego:

- Repita los pasos para entrenar su modelo, pero ahora usando librerías y compare los resultados. **¿Cuál implementación fue mejor? ¿Por qué?** (Responda dentro del mismo Jupyter Notebook usando una celda tipo *Markdown*)

Task 3.2 - Árboles de Decisión: Clasificación de Partidas de League of Legends

Usando el dataset de este enlace, (es el mismo usado en el task 3.2) implemente un modelo de Árboles de Decisión para clasificar las partidas con el fin que pueda predecir si el equipo azul ganará. Para ello considere lo siguiente

- Divida el dataset en 80% para entrenamiento, 10% para validación (tuning) y 10% para test
- Recuerde que su variable objetivo es “blueWins”
- Provea una métrica de desempeño, justificando su elección
- ¿Qué métrica usaron para seleccionar los features?

- Especifique cuales son los *features* (columnas del dataset) que mayor importancia tomaron en la construcción del árbol (top 5)
- Si experimentan overfitting, ¿qué técnica usaron para minimizarlo?
- Mencione, como comentario que variables tuvieron que hacer tunning y cualquier otra consideración extra que tuvieron que tomar en cuenta

Para este task **no usen librerías**, sino implementen el algoritmo por ustedes mismos. Además, **evite el uso de herramientas de AI generativas (ChatGPT)**.

Pueden usar librerías para la lectura del archivo, métricas de desempeño, división de dataset, etc. Pero el algoritmo principal deben hacerlo ustedes.

Luego:

- Repita los pasos para entrenar su modelo, pero ahora usando librerías y compare los resultados. **¿Cuál implementación fue mejor? ¿Por qué?** (Responda dentro del mismo Jupyter Notebook usando una celda tipo *Markdown*)

Task 3.3 - Comparación

Al finalizar de los tasks 3.1 y 3.2, seleccione el modelo que mejor lo haya hecho en cada task (el implementado por ustedes o el de la librería), es decir, el task 3.1 tome un modelo y del 3.2 otro; con esto proceda a responder dentro de su Jupyter Notebook como una celda tipo Markdown:

- ¿Cómo difirieron los grupos creados por ambos modelos?
- ¿Cuál de los modelos fue más rápido?
- ¿Qué modelo usarían?

Nota: Para las partes donde se solicita que usen librerías, consideren usar Sckit-learn. Por favor, **asegúrese de leer la documentación** de estas librerías para entender mejor los hiperparametros. Siempre recuerde, si tiene alguna duda o consulta, por favor comuníquese con el catedrático.

Entregas en Canvas

1. Jupyter Notebook respondiendo a cada task.
    a. Incluyendo las preguntas del task 1
    b. Dentro del Jupyter Notebook deben colocar el link a su repositorio en GitHub. Este puede permanecer como privado hasta la fecha de entrega.

Evaluación

1. [0.5 pt] Task 1
    a. [0.1 pts] Pregunta 1
    b. [0.2 pts] Pregunta 2 y 3 (total 0.5pts)
2. [4.5 pts.] Task 2
    a. [1.25 pts] Cada modelo (Naive Bayes, SVM, Árbol de Decisión) implementado desde cero (total 3.75pts)
    b. [0.25 pts] Cada modelo implementado usando librerías (total 0.75pts)