Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
Computación: Ciencia y Tecnología del Mundo Digital

Tarea Grande 2 - *Machine Learning*

IIC1005 - 1er semestre de 2018

Profesor Denis Parra

24 de abril de 2018

Indicaciones

- Fecha de entrega: 7 de mayo de 2018, 23:59 hrs.
- Deben entregar la tarea en el repositorio privado asignado para esta evaluación en la organización del curso en GitHub. No se aceptarán otros medios de entrega.
- Cada hora o fracción de atraso descuenta 0,5 puntos de la nota que obtengas. Se considera como entrega el último *commit* presente en el repositorio, es decir, la hora en la que el este se hace presente en GitHub, no la hora de creación del *commit*. No se revisarán *commits* anteriores.
- La tarea es **en parejas**. La copia será evaluada con nota 1 en la tarea, además de las sanciones disciplinarias correspondientes. Esta tarea está regida por el Código de Honor de la Escuela.

Objetivos

1. Enfrentar un desafío real en el contexto de aprendizaje automático (*machine learning*).
2. Preprocesar un set de datos para entrenar un modelo.
3. Analizar un set de datos para elegir las mejores características para solucionar un problema.
4. Solucionar un problema mediante el uso de del paquete `scikit-learn` de Python.
5. Entender la importancia de que un modelo se equivoque o sea acertado.
6. Medir el error de un modelo de clasificación.

Introducción

En los últimos años hemos observado a la inteligencia artificial y la minería de datos revolucionando tanto el mundo de la investigación como de la industria. La gran cantidad de datos disponibles hacen posible entrenar modelos que automaticen procesos de toma de decisiones en los cuales sólo hacen un par de años se creía siempre necesaria contar con un ser humano, como conducir vehículos, revisar cámaras de seguridad o hacer traducción automática.

Una de las tareas tradicionales de machine learning es la *clasificación* automática. Puede abarcar esta tarea la clasificación de imágenes, de textos, de personas o hasta de estrellas en el espacio. La clasificación de objetos es uno de los grandes desafíos que los modelos de *machine learning* intentan resolver. Clasificación automática se refiere a un sistema que es capaz de identificar para cada objeto la clase a la cual pertenece. Por ejemplo, un robot que ha sido entrenado con una base de datos de fotos de desechos, donde cada foto cuenta con una etiqueta de si lo que se muestra es o no reciclable, podrá clasificar basura en reciclable o no reciclable automáticamente.

Su misión

Su equipo ha sido contratado por la compañía privada **PUC Analytica** para analizar unos datos comprados a **Fakelook**, los que son fruto del análisis realizado por personal de Fakelook de distintas páginas web que han sido acusadas de *phishing*. Esta base de datos describe algunas características de la página indicada, junto con el veredicto del humano que la revisó indicando si la página es legítima, sospechosa o *phishy*. Si bien Fakelook ha mejorado sus filtros de privacidad (la página analizada no puede inferirse a partir de los datos entregados) no ha mejorado mucho en el formato de los datos, por lo que tendrán que hacer preprocesamiento de estos.

La misión de su equipo es entrenar el mejor modelo de clasificación para este problema, experimentando con distintas parametrizaciones de los siguientes modelos de clasificación:

1. Árbol de decisión.
2. K vecinos más cercanos.
3. Regresión logística.

La pregunta a responder es: ¿esta página es legítima, sospechosa o *phishy*? El fin de este clasificador es crear una herramienta que advierta a los usuarios no tan expertos cuando estén en riesgo de ser estafados.

	id	URL_of_Anchor	Result	Query_date	having_IP_Address	valid_IP_Address	Request_URL	URL_Length	Icon	age_of_domain	SSLfinal_State	SFH	web_traffic	popUpWindow
2	0	-1	-1	18-07-2013	No	0	1	0		1	0	1	0	-1
3	1	0	-1	28-07-2013	Yes	1	1	0		1	1	1	0	0
4	2	-1	-1	16-06-2013	No	0	0	0		-1	1	1	1	0
5	3	1	-1	20-02-2014	No	0	-1	1		1	1	1	-1	1
6	4	1	-1	06-02-2014	No	0	-1	-1		-1	-1	1	1	0

Figura 1: Primeras filas de la base de datos entregada.

