91.25 Ciencia de Datos para la Toma de Decisiones

Trabajo de Aplicación

¨Modelo Predictivo para casos de Phishing¨

Alumnos:

- Juan Cruz Camacho (102376)
- Tobías Pucci Romero (102912)
- Lucas Ken Kiriyama (102487)
- Brian Emanuel Slavkin (102820)

Cuatrimestre: 1º 2021

Docentes:

- Dr. Ing. Emilio Picasso
- Mag. Ing. Mariano Bonoli Escobar
- Mag. Ing. Xavier González

Resumen

¿Es posible detectar con seguridad contundente si un link es phishing sólo en función de la información que puede extraerse de su url? ¿Podríamos (de ser una computadora) saber si es fraudulento solo con verlo? Estas fueron algunas de las incógnitas que nos propusimos antes de comenzar este proyecto.

Nos encontramos en un mundo digital donde la seguridad ya no solo es un aspecto físico sino que nuestros datos, contraseñas y hasta cuentas bancarias están en peligro. Es por esto que nos propusimos encontrar un modelo que nos ponga a salvo de este tipo de estafas virtuales.

Para esto mediante la técnica computacional de Web Scraping, obtuvimos URLs sospechosas de phishing de “Phishtank”. Recopilamos alrededor de 200.000 muestras tanto fraudulentas como benignas. Como nos propusimos obtener todas las variables para el modelo a partir, únicamente, de los links, generamos variables tanto de cantidad de caracteres, características propias del URL como puede ser el sufijo y el protocolo, longitud de cada elemento, entre otras. Además añadimos variables de distancia de string, ya que muchos casos de phishing consisten en links muy parecidos a reconocidas páginas web. Para poder desarrollar estas variables utilizamos la distancia de Jaro-Winkler.

Luego de analizar las variables creadas y su influencia, entrenamos distintos modelos y comparamos sus métricas y matrices de clasificación para encontrar el de mejor performance. Los modelos que trabajamos fueron:

- Regresión Logística
- Support Vector Machine
- Random Forests
- Gradient Boosting

Luego de iterar los parámetros correspondientes a cada modelo, seleccionamos los que maximizan Accuracy, Sensitivity y Specificity, para así poder pasar a la etapa final de análisis y comparación. Finalmente, utilizando solamente información del url pudimos conseguir un modelo con 90.4% de Accuracy, 87.7% Sensitivity y 92.8% Specificity.

Índice

Introducción

Para comenzar la búsqueda de nuestro modelo predictivo, debemos comenzar entendiendo qué es el phishing y cuál es la importancia de lograr evitar estas prácticas en el ámbito cibernético.

La palabra phishing quiere decir suplantación de identidad, se utiliza para denominar al conjunto de técnicas de ingeniería social que utilizan los ciberdelincuentes para obtener información confidencial o personal de otros usuarios de la red de manera fraudulenta y así tomar la identidad del estafado.

La práctica más frecuente de estas técnicas se realiza mediante correos electrónicos falsos los cuales aparentan no serlo, cómo podría ser un servicio de banco, subscripciones de páginas reconocidas por el individuo a estafar, redes sociales, etc. Estos emails derivan en enlaces o links los cuales también aparentan ser reales. Le exigen al usuario sus datos como pueden ser nombre de usuario, contraseñas o cuentas bancarias, mediante encuestas las cuales aparentan ser de los remitentes reales, en los cuales por lo general el estafado confía y así se concreta la obtención de datos del fraude. Otra técnica muy utilizada es mediante archivos adjuntos o botones los cuales generan una descarga la cual puede ser un virus o software del cual el remitente obtiene la información buscada.

Estos enlaces van a ser nuestro principal objeto de estudio, ya que no podemos evitar que el delincuente envíe correos electrónicos puesto que al identificarlos y bloquearlos siempre está la posibilidad de crear uno nuevo.

Es importante también remarcar que los enlaces pueden llegar no solo por mail, sino que hoy en día estas técnicas mutaron a otras vías, como mensajes de texto o redes sociales como WhatsApp, Facebook, Instagram, etc.

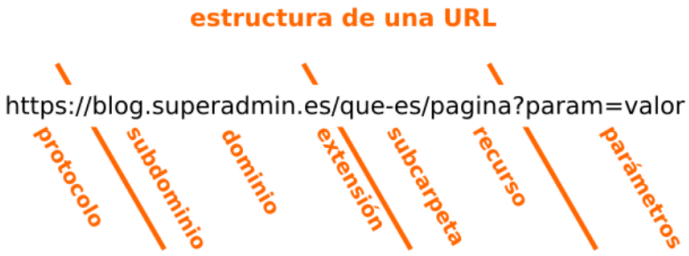
En el año 2011 en la Argentina se presentó un proyecto de ley en el Senado de la nación en el cual se define el concepto:

"La presente Ley tiene como objeto tipificar el delito de obtención ilegítima de datos confidenciales, conocido como "phishing" el cual se define como la capacidad de duplicar una página Web para hacer creer al visitante que se encuentra en el sitio Web original, en lugar del falso. Normalmente, se utiliza con fines delictivos enviando SPAM e invitando acceder a la página señuelo. El objetivo del engaño es adquirir información confidencial del usuario como contraseñas, tarjetas de crédito o datos financieros y bancarios."[1]

Los enlaces a los que nos estamos refiriendo son direcciones web, también denominada URL (Uniform Resource Locator). Estos son una serie de caracteres, números y símbolos que ayudan a localizar un recurso único en la Web, los cuales pueden ser páginas HTML, documentos CSS, imágenes, etc.

La composición de un URL es un concepto clave para el análisis de nuestro trabajo. Esto se puede observar en la siguiente imagen:

[1] Proyecto de ley incorporando el art. 157 ter al Código Penal, tipificando el delito de obtención ilegítima de datos confidenciales ("phishing"). N° de expediente: 2257/11



El protocolo (Scheme en inglés) es el primer componente que siempre se encuentra a la izquierda, la que indica cómo el navegador debe acceder al recurso al cual queremos llegar. Es una cuestión técnica. Los utilizados en las páginas web son http o https, y la diferencia entre estos es que el https impide que otros usuarios puedan interceptar la información confidencial que se transfiere entre el usuario y el servidor.

