Proyecto #1 - Predicción con kaggle

Inteligencia Artificial

Grupo 1, II 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, comparar algoritmos y reportar los resultados de una manera formal.

Para ello se propone utilizar un conjunto de datos con muestras de tejidos posiblemente malignos (cáncer o tumor benigno) extraído de kaggle, un repositorio establecido para ciencia de los datos: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data. Adicionalmente, como opción para puntos extra, se propondrá una tarea de regresión basada en la predicción de calidad de diferentes productores de vino: https://www.kaggle.com/zynicide/wine-reviews.

En la primera el objetivo de predicción será si el cáncer es maligno o benigno mientras que, en el segundo, deberá predecirse el puntaje asignado a cada vino.

Se espera que el desarrollo de todo el proyecto no se enfoque solamente en los detalles de uso de modelos sino en un proceso de desarrollo de software de alta calidad, tomando en cuenta herramientas de desarrollo colaborativo (i.e. github.com) y desarrollo basado en pruebas apoyado en pytest y pytest-cov.

Además, se espera que durante el proyecto los estudiantes tengan que aportar un componente de investigación en el uso de las bibliotecas. En otras palabras, no todos los detalles de uso de las mismas se abordarán en clase (aunque si las bases teóricas). Esto con el afán de emular la situación real con que se enfrentarán después de graduación, donde es muy común tener que leer los detalles particulares de las bibliotecas para poder ejecutar un proyecto particular.

El proyecto consistirá en un programa de consola que, mediante opciones pasadas al mismo, podrá entrenar y evaluar los distintos modelos. Además 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 para efectos de evaluación que los estudiantes deberán completar como parte del proyecto:

- Pre-procesamiento de datos, normalización y codificación (10 puntos)
- Módulo de entrenamiento y predicción con random forests (30 puntos)
- Módulo de entrenamiento y predicción con redes neuronales (15 puntos)
- Módulo de cross validation y evaluación genérico (15 puntos)
- Programa de consola primario que llama a cada algoritmo (5 puntos)
- Informe
    - Manual de instalación y uso (5 puntos)
    - Reporte por cada método implementado con análisis en detalle, métricas de rendimiento, diferentes ajustes de modelo para mejorar rendimiento, etc. (20 puntos)
- Puntos extra
    - Módulo de entrenamiento y predicción con regresión lineal (20 puntos)
    - Incorporación a módulo de cross-validation, evaluación y programa principal (5 puntos)
    - Análisis de resultados detallados (10 puntos)

Requerimientos de alta calidad de desarollo de software

La evaluación de los ítems de código se enfocará fuertemente en las pruebas y diseño. Esto quiere decir para acceder a la mayoría de los puntos de cada sección, deberán existir pruebas que muestren que los requerimientos se cumplen. A manera de ejemplo, para probar que el código de entrenamiento para un árbol de decisión funciona, se esperaría que existan pruebas que creen una instancia de la clase entrenadora de árboles, se les pueda enviar una matriz con datos de ejemplo y retorne un árbol específico que puede ser probado con operaciones "expect" en pytest.

De la misma forma se espera que la cobertura de pruebas sea alta. Para ello se pide que los estudiantes generen un reporte de cobertura con pytest-cov (https://pypi.org/project/pytest-cov) y lo adjunten en el informe para cada módulo.

Finalmente, íntimamente relacionado con testing, se espera que los estudiantes hagan uso de lo que han aprendido sobre prácticos de diseño e implementación de código durante la carrera, de manera que el código sea modular, fácil de leer, adherido a estándares de estilo de Python y fácil de probar.

Algunos de estos requerimientos pueden considerarse fuera del ámbito de Inteligencia Artificial, sin embargo, se considera que este tipo de prácticas es fundamental para el desarrollo **correcto** de modelos. Si no podemos asegurar que el código no tenga errores, es difícil confiar en los resultados.

Pre-procesamiento de datos, normalización y codificación

Este primer módulo se refiere al código que puede leer los archivos de entrada y transformarlos en datos que se puedan enviar al módulo de entrenamiento o predicción. Se espera que las pruebas en este módulo tomen ejemplos de conjuntos de datos que transformen, según sea necesario, y se pruebe con pytest que se produce la transformación correcta para el modelo correspondiente.

Con respecto a normalización y codificación, los estudiantes deben tomar en cuenta dos técnicas particulares (que se deben investigar):

- Para atributos númericos, normalización por "z-score" o "standard score". Su esencia es para cada columna, calcular la media, restarla y dividir por la desviación estándar
- Para atributos categóricos, "one-hot encoding" que consiste en convertir una columna en múltiples columnas 1/0 para cada categoría en la original

Estos pasos son necesarios en cualquier proyecto de learning para que la magnitud y representación de los datos no vaya a ser la cause de resultados pobres.

