

Aplicaciones de ciencia de datos para la detección de *phishing*

José Vicente Mellado

Introducción

El phishing es un tipo de ataque informático que consiste en hacer pasar un sitio Web malicioso por uno legítimo con el objetivo de conseguir información personal del usuario.

A pesar de ser longevo, sigue estando presente en nuestro día a día.

Aunque han surgido nuevas maneras de phishing (ej: aplicaciones móviles), su implementación mediante tecnologías Web sigue siendo la más importante.

Introducción

En este trabajo se estudia la aplicación de técnicas de ciencia de datos para la detección de phishing en sitios Web.

También se explorarán soluciones de machine learning sobre Spark.

Se ha utilizado un ciclo de desarrollo típico en un proyecto de ciencia de datos llamado CRISP-DM.

Entendimiento del problema

Lectura de informe de seguridad informática Symantec, así como el libro “Data Mining and Machine Learning in Cybersecurity”.

También se exploró de manera descriptiva el dataset Sherlock:

- Reducción del consumo de memoria y CPU, y aumento del tráfico en red cuando se produce un ataque.

Entendimiento de los datos

Lectura de documentación así como papers relacionados.

El conocimiento previo en desarrollo Web también ayudó al entendimiento de las variables.

Se realizó un análisis exploratorio donde se encontraron que algunas variables eran muy discriminatorias:

- Guión en la URL, si tiene subdominios, edad del dominio, entre otras cuatro variables.

Preparación de los datos

Los variables tenían valores -1, 0 y 1 por defecto, aunque resultó útil para el análisis exploratorio asignar etiquetas legibles a cada valor.

División de los datos entre conjunto de entrenamiento y test.

Rebalanceo de clases: varias técnicas (undersampling).