Luego viene el subdominio y el dominio que indican el servidor web que se solicita. El dominio es el nombre del sitio y el subdominio es una ramificación del dominio que sirve para crear distintas páginas. La extensión es la primera división que se realizó en internet, las cuales se comenzaron creando genéricas como *com*, *net*, *org* y una extensión para cada país del mundo. Como quedan pocos dominios libres en las extensiones fueron surgiendo con el tiempo nuevos.

Luego tenemos componentes optativos como lo pueden ser la carpeta y la subcarpeta también conocidos como el directorio el cual muestra al servidor el camino hacia donde está el recurso. En caso de buscar dentro del dominio específicamente una página, imagen o video el URL incorpora el componente recurso.

Por último tenemos los parámetros, los cuales también son opcionales, y aparecen al final del URL separados mediante el signo de pregunta ¨? ¨. Estos sirven a los programadores para incluir información adicional que utilizan las páginas web para enviar información a los servidores. Consiste, como se puede ver en el ejemplo de una clave y un valor (clave=valor).

Objetivo del análisis

Generar un modelo de clasificación, que explique si una determinada URL sospechosa de phishing lo es realmente o no. Es importante remarcar que nos enfocamos en link de los cuales ya se tenga una presunción de que estos son posibles objetos del tipo de estafa virtual que estamos analizando.

Buscamos obtener un modelo que generalice bien ante nuevas observaciones, permitiendo detectar si una página tiene contenido legítimo o si este es un método de phishing web usando sólamente información que se obtiene del enlace y pudiendo analizar qué alcance puede tener esta técnica.

Este servicio podría usarse como plugin (complemento) en buscadores web y en servicios de email o mensajería para prevenir a los usuarios el acceso a páginas potencialmente peligrosas. En caso de desearse, podremos informar la probabilidad de que dicha página pueda tratarse de una actividad ilegítima a través de una predicción discreta.

Preparación del Dataset

Para la obtención y generación de los datos para poder entrenar un modelo predictivo buscamos obtener un set de datos cuyas muestras representan tanto casos de phishing como páginas reales. Para esto, recurrimos a la página https://phishtank.org/ la cual tiene una comunidad que se encarga de analizar URLs sospechosas proporcionados por los usuarios. Esta página nos proporciona de manera gratuita una base de datos con 11.000 páginas webs inactivas que fueron verificadas como herramientas de phishing en formato CSV. De este archivo obtuvimos el URL y únicamente la variable Dummy de si es realmente phishing o no.

Para poder ampliar esta pequeña cantidad de muestras, utilizamos la herramienta de Web Scraping, en Python con la librería “Scrapy”, para ingresar a PhishTank y obtener muestras tanto “valid” (phishing) e “invalid” (no phishing). El método de scraping consistió en ingresar en la página https://phishtank.org/phish_archive.php la cual es un repositorio de URLs con la información que buscamos. Primero, se filtró la lista por valid y luego por invalid. Dependiendo del largo del URL tuvimos que crear un “re-scraping” en los casos en que no se podían visualizar de manera correcta en la tabla e ingresar en la página de la observación. Se obtuvieron alrededor de 96.000 observaciones “valid” y 91.000 “invalid”. Por lo que se generó un dataset de 198.000 muestras.

A partir de esto, se generaron las siguientes variables en función del URL total:

- URL completo (*url*)
- Phishing (variable respuesta, =0 si no es/=1 si es)
- Scheme (Protocolo del URL)
- Dominio completo (*domain_complete*)
- Dominio del URL
- Subdominio del URL
- Sufijo del URL (*suffix*)
- Dominio y Subdominio (*domain_subdomain*)
- Cantidad de puntos en el dominio (*dom_n_puntos*)
- Cantidad de guiones en el dominio (*dom_n_guion*)
- Cantidad de guiones bajos en el dominio (*dom_n_guionbajo*)
- Longitud del dominio completo (*dom_len_tot*)

- Longitud del dominio (*dom_len*)
- Longitud del subdominio (*dom_len_sub*)
- Cantidad de vocales en el dominio (*dom_vocales*)
- Cantidad de consonantes en el dominio (*dom_cons*)
- Cantidad de números en el dominio (*dom_num*)
- Cantidad de caracteres diferentes en el URL (*dom_car_dif*)
- Dominio es un IP (*dom_ip*): Variable Dummy si el dominio es un IP o no.
- Protocolo es http (*sch_http*): Variable Dummy si el protocolo es http, en caso de no serlo es https.
- Longitud del sufijo (*suf_len*)
- Variable categórica si la variable tiene los siguientes sufijos : com, net, org, ru, xyz, cn, com.br, tk, de, co.uk, info, top, co, fr, me, biz, shop, ly

A partir de estas variables, observamos que muchas observaciones de phishing tenían dominios similares a paginas mundialmente reconocidas y más visitadas. Esto era algo lógico, ya que al ser páginas fraudulentas se buscan asimilar con páginas conocidas, en búsqueda que el usuario no desconfíe y entregue información.

Es por esto, que recurrimos a un dataset de la página Kaggle[2], basado en el repositorio de Alexa.com, una filial de Amazon.com que recoge información de usuarios de internet para generar estadísticas del número de visitas y enlaces a un sitio web. Con las estadísticas, Alexa elabora un ranking de posicionamiento de la web. Este ranking muestra las páginas con mayores visitas a nivel mundial, las cuales consideramos confiables.

