Actividad 2

Datos del estudiante

- **Nombre:** Sergio
- **Apellidos:** Roselló Morell
- **DNI:** 53632974X
- **email:** Sergio-resello@hotmail.con

Información sobre el entorno

- **Sistema Operativo:** Arch Linux
- **Entorno de escritorio:** dwm
- **Versión de Python:** Python 3.8.2
- **Editor de texto:** NeoVim
- **Generación del documento:** Escrito en MD, compilado a LaTeX con

```
nnoremap <leader>e :! pandoc % -f markdown -t latex -s
-o %:r.pdf<cr>
```

Resumen

En esta actividad se van a replicar los cálculos de ejemplo del libro base con los datos del repositorio GitHub, para analizarlos y compararlos con los resultados mostrados en el libro. También se propone realizar un análisis similar con una modificación de los datos que elimine una característica importante de clasificación para ver en que varía la clasificación

Apartado 1.1 - Arboles de decisión - Cálculos con los datos de ejemplo

Se han realizado los ejercicios descritos en el libro paso a paso.

El Paso 1: Recolección de datos no difirió nada con el libro pero el paso 2: Extracción de "features" si. En mi caso, se han extraído 488114 "features" y 14 "labels".

Valores de parámetros se han usado En esta primera pasada, se ha decidido usar los parámetros ya establecidos por el framework, ya que el libro nos recomienda probar con estos primero y ajustar acorde resultados. Es por esto, que no se ha cambiado ningún valor del codigo.

Modificaciones de codigo No se ha modificado el código más allá de lo indicado en el enunciado de la práctica, para que tenga sentido y funcione con Python3.

Fallos o errores y solución Esta parte ha sido guiada, siguiendo el enunciado del libro. De esta forma, no he tenido ningún problema en seguir los pasos y no ha surgido ningún resultado inesperado.

Si que es cierto, que la curva ROC ha sido distinta, porque la mía nace en la posición `0,0` a diferencia de la de el libro, que directamente esta en `0, 0,8`. La curva ROC del libro indica que su model es mucho mas fiable porque no da prácticamente ningún falso positivo por cada verdadero positivo. En mi curva, hay menos sensibilidad, pero mas especificidad. Esto quiere decir que no es tan especifica a los casos calculados. La desventaja de esto es que puede marcar algunos servidores como malignos, cuando lo lo son realmente y viceversa.

La solución muestra el árbol de decisiones que se ha creado. En mi caso, el modelo permite determinar si una página es maliciosa (Servidor de Comando y Control) únicamente con 8 comprobaciones. Este resultado es el mismo que en el libro, aunque el árbol no sigue las mismas decisiones. Por ejemplo, las primeras dos comprobaciones si que siguen los mismos "endpoints", pero la segunda ya discrimina con valores de pertenencia menores. A partir de aquí, el árbol, aunque realiza en general el mismo numero de comprobaciones, lo hace de URL completamente distintas. Esto quiere decir que hay múltiples formas de determinar la pertenencia de las URL a un servidor maligno o no.

Las ventajas del árbol de decisiones son muy claras, hasta para una persona no técnica, ya que reducen el calculo para determinar si un servidor es maligno muchísimo. Únicamente comprobando como máximo 8 de las 488114 "features" podemos determinar si un servidor es maligno. Ademas, se puede seguir el flujo de las decisiones y ver como se ha hallado el resultado. Esto reduce nuestro coste computacional enormemente.

Apartado 1.2 - Regresión logística - Cálculos con los datos de ejemplo

En este apartado se procede a calcular el modelo de de regresión logística.

Valores de parámetros se han usado Al inicio, se ha usado los valores iniciales del algoritmo. Estos valores han generado un modelo bastante preciso. La curva ROC tiene indica un alto

Al cambiar el parámetro "L" de 2, que es el que hay por defecto a 1, la curva ROC nos indica que se ha perdido habilidad de clasificación.

Modificaciones de codigo En esta sección no se ha modificado codigo a parte del indicado por el enunciado de la practica.

Fallos o errores y solución A la hora de entrenar el modelo, se han hecho dos pasadas. La primera con los parámetros establecidos por defecto en el algoritmo de construcción del modelo. Esta pasada, ha generado un modelo muy preciso. Al realizar una segunda pasada, cambiando el valor `l2` por defecto al valor `l1`, el modelo no ha resultado ser tan preciso.

Podemos observar en la curva ROC del modelo `l2` como indica que se puede discernir entre un servidor malicioso o no con una alta probabilidad. En el valor FPR (False Positive Rate) 0.0, indica un valor 0.9 TPR (True Positive Rate) y a partir de ahí, solo hace que incrementar hasta el 100% TPR.

En la curva ROC del modelo `l1` vemos como la curva ROC no es tan precisa en todos los casos, pero es muy precisa en la mayoría de casos. Dependiendo de lo que estemos buscando, puede ser beneficioso usar el primer modelo o el segundo modelo mencionados.

Si queremos una seguridad mayor en la mayoría de los casos a coste de una baja certeza en algunos de los casos, debemos seleccionar el modelo `l1`, pero si lo que queremos es una certeza bastante aceptable en la mayoría de casos, debemos entonces seleccionar el modelo `l2`.

