TAREA 2: Aprendizaje supervisado

Profesor: Fernando Crema Garcia
Fecha: 26 de Agosto 2024

La nota final será basada el promedio de cada uno de los modelos ponderados por un coeficiente **extra** dependiendo de la calidad del reporte asociado.

R. Logística	Clasificación	Extra
RL	C	α

Fecha de entrega: 20/09/2024 23:59 pm Caracas.
Penalizaciones: 2 ptos / día. Máximo 10 ptos.
Nota final:
Para Clasificación

$$\text{mín}\left\{20\alpha\left(\frac{C}{\text{puntos totales}}\right), 20\right\}$$

Para Regresión Logística

$$\text{mín}\left\{20\alpha\left(\frac{RL}{\text{puntos totales}}\right), 20\right\}$$

I. Introducción

I-A. Objetivos

El objetivo de esta actividad es reforzar las habilidades adquiridas hasta el momento para aprendizaje supervisado específicamente para los casos:

1. Regresión Logística
2. Clasificación con k-vecinos.
3. Máquinas de soporte vectorial
4. Redes Neuronales

Siempre asumiendo que, para cada caso, aplicamos correctamente los conocimientos de preprocesamiento, selección de modelos y pipeline de proyectos en aprendizaje supervisado.

II. Evaluación

II-A. Reproducibilidad

Cada una de las secciones tiene la misma ponderación y al final será la suma de todos los ejercicios que lograron resolver ponderada por un criterio del grupo docente (α).

Para lograr que el grupo docente reproduzca sus resultados, deben asignar al comienzo de su entregable la semilla referente a su **cédula de identidad**

email: fernando.cremagarcia@kuleuven.be

II-B. Modelos entrenados

Sin importar la sección de la tarea que esté realizando, asuma que debe agregar un notebook **por sección** en el cual deben explicar toda la lógica de entrenamiento de los modelos desde preprocesamiento de datos, selección de variables, selección de hiper parámetros y evaluación de modelos. Se deja a juicio del estudiante agregar tanta información sea necesaria para justificar sus hallazgos (gráficos, tabla de resultados, etcétera) asumiendo que el Notebook va a ser corrido en un ambiente independiente [1].

De igual maneram, se deben respetar las siguientes restricciones:

1. Un Jupyter Notebook con uso adecuado de los ítems de headings (#) separando cada nivel de acuerdo al tipo de modelo y nombre del mismo. Esta vez, se deja a juicio de ustedes la configuración del mismo.
2. Asuman que el Notebook lo va a leer una persona del equipo de toma de decisiones de una empresa pero que tiene conocimiento previo en Aprendizaje Automático. Sean tan detallados como sea posible justificando cada decisión. **No hay soluciones únicas**.
3. Incentivo la creatividad de ustedes y formará parte de la evaluación de α
4. **No se pueden usar soluciones que provee Kaggle para los casos de Regresión Logística y Clasificación**. Nunca es un problema revisar soluciones de terceros. Sin embargo, su solución debe ser producto de **su** trabajo. Plagios serán penalizados con nota mínima.
5. Incentivo la colaboración como descrito en el **código de honor** de la materia. Sin embargo, copias serán penalizadas con nota mínima.

II-C. Aplicación de Streamlit (Clasificación)

En el caso específico de la parte de clasificación se espera el desarrollo de la aplicación en Streamlit siguiendo los pasos del repositorio https://github.com/ucvia/ml-tarea2B-knn-svm.

III. Clasificación

III-A. Objetivo

El objetivo principal de esta parte es finalizar la sección `A jugar` para que tengamos un panel como el siguiente:

en el cuál podamos ejecutar una operación matemática sencilla y evaluar su resultado.