Para poder comparar nuestras URLs con las obtenidas de Alexa, utilizamos una medida de similitud entre strings: La distancia o similitud de Jaro-Winkler. Esta métrica nos explica qué tan similares son los strings que se comparan. Esta es una variante de la distancia de Jaro original. Esta similitud se encuentra normalizada, por lo que todos sus valores se encuentran entre 0 y 1. Cuando las secuencias de caracteres son iguales, la métrica es igual a 1 en cambio si no son similares la métrica es igual a 0.[3]

A continuación mostramos algunos ejemplos de estas métricas.

- Dist_JW(amazon , amazon)=1
- Dist_JW(amazon , amaz0n)=0.933
- Dist_JW(amazon , amazon.co.safeamazonsecure)=0.846
- Dist_JW(amazon , breakevents)=0.505

Con esta métrica generamos las variables de similitud con respecto a las primeras páginas del ranking y las que observamos como recurrentes en estafas, añadiendo las distancias a algunos string comúnmente utilizados en páginas web:

[2] https://www.kaggle.com/cheedcheed/top1m
[3] https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

amazon	instagram	google	whatsapp	twitter	facebook	yahoo
wikipedia	baidu	paypal	mail	sfexpress	onedrive	live
excel	square	office365	irs	tencent	creditagrecole	microsoft
blogspot	payment	hsbc	secure	help	banco	bank
support	rakuten	steam	olx	drive	auth	webmail
free	service	account	docs	netflix	url	site
login	index	home				

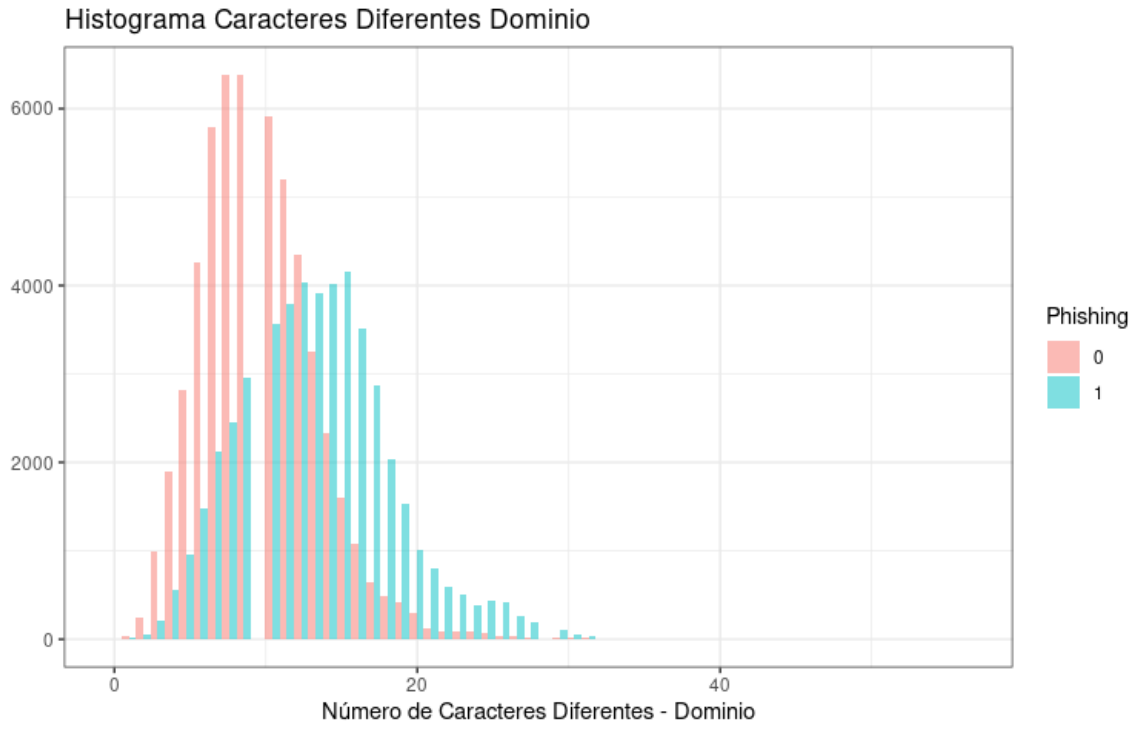
Creada la base de datos, identificamos que existían links obtenidos mediante Web Scraping similares en casi todos sus caracteres en el dominio, lo que nos generaba que las muestras se veían "repetidas". Esta situación generaba una falsa sensación de efectividad del modelo porque al entrenarlo y testearlo con muestras similares, el modelo predice de gran manera, pero no refleja el caso real en el que las muestras sospechosas a clasificar no son parte del dataset. Esto se debe en gran parte, a que las URL repetidas solo varían en una pequeña porción del path, y gran parte de las variables generadas son a partir del dominio.

Por lo cual para terminar la preparación del dataset se eliminaron todas las muestras cuyos dominios sean idénticos y así obtuvimos una base con alrededor de 104.000 muestras.