1. Los datos (1 punto)

Su equipo recibe una base de datos donde cada fila es la evaluación de una página por parte de uno de los empleados de Fakelook junto con su veredicto. En la Figura 1 pueden verse las primeras filas del archivo entregado. En el anexo se entrega una descripción del significado de cada variable.

1.1. Preprocesamiento de datos

Para sacar el máximo provecho a la información disponible, es necesario transformar los datos. Además, cada modelo se entrena con un tipo distinto de datos, por lo que debes asegurarte de que los datos que se le muestran a cada uno estén en el formato correcto.

Como se indicó antes, Fakelook no es muy bueno con el orden de los datos, por lo que notará que la columna a predecir (`Result`) ni siquiera es la última. Además, es posible que haya columnas donde no exista información relevante.

2. Modelos (2 puntos)

2.1. Árbol de decisión

Estructura de árbol formada por nodos que representan reglas de decisión.

2.2. K vecinos más próximos (K-NN)

Método que sirve para clasificar una instancia según la clase más popular entre sus vecinos más similares.

2.3. Regresión Logística

Tipo de clasificador basado en regresión para predecir variables categóricas.

3. Evaluación de clasificadores (1 punto)

Se te entregan los datos en un archivo llamado `data.csv`. Es su responsabilidad decidir la mejor forma de dividir estos datos para poder entrenar y testear sus modelos. La sugerencia desde PUC Analytica es usar

una división aleatoria de 70 %/30 % entre entrenamiento y *testing*.

Para entrenar el modelo solo deben utilizar una parte de la base de datos de entrenamiento, el resto deben dejarlo para testear qué tan bien clasifican los modelos. Para llegar a una mejor estimación del error futuro de tu modelo, se recomienda utilizar **validación cruzada** con los datos del set de entrenamiento para predicción del error.

4. Informe (2 puntos)

Para comprobar que la realización de su tarea fue a conciencia y no una mera importación de código, se exige un informe que responda lo siguiente, de manera detallada:

1. ¿Cómo fue el preprocesamiento de los datos? ¿Cuáles fueron los pasos a seguir?
2. ¿Cuál de los modelos probados en esta tarea es mejor? ¿Por qué? Incluyan métricas que permitan validar su respuesta.
3. Describa los parámetros más importantes que debió validar en cada modelo (por ejemplo, el número de vecinos k para el modelo k-NN)
4. ¿Cuál es la matriz de confusión de cada modelo? Incluye imágenes de cada matriz ¿Qué significa cada celda?
5. Para realizar este informe, seguramente se toparon con las métricas *precision* y *recall*, ¿cuál de las dos entrega más información sobre el desempeño de un modelo? ¿Existe alguna métrica que agrupe a las dos antes mencionadas?

Cuiden la **ortografía, redacción y presentación** de su informe, pues estos ítems también tienen puntaje. Es deseable que fundamenten sus respuestas con datos y gráficos apropiados.

5. Entrega

Para esta tarea, **deben trabajar en parejas**. En particular, se espera que entren a este link y ejecuten el siguiente flujo: un miembro de la pareja debe ingresar primero y bautizar al equipo con un nombre ojalá creativo y no ofensivo ni vulgar. A continuación, el otro integrante del equipo debe entrar al mismo enlace y buscar el repositorio creado por su compañero. **Solo se aceptarán entregas en el repositorio creado para el equipo en la organización del curso en GitHub** (no *forks*, no SIDING, no email).

El plazo para realizar esto vence el **29 de abril**. Posteriormente, el equipo docente designará un compañero **de forma arbitraria** a todos quienes no hayan inscrito un grupo. Se creará una *issue* en GitHub para que puedan completar su equipo si no tienen con quien formarlo.

6. Entregables

Para el día y hora de la entrega, en su repositorio debe encontrarse un **Jupyter Notebook** (archivo de formato .ipynb) que cumpla con lo siguiente:

1. **Informe**: debe responder a todas las preguntas solicitadas en la sección informe. El jupyter notebook entregado debe partir con esta sección, donde se responden las preguntas usando *markdown*. A continuación debe, venir el código.

2. **Código**: las tareas cuyo código sea difícil de entender, tendrán un descuento de máximo 0.5 décimas, a criterio del corrector. Además, el código debe ser autocontenido, es decir, si se reinicia el notebook y se vuelve a ejecutar, el *output* de las celdas debe ser el esperado: debe tener los resultados que permiten responder las preguntas del informe.