[1] Esto significa que no puede depender de archivos externos, si tienen alguna duda de cómo lograr esto contacte al grupo docente

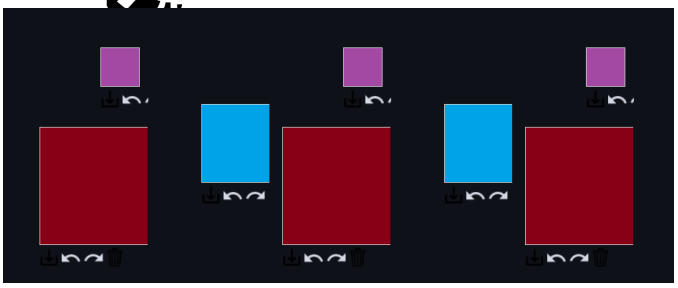

Figura 1. Canvas de imagen donde los cuadrados rojos (base) y magenta (exponentes) reciben números y los cuadrados color azul reciben Operadores. Se deben realizar en total 8 predicciones y luego ejecutar la operación $base_1^{exponente_1}\ operador_1\ base_2^{exponente_2}\ operador_2\ base_3^{exponente_3} =$ resultado

En el notebook asociado a esta sección debe comparar exhaustivamente **al menos** tres modelos vistos en clases [2]. Dependiendo de sus resultados, escoja y almacene uno de ellos [3] y úselo en la aplicación para ejecutar las operaciones.

La parte crítica de la tarea es la comparación de los modelos por lo que el uso de figuras y tablas es obligatorio [4]. Además, recuerde que existen para cada modelo distintas técnicas de regularización para evadir sobreajuste por lo que **deben ser consideradas en todos los modelos**

El canvas es la sección de nuestra aplicación de Streamlit que da uso al modelo escogido en la sección anterior.[5]

Tenemos entonces tres tipos de *input* en nuestro *canvas*:

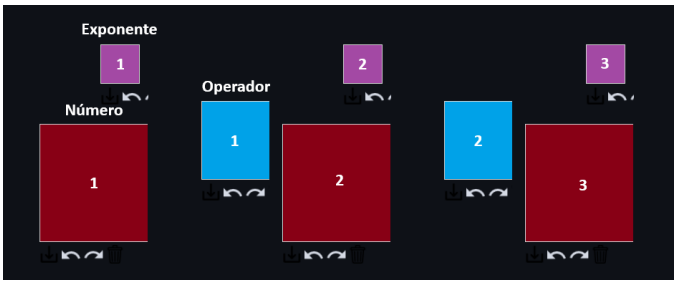

Figura 2. Canvas de imagen

1. Exponentes: 3 posibles referentes a los cuadrados morados. Deben ser números del 0 al 9.
2. Operadores: 2 posibles referentes a los cuadrados azules. Explicados en la siguiente sección.
3. Números: 3 posibles referentes a los cuadrados rojos. Deben ser números del 0 al 9.

[2]Entre ellos están k-vecinos, SVM, Redes Neuronales, Redes Neuronales Convolucionales, entre otros

[3]Si así desea puede cargar tantos modelos como quiera en la aplicación siempre que sea natural el proceso de selección dentro de la aplicación. Por comodidad, solo es necesario usar uno.

[4]Es recomendado dividir el trabajo al menos en tres secciones: metodología, resultados y discusión. Tanto para el preprocesamiento y transformación de los datos como para el entrenamiento de los modelos

[5]Si bien incentivamos el desarrollo de una aplicación usable, robusta, eficiente y eficaz recuerden que lo más importante es cubrir los objetivos asociados al aprendizaje automático. Esta sección es para divertirse y hacer una pequeña aplicación. La creatividad queda del lado de ustedes.

III-B. Modelo de operadores

III-B1. Datos de entrada: El dataset **debe** ser creado por ustedes! La cantidad de imágenes, estilo, resolución y cualquier elemento que consideren relevante debe ser decidido por ustedes: sean creativos. En el `notebook` referente al modelo de operadores, es necesaria una sección que explique toda la toma de decisiones.