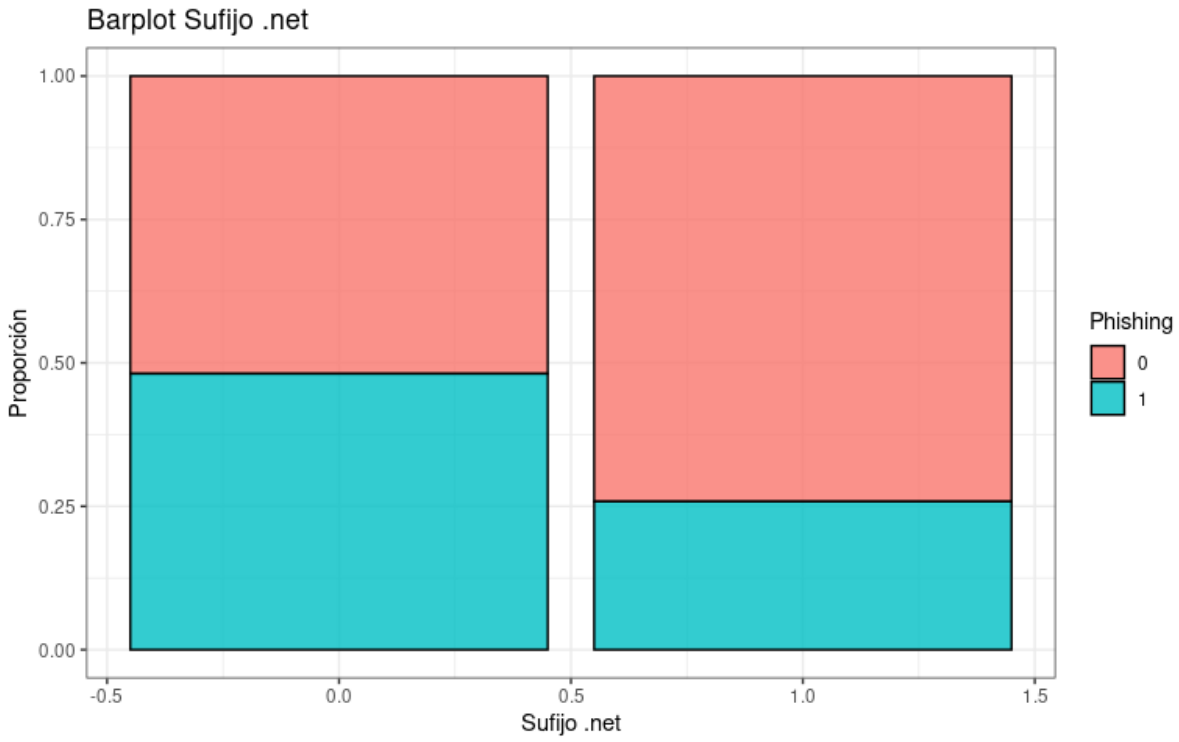
Análisis Exploratorio

Una vez finalizada la etapa de generación de los datos realizamos su análisis exploratorio para conocer las distribuciones de las variables creadas. Pudimos observar que algunas variables creadas a partir de los caracteres o el tipo de sufijo presentan distribuciones diferentes según la clase de la variable respuesta. Consideramos que pueden ser variables que ayudarán al modelo a realizar predicciones correctas.

Vemos el caso de la variable continua **cantidad de caracteres diferentes en el dominio** que presenta cierta diferencia en la distribución, con una mediana de 13 caracteres para sitios malignos frente a 9 para los casos que no son phishing.