A modo de aclaración, la tarea contempla los siguientes descuentos posibles: descuento por atraso (ver indicaciones), descuento por no entregar la tarea de forma correcta (ver indicaciones), descuento por inconsistencia entre respuestas del informe y el *output* del código, y descuento por baja calidad del código. Todos los descuentos, salvo el último, pueden llegar a descontar el puntaje completo de la tarea. **Estos descuentos no serán reconsiderados en la eventual recorrección de la tarea**.

7. Bonus

Si quieren aumentar su nota, los invitamos a participar en nuestra propia versión de Los Juegos del Hambre. La competición consiste en premiar a los 5 grupos que logren perder la menor cantidad de dinero, de acuerdo a lo siguiente:

- Si se clasifica un sitio como "legítimo" cuando en realidad es "*phishy*", se pierden $5.000.000.
- Si se clasifica un sitio como "*phishy*" cuando en realidad es "legítimo", se pierden $5.000.000.
- Si decide que una página es "sospechosa" cuando en realidad existe evidencia para determinar que es "legítima" o "*phishy*", se pierden $7.000.000.

El premio consistirá de hasta 5 décimas según la siguiente tabla de premios:

Posición	Décimas
1^{er} lugar	5
2^{do} lugar	4
3^{er} lugar	3
4^{to} lugar	2
5^{to} lugar	1

Cuadro 1: Décimas según *ranking*

Para esta competencia se entregará el dataset `data-bonus.csv`, donde se entregan las descripciones de otras páginas, pero sin entregar la clase esperada. Usted debe entregar sus predicciones para estas páginas en un archivo llamado *prediccion-bonus.csv* que debe ser generado por el archivo **bonus.ipynb**. Su respuesta solo será válida si dicho *notebook* es autocontenido y es capaz de generar el mismo archivo de predicciones entregado.

Para comprobar el desempeño de sus grupos, se habilitará en GitHub una *issue* donde podrán comparar sus resultados con los demás interesados. Como queremos una competencia honesta, para determinar a los 5 ganadores, probaremos sus modelos con un *dataset* desconocido para ustedes. En ese sentido, es conveniente ser **honestos** en la publicación de sus resultados. Cualquier irregularidad detectada, provocará la **descalificación** de su grupo de esta competencia.

8. Anexos

Información atributos en *dataset*.

- `id`: Número identificador de la página.
- `URL_of_Anchor`: Variable binaria. Indica si los *links* en la página apuntan a un dominio distinto del tipeado en la *address bar*.
- `Result`: Clase a predecir. Indica si un sitio es "*Legitimate*" (1), " *Suspicious*" (0) o "*Phishy*" (-1).
- `Query date`: Fecha en que se realizó la evaluación del sitio.
- `having_IP_Address`: Variable cualitativa. Indica si el dominio contiene una dirección IP.
- `valid_IP_Adrress`: Variable binaria. Indica si la dirección IP del dominio es válida.
- `Request_URL`: Variable ternaria. Indica si un alto porcentaje de los *links* son de otros dominios (1), si una cantidad aceptable lo es (0) o si muy pocos lo son (-1)
- `URL_Length`: Variable ternaria. Indica si una URL tiene tiene pocos caracteres (1), una cantidad aceptable (0) o muchos caracteres (-1) en la URL.
- `Icon`: Variable cualitativa. Indica URL al ícono de la página, si es que fue considerado en el análisis.
- `age_of_domain`: Variable binaria. Indica si el sitio tiene presencia online hace menos de 6 meses (1) o más (-1).
- `SSLfinal_State`: Variable ternaria. Describe si la página cumple con los protocolos correspondientes: si usa el protocolo HTTPS y tiene certificados por más de 2 años (1), si usa HTTPS pero los certificados no son de confianza (0) y cualquier otro caso (-1).
- `SFH`: Variable ternaria. Describe cómo la página responde al enviar un form: si el servidor receptor de los forms en la página entrega una respuesta vacía (-1), entrega una redirección a otro dominio (0) o cualquier otro caso (1)
- `web_traffic`: Variable ternaria. Describe el tráfico recibido por la página en términos de su *ranking* en la "Alexa database": mejor que 150.000 (1), peor que 150.000 (0) o no aparece (-1).
- `popUpWindow`: Variable ternaria. Si el evento "click derecho" está desactivado (-1), si el mismo evento responde con un "alert()" (0) o cualquier otro caso (1).