III-B2. Clases del modelo: Solo vamos a usar las 4 operaciones fundamentales: suma, resta, multiplicación y división.

En el caso de suma y resta las únicas opciones posibles son: + (ASCII Code 43) y - (ASCII Code 45), respectivamente.

En el caso de multiplicación y división tendremos 2 opciones como sigue:

III-B2a. Multiplicación: Una × (ASCII Code 215) o un asterisco * (ASCII Code 42)

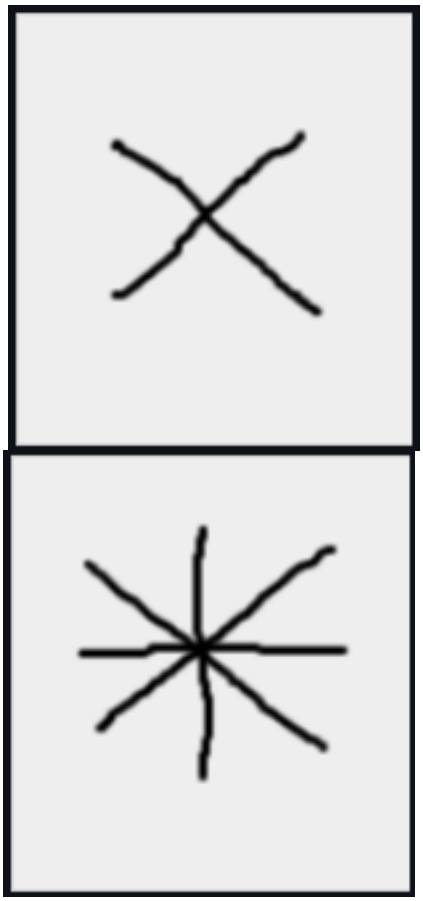

Figura 3. Multiplicación

III-B2b. División: Un `slash` / (ASCII Code 47) o el operando convencional ÷ (ASCII Code 247)

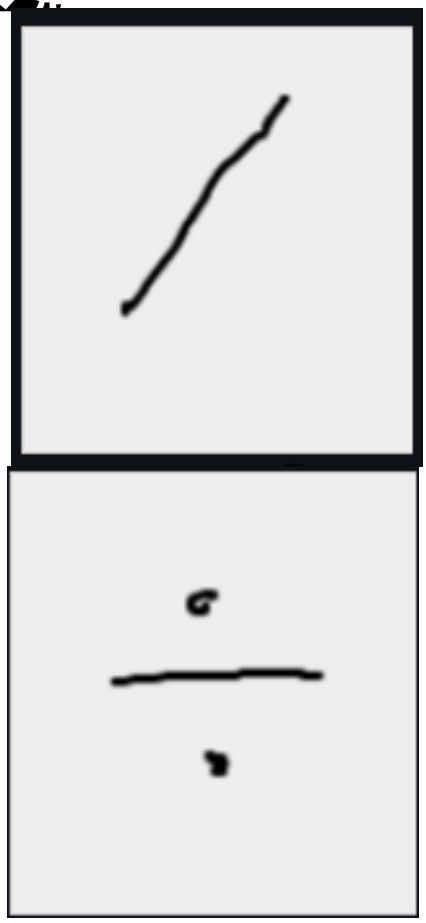

Figura 4. División

III-C. Modelo de clasificación de números

III-C1. Datos de entrada: Existen muchísimas opciones para leer los datos de MNIST. Por ejemplo, el siguiente código aparece en la aplicación del repositorio:

```
from keras.datasets import mnist

@st.cache_data
def get_mnist_data():
    return mnist.load_data()
```

Esta sección de código, usa el dataset mnist con la interfaz de Keras para facilitar consistencia en todos los proyectos.