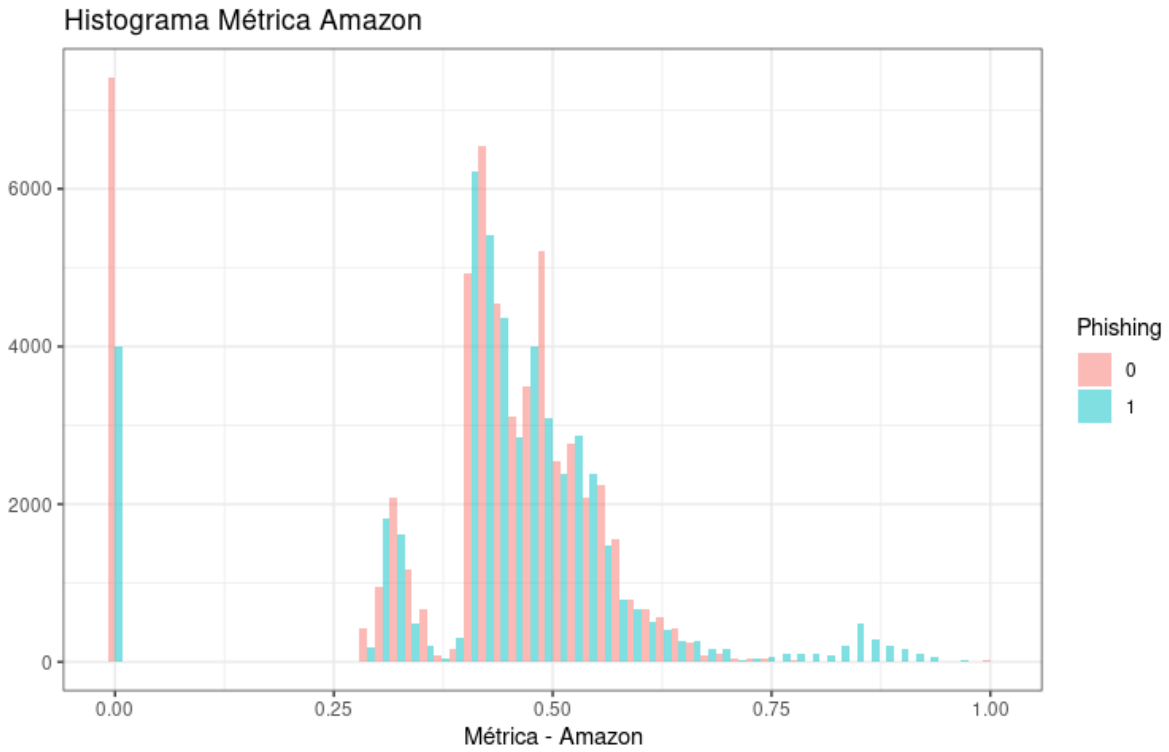


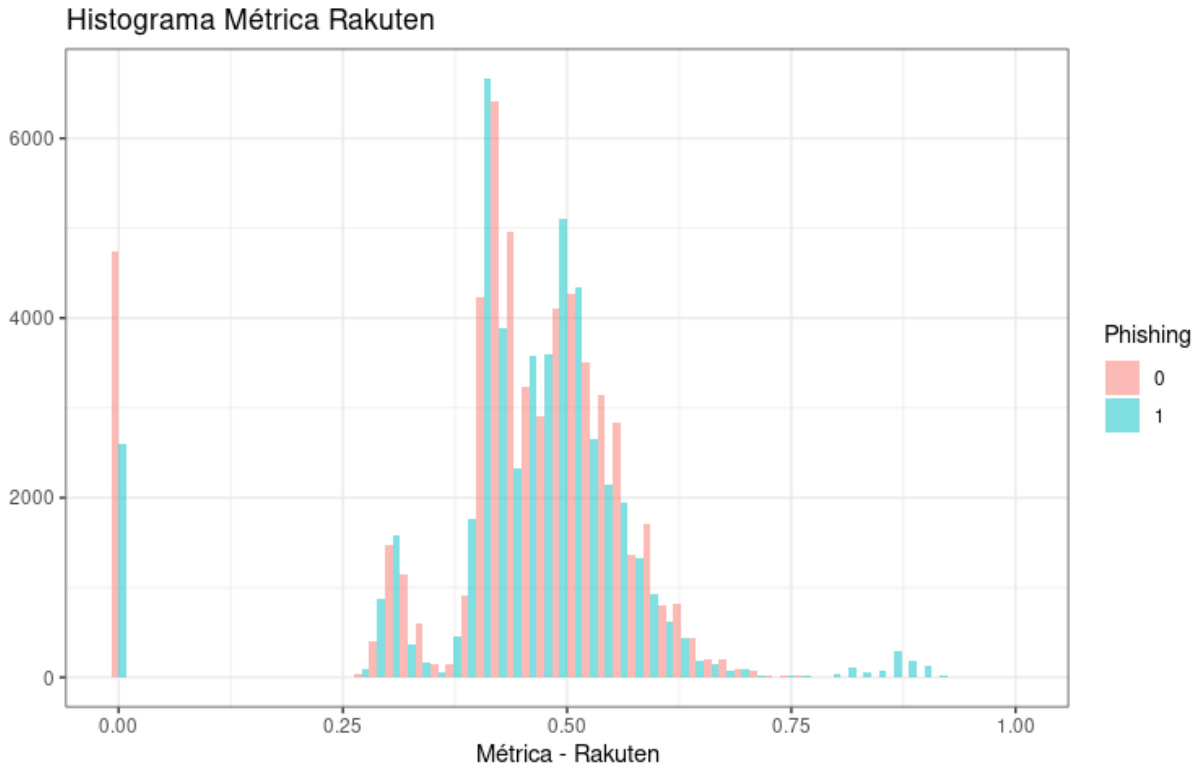
También pudimos observar una diferencia en la proporción cuando se trata del **sufijo .net** donde solamente cerca del 25 % de las observaciones son sitios malignos.



Podemos destacar que el análisis exploratorio con visualización de las variables nos permitió verificar que algunas de las variables de métricas de similitud de strings cumplían con la función que esperábamos. Una cierta cantidad de sitios malignos buscan tener un dominio similar a algunas de las webs más visitadas de e-commerce, como son Amazon y Rakuten. En estos casos, la métrica de similitud Jaro-Winkler tendrá un valor de 1 cuando el dominio sea exactamente igual al de las páginas oficiales, por lo tanto en dicho valor se acumularán todos enlaces benignos. En el caso de los sitios malignos que buscan parecerse a estas tiendas online, la métrica de similitud dará un valor alto (superior a 0.7) por lo que este criterio de corte puede ser utilizado por un modelo de árbol para identificar páginas de phishing.

En las siguientes visualizaciones para la métrica de Amazon y Rakuten podemos encontrar una mayor densidad de sitios malignos entre los valores 0.75 y 0.95 para la medida Jaro-Winkler.

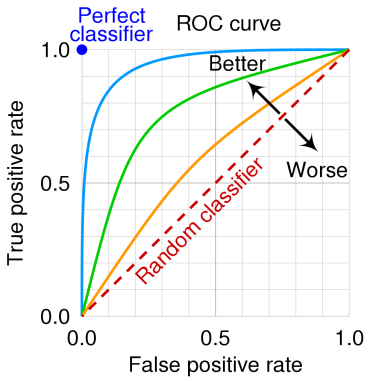

Histograma Métrica Amazon
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
Métrica - Amazon
Phishing
0
1


Histograma Métrica Rakuten
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
Métrica - Rakuten
Phishing
0
1

Métricas de Evaluación

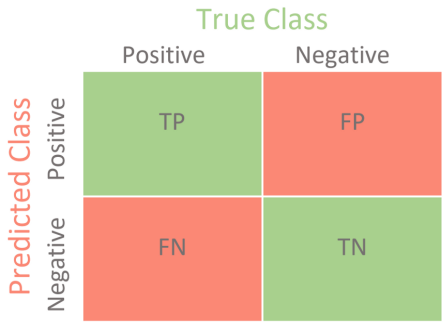
Para poder analizar qué modelo se ajusta mejor a nuestras necesidades nos basamos en las siguientes métricas de evaluación:

- ROC: Es una curva de probabilidad que indica que tanto el modelo de clasificación es capaz de distinguir entre clases. El gráfico de ROC parte de graficar para cada corte de la probabilidad la sensibilidad y la especificidad. El área bajo dicha curva puede usarse como métrica general de predicción de clasificación binaria.



- AUC (Área debajo de la curva ROC) es un índice que varía entre 0.5 y 1.0 donde un 0.5 indica que el modelo no discrimina, es equivalente a tirar una moneda y 1.0 equivale a que el modelo tiene una predicción o discriminación exacta.

A partir de la matriz de confusión o clasificación para métodos de predicción discreta, nos centramos en los siguientes puntos.



- Sensitivity: nos indica la capacidad del modelo de dar verdaderos positivos, es decir, predecir acertadamente los casos de phishing. Es la medida que más nos interesa optimizar, dado que un falso negativo conlleva a la posibilidad de caer en una estafa.
- Specificity: nos indica la capacidad del modelo de dar verdaderos negativos, es decir de predecir correctamente las URLs no maliciosas.

- Accuracy: Indica el acierto general del modelo, calculando qué proporción del total de las muestras son TP y TN. Esta medida, como el ROC, explica que tan bien nuestro modelo se ajusta a los datos.

