Proyecto #1 - Predicción Votaciones

Inteligencia Artificial

Grupo 1, I Semestre 2018

El objetivo primario del proyecto será enfrentar a los estudiantes con una situación cercana a un proyecto de clasificación real donde existe una fuente de datos cruda, debe procesarse los mismos, compara algoritmos y reportar los resultados de una manera formal.

En particular, utilizaremos como base la biblioteca simuladora de votantes desarrollada por los estudiantes en el proyecto corto #1, con modificaciones mencionadas posteriormente.

El proyecto consistirá en un programa primario que podrá entrenar distintos modelos de clasificación de votantes y generará una serie de archivos de salida analizando el rendimiento de cada modelo.

Descripción general de entregables

A continuación se listan los elementos que se considerarán en los entregables para efectos de evaluación que los estudiantes deberán completar como parte del proyecto:

- Simulador de votantes aumentado con información de segunda ronda (10 puntos)
- Programa principal para iniciar entrenamiento de un modelo
    - Clasificación basada en modelos lineales (15 puntos)
    - Clasificación basada en redes neuronales (15 puntos)
    - Clasificación basada en árboles de decisión (25 puntos)
    - Clasificación basada en KNN con Kd-trees (25 puntos)
    - Clasificación basada en SVM (25 puntos opcionales)
    - Operación básica y detalles de ingeniería de software según especificación (5 puntos, si al menos un método es funcional)
- Informe
    - Manual de instalación y uso (5 puntos)
    - Reporte por cada método implementado (puntaje contenido en cada rubro anterior)
- Presentación de resultados a la clase (Proyecto corto #2)

Simulador aumentado

La primera tarea realizada tendrá que ver con modificar el simulador de datos ya creado. La interfaz de las funciones se mantendrá igual pero los estudiantes deberán retornar una lista de listas con una columna más: la etiqueta que representa por quién se votó en segunda ronda.

Se espera que el simulador sea llamado internamente por el programa principal del proyecto. Es decir, no será aceptable que primer haya que correr el simulador para generar conjuntos de datos y después llamar por separado al programa de clasificación.

Se espera que importar las funciones se pueda hacer de la misma manera:

```
from tec.ic.ia.pc1.g01 import generar_muestra_pais, generar_muestra_provincia
```

Modelos lineales

El programa principal deberá realizar el entrenamiento y evaluación de un modelo de regresión logística cuando se le pase el parámetro --regresion-logistica por la línea de comandos. Se espera que los estudiantes utilicen tensorflow directamente para realizar la creación del modelo.

Análisis de resultados: Deberán evaluar diferentes niveles de regularización (tanto L1 como L2) provisto por tensorflow a la hora de crear los modelos. Deberá exponerse dos banderas para especificar el valor por línea de comandos: --l1 y --l2 (L minúscula).

Redes neuronales

De manera similar, se deberá entrenar un modelo con redes neuronales. En este caso los estudiantes deberán utilizar keras para tales efectos, cuando se provea la bandera --red-neuronal.

Análisis de resultados: Evaluar diferentes estructuras (al menos 3) y funciones de activación (al menos 2). Exponer banderas llamadas --numero-capas, --unidades-por-capa y --funcion-activacion para configurar cada uno de los 3. Los estudiantes podrán documentar en el manual cuáles valores son válidos para la bander de funcion-activacion.

Árboles de decisión

Los estudiantes deberán implementar su propio árbol de decisión incluyendo un paso de poda. Para ello el programa recibirá la bandera --arbol indicando que este es el tipo de modelo a entrenar.

Análisis de resultados: Evaluar diferentes criterios de poda, utilizando la bandera --umbral-poda para especificar la ganancia de información mínima requerida para realizar una

partición. Es importante recordar que la poda se aplica hasta que el árbol completo haya sido construido.

KNN

La bandera --knn indicará que se debe entrenar un modelo de nearest neighbors. Se utilizará la bandera --k <numero> para indicar el número de vecinos a considerar. La estructura de datos interna a utilizar para agrupar los votantes deberá ser un kd-tree, implementado por los estudiantes.

Análisis de resultados: Evaluar diferentes valores de k

SVM

Como se mencionó anteriormente, este apartado es opcional. El objetivo es que los estudiantes utilicen alguna biblioteca para clasificación con SVM. Se sugiere utilizar scikit pero se deja a discreción de los estudiantes. Se señalará al programa con la bandera --svm. Dado que la escogencia de la biblioteca es abierta se le solicita a los estudiantes documentar en la sección de manual de usuario cuáles banderas adicionales deben agregarse.