En los notebooks del curso hemos usado el paquete torchvision en el caso de usar modelos de redes neuronales.

```
from torchvision.datasets import MNIST

train_dataset = MNIST(
  root='../data',
  train=True,
  transform=torchvision.transforms.ToTensor(),
  download=True
)
test_dataset = MNIST(
  root='../data',
  train=False,
  transform=torchvision.transforms.ToTensor(),
  download=True
)
```

Existen muchas maneras de usar el dataset y el diseño de su solución depende de ustedes. Sea lo más explícito posible en cada paso y decisión.

III-C2. Sobre la evaluación del pipeline:

1. En la sección `notebooks` deben crear uno o dos archivos `.ipynb` donde expliquen claramente todo el `pipeline` que usaron para crear los modelos, explicar el proceso de preprocesamiento, creación de datos (de ser necesario) y cualquier elemento que consideren relevante.
2. Los modelos asociados a la solución principal deben haber sido vistos en clase.
3. Puede usar modelos **extras** para comparar rendimiento aún cuando no hayan sido vistos en clases.

III-D. Comentarios

III-D1. Sobre el framework de la aplicación: El código base está hecho en Streamlit, un *framework* para desarrollar aplicaciones en Python enfocadas en datos.

La Documentación de *streamlit* es bastante sencilla de entender y la mayoría de funcionalidades necesarias ya están implementadas.

III-D2. Cómo ejecutar la aplicación: Necesitamos solamente los siguientes comandos de `Docker compose`:

- `docker compose build` crea el contenedor.
- `docker compose up` lo ejecuta en modo desarrollador.
- `docker compose up -d` lo ejecuta en modo `daemon`.

III-D3. Sobre las operaciones:

1. Asumimos que la aplicación siempre será usada por un agente honesto. No se debe validar para datos que no sean los referentes al modelo (aunque es un problema interesante de resolver).
2. Somos consistentes en la entrada de cada `canvas` así como en el orden de las operaciones: de izquierda a derecha y con prioridad de operadores: $\hat{}$, (*, /), (+, -).

III-D4. Sobre la parte visual: Escoger las secciones útiles de `02_Canvas.py` y crear la vista referente a cada uno de los elementos de entrada:

- 3 Coeficientes.
- 3 exponentes.
- 2 operadores.

Para luego llamar a los modelos y evaluar la función.

IV. Regresión Logística (Opcional sustitución de tarea)

La popular empresa de desarrollo de videojuegos Riot Games recientemente introdujo un modelo que calcula la probabilidad de que un **equipo profesional** gane en base a

situaciones similares de experiencias pasadas en su popular juego https://www.leagueoflegends.com/es-es. Una introducción sencilla al juego se puede ver en ¿Qué es League of Legends?

Citando su artículo sobre el tema dev: Probabilidad de Victoria con tecnología AWS: El porcentaje de probabilidad de victoria (PV) se obtiene a partir del número de equipos que se enfrentaron a una situación similar en el pasado y ganaron la partida. Nuestra estadística de PV se desarrolla utilizando un algoritmo de aprendizaje automático (AA) llamado xgboost, que tiene en cuenta muchos factores en su iteración actual.

IV-A. Variables usadas en este modelo

En su artículo, el equipo de datos de Riot explica cuáles son las variables usadas para su modelo:

1. Tiempo de la partida (tiempo dentro de la partida)
2. Porcentajes de oro (oro del jugador / oro total en la partida)
3. XP total de equipo
4. Número de jugadores con vida
5. Derribos de torretas
6. Asesinatos de dragón (si un equipo tiene un alma de dragón o no)
7. Baratija de Heraldo en el inventario
8. Temporizadores de inhibidor (cuánto tiempo tarda en reaparecer un inhibidor) para cada inhibidor
9. Temporizadores de Barón (tiempo hasta que expira la mejora de Barón del equipo)
10. Temporizador de Dragón Ancestral (tiempo hasta que expira la mejora del Dragón Ancestral del equipo)
11. Número de jugadores con el Barón activo
12. Número de jugadores con el Dragón Ancestral activo

Sin embargo, se han dado cuenta que el modelo funciona bien solo a nivel profesional y es complicado implementar una funcionalidad para su API de desarrollo por lo que XGBoost ha dejado de funcionar [6]

IV-B. Cómo se ve este modelo?