Modelos

Una vez terminada la preparación del dataset y la exploración de las variables más importantes pasamos a la etapa de entrenamiento de modelos. Realizamos la partición de 80% de los datos para el set de Train y el 20% restante para el Test, reservado para la comparación final.

Usando los datos de entrenamiento trabajamos en la selección de hiperparámetros óptimos para cada familia de modelos. Realizamos 5-fold Cross-Validation para la estimación de las métricas de AUC, Sensibilidad y Especificidad. Con los valores estimados elegimos el mejor set de hiperparámetros para cada modelo y estos fueron comparados entre sí usando el set de testing.

Regresión Logística

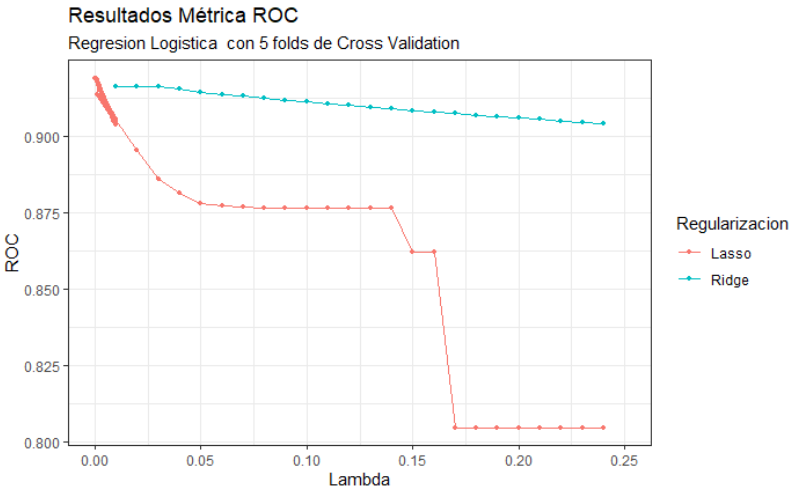
Este modelo resulta útil cuando el resultado a obtener es una variable binaria, tal como es nuestro caso en el cual buscamos predecir si es o no es una herramienta de phishing la URL. Esta es una adaptación de las regresiones lineales múltiples en el cual la variable respuesta tiene una distribución Bernoulli. Mediante la la transformación logit permite realizar la regresión lineal y luego transformar la variable respuesta en la variable deseada.

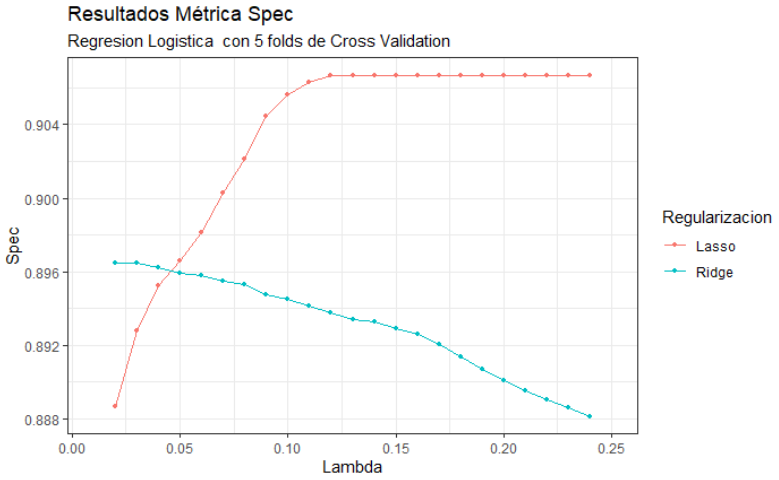
Para poder entrenar el modelo y mejorarlo le añadimos un método de regularización (Shrinking). Este método se utiliza para aliviar la alta colinealidad entre variables explicativas. Cuando existe colinealidad parcial entre las variables produce coeficientes de regresión altos que generan errores en la estimación. Esta técnica lo que introduce en el modelo es una penalidad sobre los coeficientes grandes a fin de mejorar la predicción. Los métodos de regularización que utilizamos fueron:

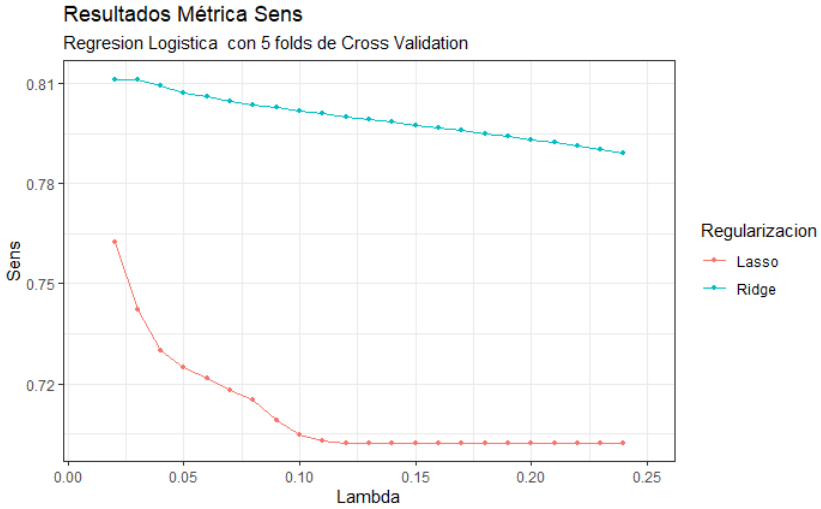
- Ridge: $Error = \sum ||\varepsilon_i||^2 + \lambda\sqrt{\sum \beta_i}$
- Lasso: $Error = \sum ||\varepsilon_i||^2 + \lambda \sum |\beta_i|$

El coeficiente lambda es el parámetro que nos permite regular el grado de penalidad y además permite manejar el compromiso sesgo-varianza. Para poder llevar a cabo la regularización debimos transformar las variables originales obtenidas del dataset original a variables centradas.