Análisis de resultados: Al ser la escogencia abierta no hay un parámetro particular dado en esta sección, sin embargo, los estudiantes deberán escoger algún(os) parámetro(s) expuestos por la biblioteca seleccionada y generar análisis basado en ellos. Los parámetros seleccionados deberán ser expuestos como banderas.

Requerimientos básicos y operativos programa principal

La entrada del programa será una de las banderas mencionadas anteriormente, una bandera llamada --prefijo que se utilizará para poner al frente del nombre de todos los archivos necesarios durante una corrida y un tamaño de población a generar (--poblacion <numero>). Además, se utilizará una bandera llamada --porcentaje-pruebas <porcentaje>. Esto quiere decir que, por ejemplo, cuando tengamos --poblacion 10000 y --porcentaje-pruebas 20, 2000 votantes deberán reservarse para la validación final.

La salida por línea de comandos para cada modelo deberá ser el error de entrenamiento (utilizando cross validation) y pruebas (utilizando un set aparte). Además deberá generar un archivo de salida CSV, que contiene todos los datos originalmente generados por el simulador y cuatro columnas adicionales:

- es_entrenamiento: que deberá ser verdadera si para una corrida particular se utilizó en el proceso de cross validation
- prediccion_r1: predicción del partido político por el que se votó en primera ronda
- prediccion_r2: predicción del partido político por el que se votó en segunda ronda. No incluye la columna de voto real en primera ronda

- prediccion_r2_con_r1: predicción del partido político por el que se votó en segunda ronda. Si incluye el voto de primera ronda como atributo para entrenar el modelo

Todo el código deberá ser administrado en github.com en un repositorio privado. Nótese que existen cuentas gratuitas para estudiantes.

Informe

Los estudiantes deberán realizar un reporte técnico cuya audiencia será miembros de la escuela que NO están cursando Inteligencia Artificial (ya sea estudiantes o profesores). Deberán describir todos los aspectos técnicos de la realización (iniciando con el simulador de votantes) hasta un análisis de resultados.

Para cada uno de los modelos se espera que se haga un análisis de resultados comentando qué detalles se tomaron en cuenta para la configuración de cada modelo (e.g. cantidad de capas o unidades en redes neuronales), y las medidas de rendimiento detrás de cada modelo.

El puntaje del análisis para cada sección será parte del puntaje general del modelo. La descripción del simulador será contemplada en el puntaje del mismo

Además del análisis se pide generar un apéndice en el documento que sea un manual de usuario para instalar y correr el proyecto. Se puede asumir que la computadora posee Python instalado únicamente (e.g. debe explicarse cómo instalar tensorflow con pip).

El informe deberá realizarse utilizando markdown en un repositorio público de github.com, perteneciente a los estudiantes. Deberá enviarse una copia en PDF al profesor a través de TEC Digital.

Presentación de resultados

Una vez concluido el proyecto se calendarizarán presentaciones donde los estudiantes mostrarán sus resultados a la Escuela. Los detalles se darán posteriormente, sin embargo, es importante que a lo largo del proceso de este proyecto los estudiantes entiendan que su trabajo podrá ser analizado afuera de la audiencia del curso.

Detalles de Entrega y Revisión

El proyecto deberá realizarse en los mismos grupos creados para el Proyecto Corto #1. Se debe enviar todos los entregables a través de TEC Digital a más tardar el 4 de mayo del año en curso, antes de las 11:59PM.

El profesor se reserva el derecho a asignar una nota de cero si los requerimientos no son cumplidos de forma tal que impida ejecutar las funciones principales.

A manera de resumen los entregables serán:

- Código fuente con prefijo modular de Python tec.ic.ac.p1.g01. Este deberá estar almacenado en un repositorio privado de github. Se deberá enviar también un archivo comprimido a través de TEC digital.
- Los archivos CSV utilizados por el simulador de votantes.
- Pruebas unitarias. En particular deberá hacerse énfasis en las secciones implementadas por los estudiantes (KNN y árboles de decisión).
- Informe, en formato PDF (a TEC Digital) y Markdown en github.com

Consideraciones Python

- Se utilizará Python 3 (no 2.7)
- Se utilizará pip para instalar el módulo enviado. Se debe respetar la estructura de directorios apropiada para que la instalación sea exitosa.
- Se debe utilizar pytest para pruebas unitarias
- Seguir recomendaciones en http://docs.python-guide.org/en/latest/
- Seguir PEP 20. Una guía rápida con ejemplos está disponible en: http://artifex.org/~hblanks/talks/2011/pep20_by_example.pdf
- Utilizar PEP 8 como guía de estilo: http://pep8.org/