En la figura 5 podemos ver un ejemplo de la aplicación del modelo para explicar al usuario los cambios en probabilidad en el tiempo.

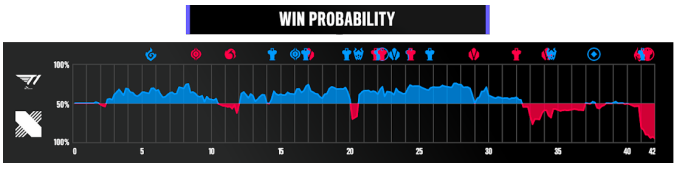

Figura 5. Obtenemos la probabilidad de victoria en el tiempo. Si la probabilidad de que el equipo azul es mayor que la del rojo, la gráfica tiene color azul y, en caso contrario, toma color rojo. En la parte superior de la gráfica hay íconos que se refieren a eventos de importancia que modifican radicalmente la probabilidad de victoria. Por ejemplo, asesinatos entre campeones, torres tiradas o dragones conseguidos.

[6] XGBoost sigue funcionando... pero lo usaremos luego =D

IV-C. Necesitamos tu ayuda

El equipo de datos de Riot nos ha pedido realizar un modelo cuyo output sea: **Probabilidad de que el equipo Azul Gane luego de 10 minutos**. Para hacer el problema más sencillo, usamos un conjunto de datos donde en el minuto 10 almacenamos variables similares a las que usa Riot.

Nuestro trabajo, además de conseguir el modelo, es explicar detalladamente cuáles variables seleccionamos y una breve explicación de la importancia en el modelo. Para ello, nos han pedido revisar la sección de Selección de variables y probar **dos** métodos que nos llamen la atención.

Además de ello, piden como entrega un Pipeline que involucre también el método selección de modelos justificado en su entrega.

Como tienen poco tiempo para revisar el informe, piden con ahínco que los resultados sean explicados con gráficas que puedan mostrar al equipo de negocios y decidir si seguir adelante con el proyecto (usando XGBoost) y datos nuevos [7].

IV-D. Los datos disponibles

Usaremos el dataset disponible en Kaggle Diamond ranked games after 10 minutes. Los datos están divididos en 3 secciones:

1. gameId
2. Variables del equipo azul con prefijo blue.
3. Variables del equipo rojo con prefijo red.

Para cada tipo red y blue tenemos las variables:

1. **Wins**: Ganó el equipo o no.
2. **WardsPlaced**: Número de centinelas puestos por el equipo.
3. **WardsDestroyed**: Número de centinelas destruídos por el equipo.
4. **FirstBlood**: Tuvieron primer asesinato o no.
5. **Kills**: Número de asesinatos.
6. **Deaths**: Número de muertes.
7. **Assists**: Número de asistencias.
8. **EliteMonsters**: Número de monstruos élite asesinados.
9. **Dragons**: Número de dragones asesinados.
10. **Heralds**: Número de heraldos asesinados.
11. **TowersDestroyed**: Torres destruídas.
12. **TotalGold**: Oro total.
13. **AvgLevel**: Average del nivel de las leyendas.
14. **TotalExperience**: Experiencia total.
15. **TotalMinionsKilled**: Número total de minions asesinados.
16. **TotalJungleMinionsKilled**: Número total de minions de la jungla asesinados.
17. **GoldDiff**: Diferencia de oro.
18. **ExperienceDiff**: Diferencia de experiencia.
19. **CSPerMin**: Número de minions asesinados por minuto.
20. **GoldPerMin**: Oro por minuto.

Una sección de preprocesamiento de datos es necesaria y pueden usar las gráficas que deseen siempre que expliquen detalladamente lo que hacen.

[7] Uno de los proyectos puede ser hacer esto!