Escuela de Informática
Universidad Nacional de Costa Rica (UNA)
Proyecto Scikitty
EIF420O Inteligencia Artificial I-2024. Dr. Carlos Loría-Sáenz

Especificación General de un Proyecto

Acrónimo de referencia rápido del proyecto para efectos del curso: **SciKitty**

Introducción

Esta especificación (`SPEC`) determina los elementos principales para la definición de un proyecto programado sobre inteligencia artificial en el marco y alcances del curso EIF420O-I-2024. El resultado final esperado es un paquete en Python scikitty, el que permite experimentar, en este caso, con árboles decisión que manejen variables categóricas en forma directa (sin una transformación numérica de las misma) en concordancia con los temas de AI cubiertos en el curso. (Nota "Jocosa": Categóricas a veces se abrevia a Cat, que en inglés es gato, como es un prototipo le diríamos gatito, o Kitty. La analogía con scikit nos lleva al nombre escogido).

Alcances

Se quiere alcanzar el desarrollo propio de una librería prototipo en Python que permita experimentar con algunos de los conceptos estudiados. Se entiende que los alcances están acotados por el nivel del curso y el tiempo disponible para el proyecto.

Marco de Referencia

Se han estudiado algoritmos clásicos para árboles de decisión (DT) inspirados en la línea de los bien conocidos ID3, C4.x y sucesores y CART, con énfasis en el manejo directo de variables categóricas, es decir, sin su transformación en variables numéricas. Se quiere experimentar con una implementación propia (casera) de este enfoque de clasificación y regresión por- simple interés académico, el que permita practicar con conceptos propios de árboles y más generales de ML supervisado, al nivel del curso.

Asimismo, se estudiarán enfoques basados en reglas de IA (simbólica) que permita "conectar" los conceptos sub-simbólicos de algoritmos de ML con conceptos de "mayor" nivel de representación (interpretabilidad, en particular). Se quiere que el prototipo permita ilustrar esa conexión.

Requerimientos Generales

Se debe desarrollar:

1. Un paquete `scikitty` que engloba toda la funcionalidad. Con al menos subpaquetes `models`, `metrics`, `persist`, `view`, `export` que organizan los principales módulos apropiadamente en forma comparable con `scikit`. Incluye un proceso de distribución estándar del paquete.

2. Una clase `scikitty.models.DecisionTree` (`RegressionTree`) que permite entrenar un modelo de clasificación (o regresión) que maneja tanto variables categóricas como contínuas (en un mismo modelo) sin codificar las primeras numéricamente. En el caso de clasificación puede ser multiclase, no solo binaria. Permite al menos entropía y Gini como criterios de impureza. Usa al menos `MSE` para el caso de regresión. Tiene hiperparámetros de control como altura máxima del árbol, mínimo de muestras para hacer split. Que permite estimar la probabilidad de una predicción. Permite calcular distintas métricas estándar de evaluación como la matriz de confusión, exactitud, precisión, F1, TPR, TFR. Permite generar una ROC. Permite evaluar si un dataset está balanceado en el target. Permite serializar (persistir) el modelo a un formato de persistencia (de escogencia libre, pero ver adelante la parte de Prolog). Permite recuperarlo para usarlo de nuevo sin reentrenarlo.

3. Una forma de visualización del árbol que permita su captura en un formato exportable.

4. Un compilador en Prolog que compile una instancia de un modelo de un DT salvado en el formato de persistencia usado por `scikitty`, y convertirlo en un programa `Prolog` ejecutable. A esto se le llama generación de reglas de decisión.

5. Las funcionalidades trabajan bien en los casos de prueba (demos) que se pidan.

Demos de Apoyo

En su debido momento se han entregado y se podrán entregar, por parte del profesor, scripts demostrativos que se explicarán, para como puntos de partida para el desarrollo de este proyecto, su comprensión y como casos a lograr.

Criterios de Evaluación

El profesor revisará a su criterio personal y profesional el trabajo tanto en forma como en fondo. Se usarán, en general, dos criterios: 1) funcionalidad en los demos (60%) en una demostración en la clase ante el profesor y 2) revisión offline (40%) por el profesor de los requisitos no funcionales. El incumplimiento en algunos requisitos no funcionales puede devaluar el proyecto parcial o hasta anular totalmente, según se indica abajo. Se publicará una guía de revisión con antelación.