- Rebalanceo aleatorio simple
- Condensed Nearest Neighbor
- One-Sided Selection

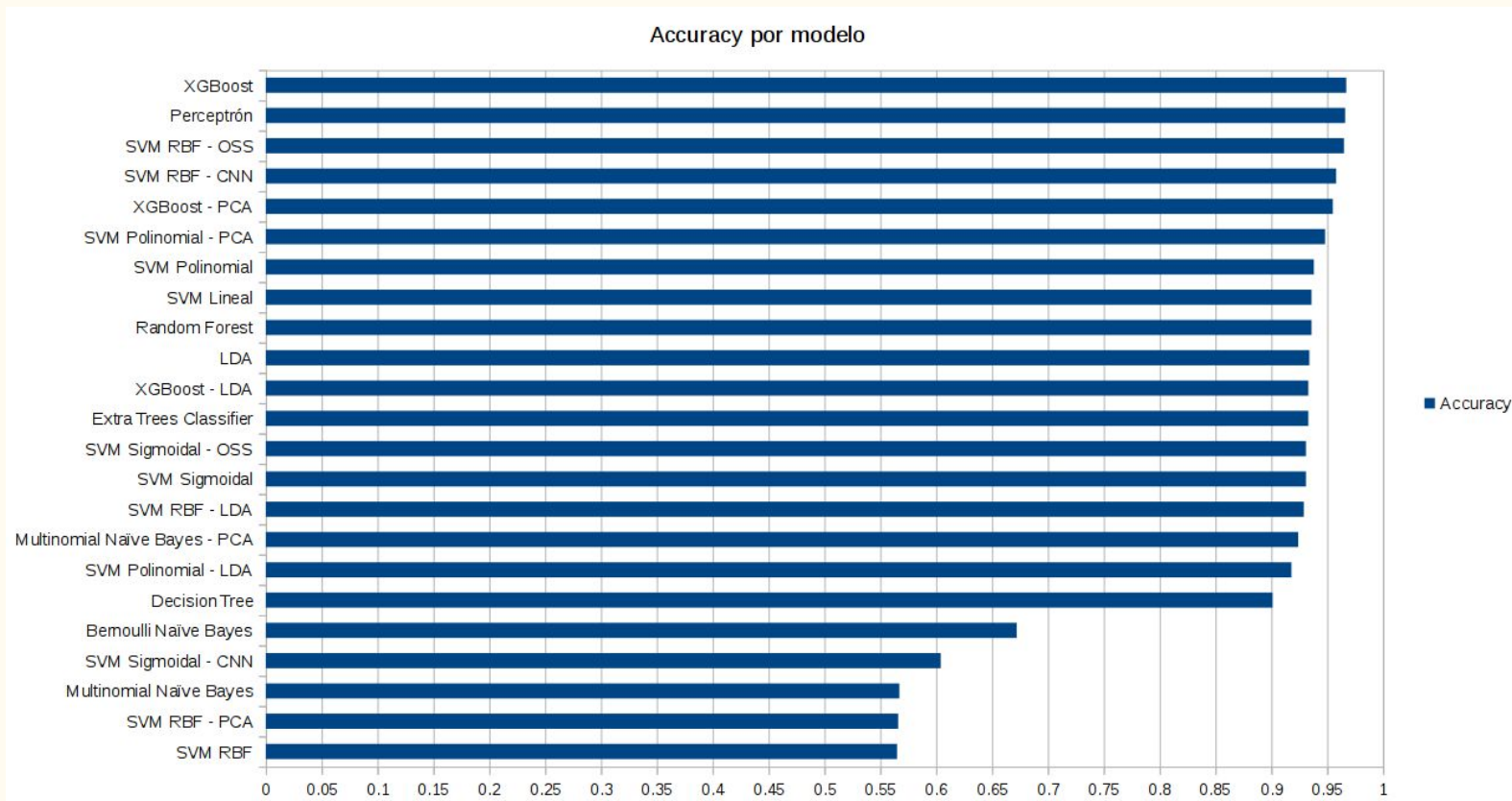
Modelado

Se realizaron multitud de modelos, tanto los recomendados por la literatura como los vistos en clase.

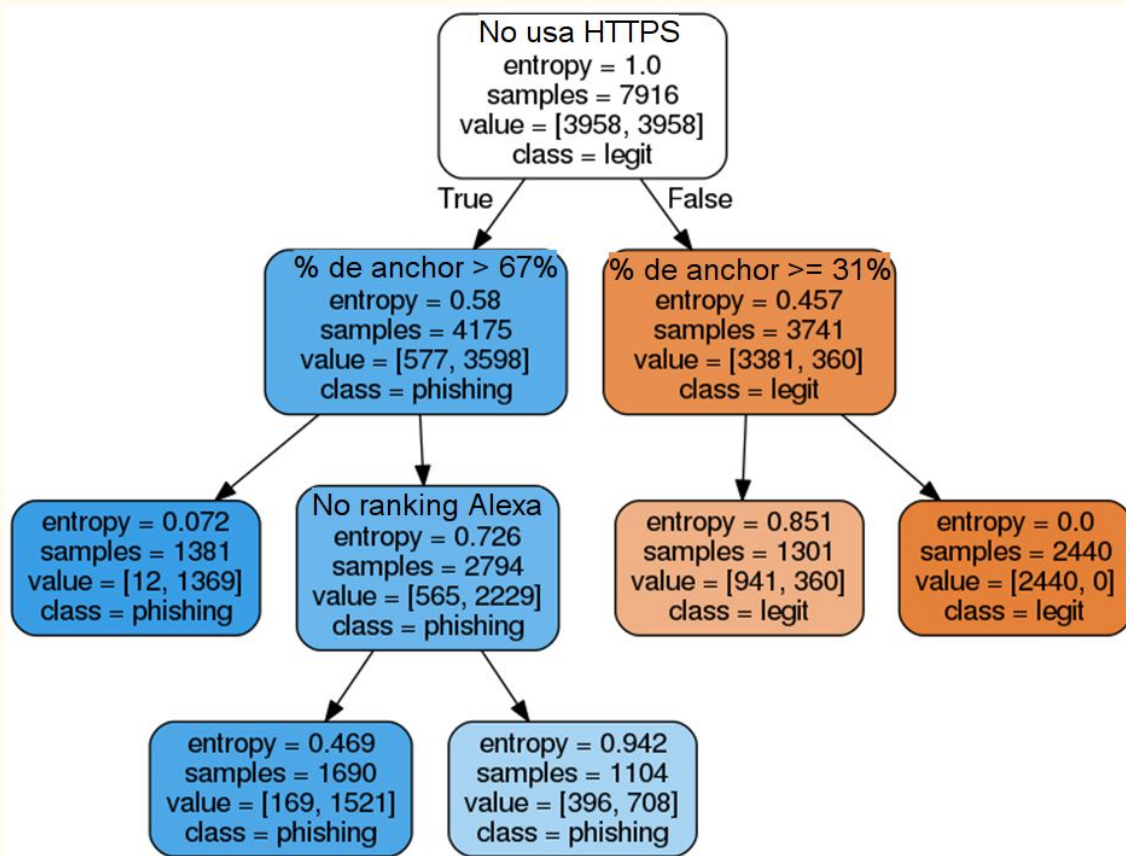
En todos los casos se usó Cross Validation (5-fold) y Grid Search para la selección de hiperparámetros.