Como podemos ver en el resultado del siguiente comando y su resultado:

```
$python classify_panel.py bot_model.lrmdl https://bwall.github.io
Identifying panels we can actually reach
We can reach https://bwall.github.io/
Making 4789 total requests to 1 servers
https://bwall.github.io/, Not Panel, [9.99224387e-01 7.75612527e-04]
```

se hacen 4789 peticiones para determinar si la URL es valida o no. El resultado indica con mucha certeza que no es una URL maliciosa. Efectivamente, el modelo esta en lo cierto.

Cabe destacar el numero de comprobaciones que necesita el modelo de regresión logística en comparación con el modelo de árbol de decisiones, que para decidir si es un servidor malicioso o no, únicamente necesita 8 comprobaciones como máximo.

Apartado 2.1 - Árboles de decisión - Calculo eliminando una característica principal

Esta sección de la practica consiste en analizar el comportamiento de los algoritmos que generan los modelos de árbol de decisiones y de regresión logística habiéndoles quitado el rasgo mas determinante para discriminar entre tipos de servidores (Benignos o malignos)

Valores de parámetros se han usado En la primera pasada de la practica, se han dejado los valores por defecto de los dos algoritmos de generación de modelos.

En la segunda pasada, he reducido la profundidad del árbol a 5 niveles. He decidido esto porque he observado que en el árbol anterior, las siguientes capas tenían un "gini" cada vez mas bajo. Ademas de esto, he indicado que el algoritmo de ordenación sea "besa" en lugar de aleatorio.

Modificaciones de codigo En este apartado se ha preparado el entorno con los cambios sugeridos en el enunciado de la practica.

Por tanto, se ha añadido la cadena `config.php` a la lista `feature_blacklist` de la siguiente forma:

```
feature_blacklist = [
    "config.php",
    ".htaccess",
    #".gz",
]
```

De esta forma, se ignoraran los vectores con ese rasgo.

Fallos o errores y solución Al generar el modelo de árbol de decisión, se ve una deducción en los casos TPR. Esto quiere decir que el model no es tan especifico como el analizado con todos los "offsets". Este resultado no sorprende, ya que le hemos quitado el "offset" mas importante de decisión. Según que aplicación, sigue siendo un modelo valido.

Cuando he generado el segundo modelo, con las medidas indicadas en la sección anterior, han hecho que tenga una curva ROC bastante mejor que el anterior modelo. En esta ocasión, el modelo discrimina casi perfectamente entre tipos de servidores. A FPR `0.07` TPR `0.95` y desde ese punto, solo crece, hasta FPR `1` TPR `1`.

Apartado 2.2 - Regresión logística - Calculo eliminando una característica principal

Valores de parámetros se han usado Con el modelo nuevo, se ha probado cambiando "penalty" a `l1`. Los resultados de este modelo son bastante sorprendentes. Se analizan mas en detalle en la sección pertinente.

Ademas, se ha probado con la configuración por defecto, ("penalty" a `l2`) pero este modelo no ha dado resultados tan sorprendentes.

Modificaciones de codigo En este apartado se ha preparado el entorno con los cambios sugeridos en el enunciado de la practica.

Por tanto, se ha añadido la cadena `config.php` a la lista `feature_blacklist` de la siguiente forma:

```
feature_blacklist = [
    "config.php",
    ".htaccess",
    #".gz",
]
```

De esta forma, se ignoraran los vectores con ese rasgo.

Fallos o errores y solución Con la modificación del "penalty" a `l1`: Habiendo quitado el "offset" mas importante parece que ha sido beneficioso para ese modelo ya que tiene una curva ROC prácticamente perfecta. A FPR `0` TPR `0.95` que se mantiene estable hasta FPR `0.1` donde TPR sube hasta `1` y se mantiene a este nivel hasta el final. El valor de "penalty" `l1` determina la agresividad con la que se eliminan las "features" con poco valor predictivo. Dejando el valor por defecto, se ha conseguido la mejor tabla ROC.

En la configuración por defecto de regresión logística: La curva ROC, aunque aun adecuada, no es tan precisa como en la prueba anterior. En este caso, debe de ser debido al valor de "penalty", ya que es lo único que ha variado. El valor de "penalty" `l2` minimiza el impacto de un grupo de "features" altamente correlacionadas de forma que su influencia colectiva no varían los resultados.

Comentarios y opiniones

El análisis de los mismos datos con dos algoritmos de generación de modelos diferentes me ha hecho comprender mejor la forma en la que se comportan ambos. Cada uno tiene sus ventajas e inconvenientes. Ademas, la complejidad de los datos y distintas formas de generar el modelo requieren de un experto, no solo en Aprendizaje Automático, sino en los datos a analizar en quistión para que se genere el mejor modelo para cualquier caso.

Dificultades/Problemas encontradas La mayor de las dificultades ha sido pensar en la mejor forma de configurar los modelos, ya que es lo que requiere mas atención. Al final, aplicar las configuraciones y generar los modelos ha sido sencillo, gracias a los scripts proporcionados por el libro base de la asignatura.

Programas/Ayudas utilizadas Los programas que he usado para realizar estas tareas son:

- Python
- NeoVim y su capacidad de búsqueda y sustitución

Comentarios sobre la realización de la actividad Es una actividad interesante, para aplicar los conocimientos teóricos aprendidos en el libro base de la asignatura.

Bibliografía

- *scikit-learn:*
  - `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`
- *Logistic Regression:*
  - `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`