Módulo de entrenamiento y predicción con random forests (30 puntos)

Los estudiantes deberán utilizar su propia implementación de árboles de decisión (como proyecto corto) y expandir dentro del esquema de random forests. Para estandarizar la posible implementación se debe seguir un método canónico descrito en la entrada de Wikipedia de random forests (en la sección "From bagging to random forests"): https://en.wikipedia.org/wiki/Random_forest

Ambos pasos deberán crear archivos de texto con reportes de estadísticas de rendimiento que puedan ser chequeados en la revisión presencial. Lo mismo aplica para redes neuronales y puntos opcionales, de realizarse

Como se mencionó los estudiantes implementarán 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.

Módulo de entrenamiento y predicción con redes neuronales (15 puntos)

Para esta sección no se le pedirá a los estudiantes implementar su propia red neuronal, sino que deberá utilizarse la infraestructura de tensorflow, una biblioteca de Machine Learning

desarrollada por Google, comúnmente usada en la comunidad. Se le permitirá a los estudiantes utilizar Keras, que es una capa de abstracción para redes neuronales, pero si se pide que tensorflow sea la implementación usada a bajo nivel. Se indicará con la bandera --red-neuronal.

Desde punto de vista de pruebas no se pide que se utilice pytest para probar funcionalidades de tensorflow (no tendría sentido), pero si se espera que los estudiantes tengan clases que abstraigan el entrenamiento y evaluación del modelo, potencialmente con algún tipo de herencia que se reutiliza entre Random Forests y Redes Neuronales.

Por lo tanto, las pruebas lo que deben centrarse es en asegurarse que los métodos de tensorflow sean llamados con los parámetros esperados. Se sugiere como lectura https://docs.python.org/3/library/unittest.mock-examples.html para que los estudiantes se familiaricen con "mocking". En particular, se recomienda revisar la función "assert_called_with".

Se insta a los estudiantes a tener discusiones con el profesor sobre estos aspectos muy importantes para el desarrollo de modelos correctos, pero que debieron ser cubiertos en cursos anteriores. Es esperable y natural que los estudiantes tengan dudas y las horas extra clase se pueden utilizar para ayudarles en este respecto.

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 bandera de funcion-activacion.

Módulo de cross validation y evaluación genérico (15 puntos)

Asumiendo que los módulos de entrenamiento y predicción fueron realizados correctamente, el entrenamiento total de los modelos deberán utilizar las ideas de cross-validation. Este módulo puede ser basado en el pseudocódigo disponible en el libro, sin embargo, se recuerda a los estudiantes que debe existir pruebas asociadas a TODO el código, incluído esto.

Además, la generación de archivos de resultados y salida en consola probablemente deba ser manejado por esta capa de software.

Por último, se espera que el diseño de este módulo sea suficientemente genérico para que reciba clases abstractas para cada uno de los 3 modelos a implementar.

Programa de consola primario que llama a cada algoritmo (5 puntos)

Se espera que la cantidad de código de este módulo sea poca y básicamente se encargue de leer parámetros con la biblioteca argparse y llame al módulo de cross validation y evaluación. De nuevo, deberá existir pruebas para la mayoría del código (exceptuando las pocas líneas de código que sean sólo del main).

Puntos extra

Para optar por los puntos extra los estudiantes deben realizar lo mismo descrito con anterioridad para árboles y redes neuronales, pero usando regresión lineal de scikit-learn. Todos los comentarios de las secciones anteriores aplican. Se deberá utilizar el dataset de vinos.

Se deberá pasar el parámetro --regresion-lineal por la línea de comandos.

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

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. Además, se utilizará una bandera llamada --porcentaje-pruebas <porcentaje>. El porcentaje restante deberá utilizar las funciones de predicción, únicamente, basado en el modelo entrenado.

La salida por línea de comandos para cada modelo deberá ser el error de entrenamiento (utilizando cross validation) y pruebas (utilizando el set aparte). Además deberá generar un archivo de salida CSV y una columna adicional para la predicción (favor agregar como última columna)

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 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.

Parte de la nota del informe se asignará basado en los diferentes intentos de los estudiantes por mejorar los resultados. Debe verse como un análogo a una pequeña defensa de proyecto de tesis donde se puede cuestionar el qué y el por qué de los parámetros, manipulación de datos, etc. El informe deberá proveer suficiente información para aclarar estos detalles.

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.

Distribución de nota y trabajo realizado. Los estudiantes deberán proveer, al final del informe, una descripción de las tareas de cada integrante así como el porcentaje de la nota que debe ser asignado a cada estudiantes. Por ejemplo, si cada estudiante contribuye por partes iguales, entonces deberán indicar que se asignará el 100% de la nota a cada uno. Sin embargo, si un miembro trabajó la mitad que los restantes, podrá indicarse que la distribución será 100, 100, 50, lo cual asignaría sólo la mitad de los puntos obtenidos al tercer miembro.

Detalles de Entrega y Revisión

El proyecto deberá realizarse en los mismos grupos creados para los proyectos cortos. Se debe enviar todos los entregables a través de TEC Digital a más tardar el 19 de octubre 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.

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/