También se intentaron aplicar técnicas de reducción de la dimensionalidad como PCA y LDA.

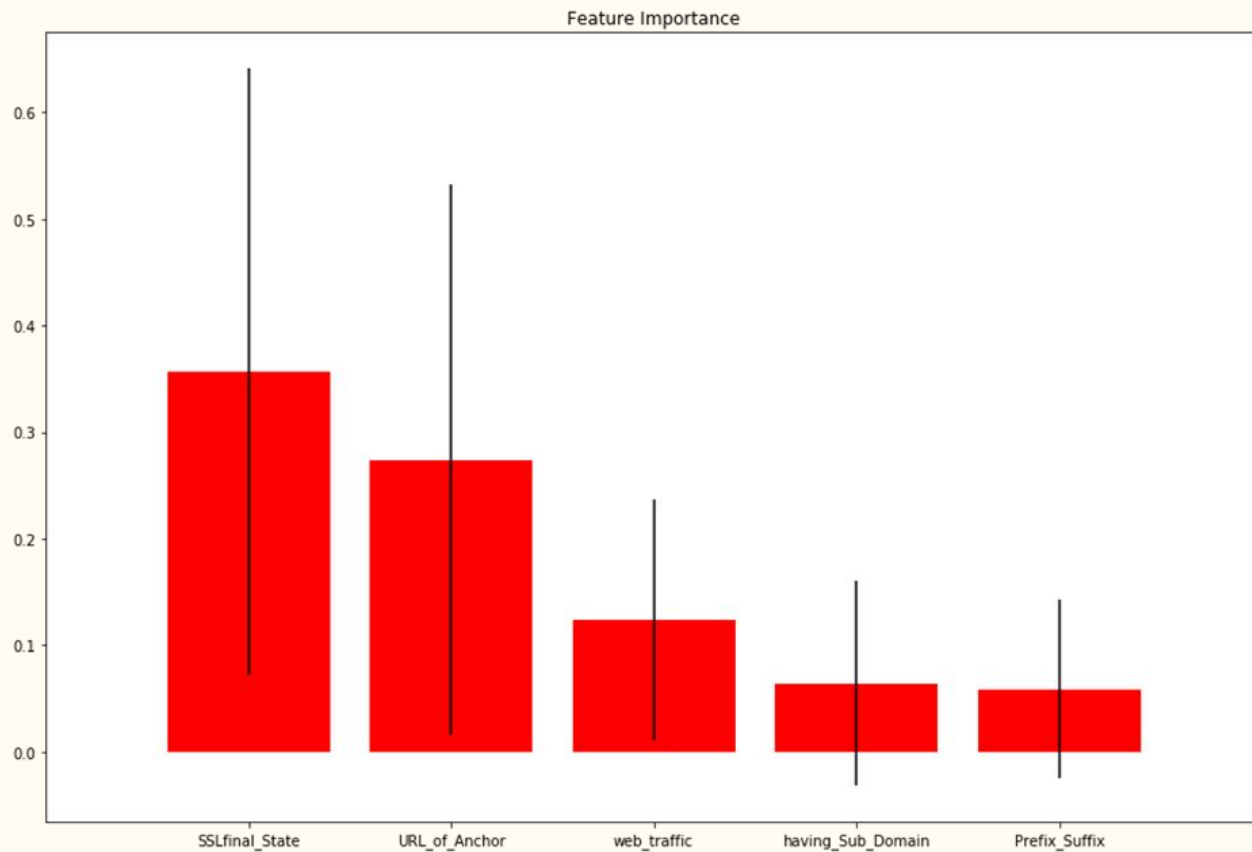
Evaluación



Evaluación



Evaluación



Evaluación

Algunas conclusiones:

- Los métodos de rebalanceo especiales suponen una mejora en algunos casos, además de acelerar el entrenamiento de SVM.
- Los modelos de Naïve Bayes no suelen funcionar, seguramente porque no se cumplen las premisas en las que se basan.
- Perceptrón ajustable casi tanto como se desee.
- XGBoost el mejor modelo pero el más costoso de entrenar.
- Las variables más importantes en RF coinciden con algunas de las más discriminatorias en el análisis exploratorio.