Los criterios no funcionales de evaluación incluyen los señalados abajo. En su momento se dará una guía de puntajes específica con suficiente tiempo antes de la revisión final.

a) Puntualidad de la entrega pedida y esperada. Facilidad y agilidad de uso (puede anular el proyecto).
b) Cobertura de los objetivos planteados (el incumplimiento puede hasta anular el proyecto)
c) Uso de las herramientas pedidas a acordadas (el incumplimiento puede hasta anular el proyecto).

d) Cobertura y correctitud de las funcionalidades específicas pedidas (puede anular el proyecto).
e) Suficiente calidad del código. Incluye organización apropiada de funcionalidades y modularidad.
f) Respeto a terceros incluyendo una IA (puede anular el proyecto por copia/plagio)
g) Calidad de las funcionalidades requeridas
h) Entregable de fácil y estándar distribución y ejecución ágil para pruebas fuera del entorno de desarrollo (puede hasta anular el proyecto).

Herramientas y Estilos

- Lenguajes: `Anaconda Python, SWI-Prolog`
- Pueden usar en las partes no propias del proyecto librerías estándar (numpy, pandas, matplotlib o análogas, sckitlearn, scipy). Consulte si tiene dudas
- OOP y FP donde corresponda
- Semejanza en los APIs con los de librerías como scikit (fit, predict, etc).
- El uso de una IA debe ser limitado a soporte en uso de librerías y o demos de apoyo. Todo uso de una IA o de código de terceros en su código debe ser documentado y así expresado en forma explícita en el código entregado.
- Todo código está muy bien documentado, con comentarios y docstrings según estándares de Python.

Valor en la Nota

Este trabajo determina la nota dedicada a proyectos en la carta al estudiante con un valor de un 50% a ese rubro, es decir, un 20% de la nota total. Se permiten puntos extras en cada etapa, los que pueden hacer la nota mayor que 100 hasta un 25% extra. Solo aplica si se cumple con un 90% al menos de lo pedido como obligatorio. A criterio del profesor.

Igualmente, este trabajo podrá según su calidad ser usado más allá por el docente como un elemento de valoración y aprecio del interés por aprender y ganar el curso y así reflejarlo en el cálculo de la nota final del curso, como una adicionalidad.

Forma de Trabajo y Fechas

Se desarrolla **solamente en los grupos de trabajo formalmente establecidos en el curso** y conocidos por los estudiantes. Se harán sesiones de revisión presenciales. Durante las revisiones todo miembro del grupo debe estar presente y participar activamente y su **participación individual puede afectar la nota individual o grupal**. Las revisiones son en el horario de clase matriculado. Cualquier ausencia debe ser justificada como en una evaluación de examen.

Sobre fechas se plantea tentativamente lo siguiente: **semana 10**. Se notificarán la forma y hora de entrega, a definir al menos una semana antes por el profesor. Se podrán pedir avances que afecten la nota final del trabajo.

Para entregar el proyecto al profesor se abrirá un drive donde el coordinador de cada grupo subirá el entregable según se le pedirá a cada grupo. El entregable deberá ser razonablemente liviano, bien documentado en cuanto a uso y desarrollo, será tal que el profesor lo pueda usar sin demoras especiales en una máquina configurada como la usada en el Lab del curso (ambiente nn). Se puede asumir que Prolog está instalado en dicha máquina. Los requerimientos deberán estar explícitos en el proyecto.

El entregable es independiente de si usan o no un repositorio como GITHUB, que se recomienda hagan. Pero, en ningún caso el profesor hará "clone" para tener que revisar el proyecto. Los cuidados de seguridad de tal repositorio corren por cuenta del grupo respectivo.

Puntos Extra

Se pueden, a criterio del profesor, conceder puntos extra de hasta un máximo de un 20% adicional sobre la nota del trabajo, siempre que el trabajo cumpla los mínimos pedidos (85%). Los extras deben ser sugeridos por los estudiantes y aceptados de previo por el profesor. Una vez comprometido un extra se vuelve obligatorio. Se recomienda medir bien los riesgos de las decisiones de extras por las curvas de aprendizaje. Los extras, en ningún caso, podrán cambiar las herramientas obligatorias ni la arquitectura básica ni estilos de programación esperados.