Evaluación: Caracterización de sitios phishing

Tipo 1 de Website con phishing:

- Request_URL: requestUrl%_lt_22
- Links_in_tags: links%_gt_81
- popUpWindow: other
- web_traffic: alexa_gt_100K

Tipo 2 de Website con phishing:

- Favicon: external_domain
- port: preferred_status
- Links_in_tags: links%_gte_17_lte_81
- Submitting_to_email: mail_or_mailto
- on_mouseover: statusbar_change
- popUpWindow: yes&with_form
- web_traffic: alexa_gt_100K

Tipo 3 de Website con phishing:

- Domain_registration_length: gt_1_year
- Links_in_tags: links%_gt_81
- popUpWindow: other
- web_traffic: not_in_alexa

Tipo 4 de Website con phishing:


- having_IP_Address: true
- Shortining_Serice: true
- double_slash_redirecting: true
- HTTPS_token: true
- Links_in_tags: links%_lt_17
- Abnormal_URL: no_hostname_in_url
- Redirect: 0_or_1
- popUpWindow: yes&with_form
- age_of_domain: gte_6_months
- DNSRecord: not_found
- web_traffic: alexa_lt_100K
- Links_pointing_to_page: gt_2

Despliegue

Uso de Spark con la librería de *machine learning* H2O

Se realizaron dos modelos costosos de entrenar: Gradient Boosting y Random Forest

Despliegue

 2.0.2

Jobs


Stages

Storage

Environment

Executors

SQL

Sparkling Water 

PySparkShell application UI

Spark Jobs (?)

User: root
Total Uptime: 1.7 h
Scheduling Mode: FIFO
Completed Jobs: 7

[Event Timeline](#)

Completed Jobs (7)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	runJob at WriteConverterCtxUtils.scala:85	2017/09/10 14:12:19	0.2 s	1/1	4/4
5	count at NativeMethodAccessorImpl.java:-2	2017/09/10 14:12:19	0.3 s	2/2	5/5
4	runJob at WriteConverterCtxUtils.scala:85	2017/09/10 14:10:19	0.6 s	1/1	4/4
3	count at NativeMethodAccessorImpl.java:-2	2017/09/10 14:10:15	3 s	2/2	5/5
2	foreach at InternalBackendUtils.scala:175	2017/09/10 14:07:49	51 ms	1/1	2/2
1	collect at InternalBackendUtils.scala:163	2017/09/10 14:07:49	0.8 s	1/1	2/2
0	collect at SpreadRDDBuilder.scala:105	2017/09/10 14:07:43	5 s	1/1	21/21

Despliegue



Spark Master at spark://ip-172-31-15-58.eu-west-1.compute.internal:7077

URL: spark://ip-172-31-15-58.eu-west-1.compute.internal:7077

REST URL: spark://ip-172-31-15-58.eu-west-1.compute.internal:6066 (cluster mode)

Alive Workers: 2

Cores in use: 4 Total, 4 Used

Memory in use: 5.2 GB Total, 4.7 GB Used

Applications: 1 [Running](#), 9 [Completed](#)

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20170909114445-172.31.1.83-57562	172.31.1.83:57562	ALIVE	2 (2 Used)	2.6 GB (2.3 GB Used)
worker-20170909114445-172.31.5.233-49908	172.31.5.233:49908	ALIVE	2 (2 Used)	2.6 GB (2.3 GB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170910140638-0009	(kill) PySparkShell	4	2.3 GB	2017/09/10 14:06:38	root	RUNNING	57 min

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170910140003-0008	Spark shell	4	2.3 GB	2017/09/10 14:00:03	root	FINISHED	22 s

Despliegue

El uso de tecnologías para el cómputo distribuido y librerías preparadas para ello es sin duda una ventaja muy importante, reduciendo el tiempo de entrenamiento de los modelos ejecutados considerablemente.

Líneas de trabajo futuras

En cuanto a realización de un proyecto real:

- Realización de un sitio Web que se encargue de realizar un listado de páginas Web maliciosas.
- Implementación en los navegadores de Internet de un sistema que detecte si el sitio que se está visitando es phishing o no y en caso afirmativo se bloquee la navegación y se notifique al usuario de que el sitio que está visitando es peligroso.
- Eliminación de publicaciones con enlaces de sitios Web con phishing en redes sociales, foros o blogs. De la misma manera que se procesa parte del contenido Web para realizar vistas previas, se podría procesar para detectar sitios Web maliciosos.
- Inclusión de estos modelos en antivirus.
- Inclusión de los modelos entrenados en sistemas de detección de Spam para evitar casos de *spear phishing*.

En cuanto investigación:

- La aplicación de métodos de *undersampling* de selección de prototipos en modelos distintos a aprendizaje por similitud

Gracias por su atención

Takeaways:

- Aplicación con éxito de modelos de *machine learning*
- Uso de Python con SkLearn
- Uso de Spark y H2O en instancias de AWS
- Hincapié en la reproducibilidad (notebooks, Anaconda, seeds)