En el caso de este método contamos con un único parámetro de tuneo, lambda. Tanto al regularizar con Ridge y Lasso realizamos el entrenamiento mediante Cross Validation, con 5 particiones del dataset de training y obtuvimos los siguientes resultados.







Alpha	Lambda	ROC	Spec	Sens
1(Lasso)	3.00E-04	0.919204156	0.894273599	0.820178571
0(Ridge)	0.01	0.916282945	0.896472574	0.81130102

Analizando los valores obtenidos, podríamos considerar que como el coeficiente de penalización tienden en ambos casos de regularización a 0, evaluamos la regresión logística sin regularización. Los resultados obtenidos fueron similares, pero menores por lo que decidimos quedarnos con estos dos.

Otro factor que analizamos fue añadir un PCA como preprocesamiento. Pero nuevamente no significó cambios sustanciales y en el caso del Ridge disminuye las métricas analizadas del modelo, por lo que también optamos por no utilizarlo.

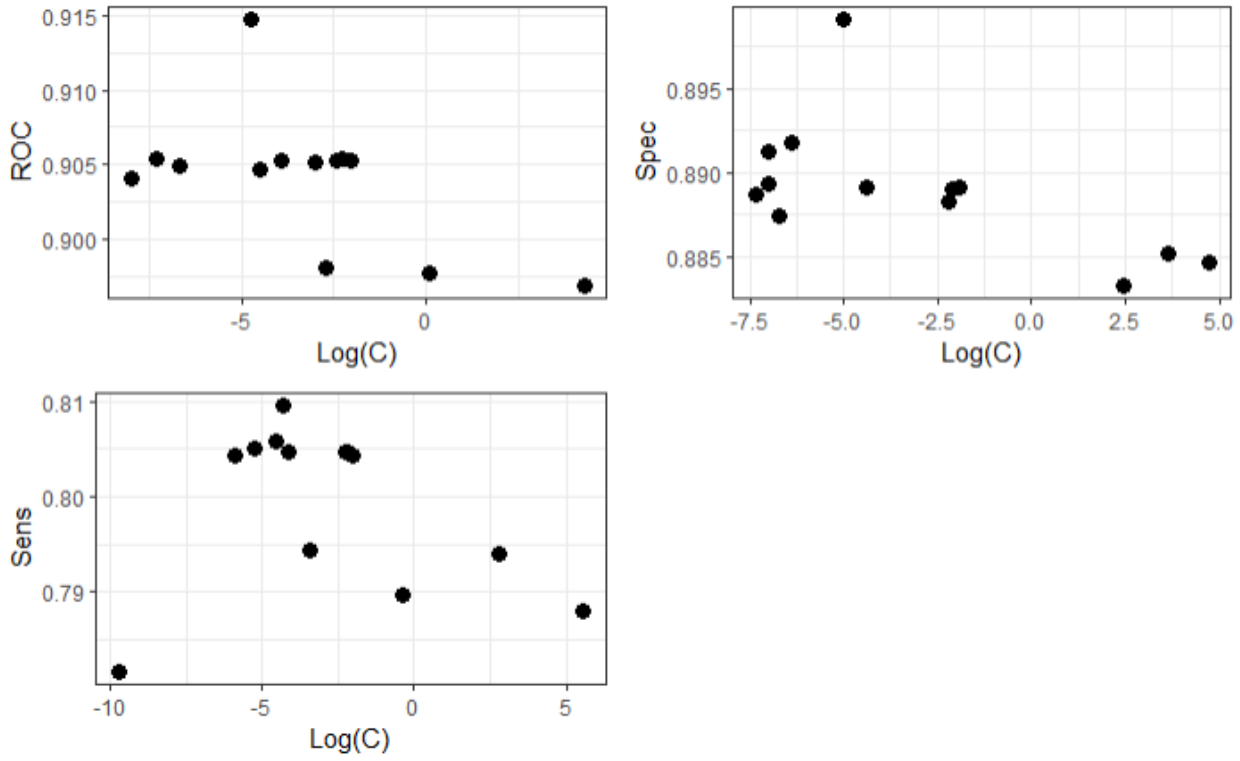
Support Vector Machine:

Este es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Las nuevas muestras se clasifican en función a su posición con el hiperplano (o conjunto).

Existen diferentes tipos de SVMs en función del kernel que se utilice, en nuestro caso, analizamos tanto un SVC (support vector classifier) con kernel "lineal" y otro con un kernel "radial" (RBF). Debemos tener en cuenta que a diferencia de los otros modelos considerados, los SVCs fueron los más computacionalmente exigentes. Por esto, tomamos una muestra del 15% del dataset de training total para realizar la búsqueda de los hiperparametros óptimos por cross validation.

-Kernel lineal:

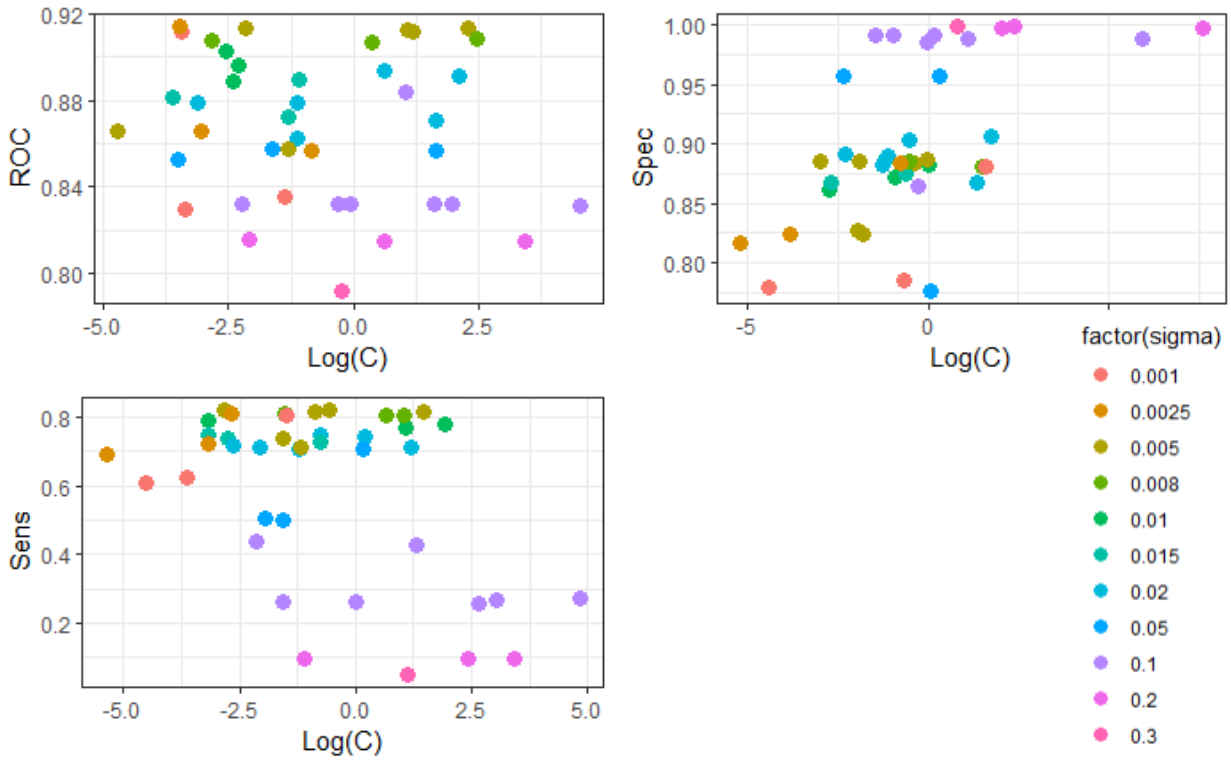
El parámetro a tunear es "C", el cual indica cuánto queremos evitar al clasificar de manera errónea cada observación en el training set.



Como este puede tomar valores óptimos con diferentes órdenes de magnitud, tomamos su logaritmo para graficarlo.

-Kernel radial:

Además del parámetro "C", el modelo con kernel lineal toma el parámetro "sigma" modifica el alcance del hiperplano, un sigma demasiado alto genera overfitting.



KERNEL	C	Sigma	ROC	Spec	Sens
Radial	0.6	0.0025	0.9139168	0.8844139	0.81052283
Lineal	0.1	-	0.9147200	0.8991147	0.8095472

Random Forests

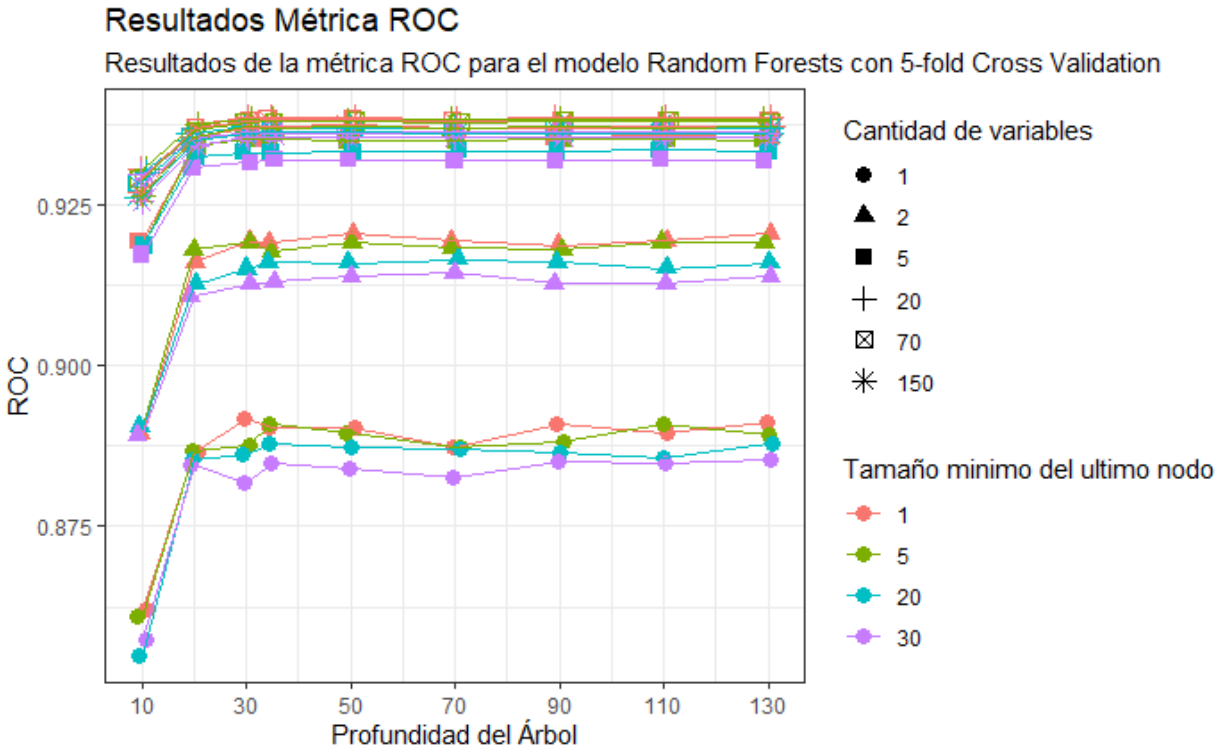
El Random Forests es un modelo que genera "n" árboles de clasificación mediante un bootstrapping a partir del dataset con reposición simulando una "democracia"entre los árboles y selecciona aleatoriamente "m" variables sobre las "p" para cada árbol asegurándose de que los árboles que se obtienen sean distintos. De esta forma se reduce la varianza de los resultados al realizar un promedio entre los resultados de los árboles y eliminar la covarianza entre estos.

Entre los hiperparametros que modificamos del modelo se encuentran:

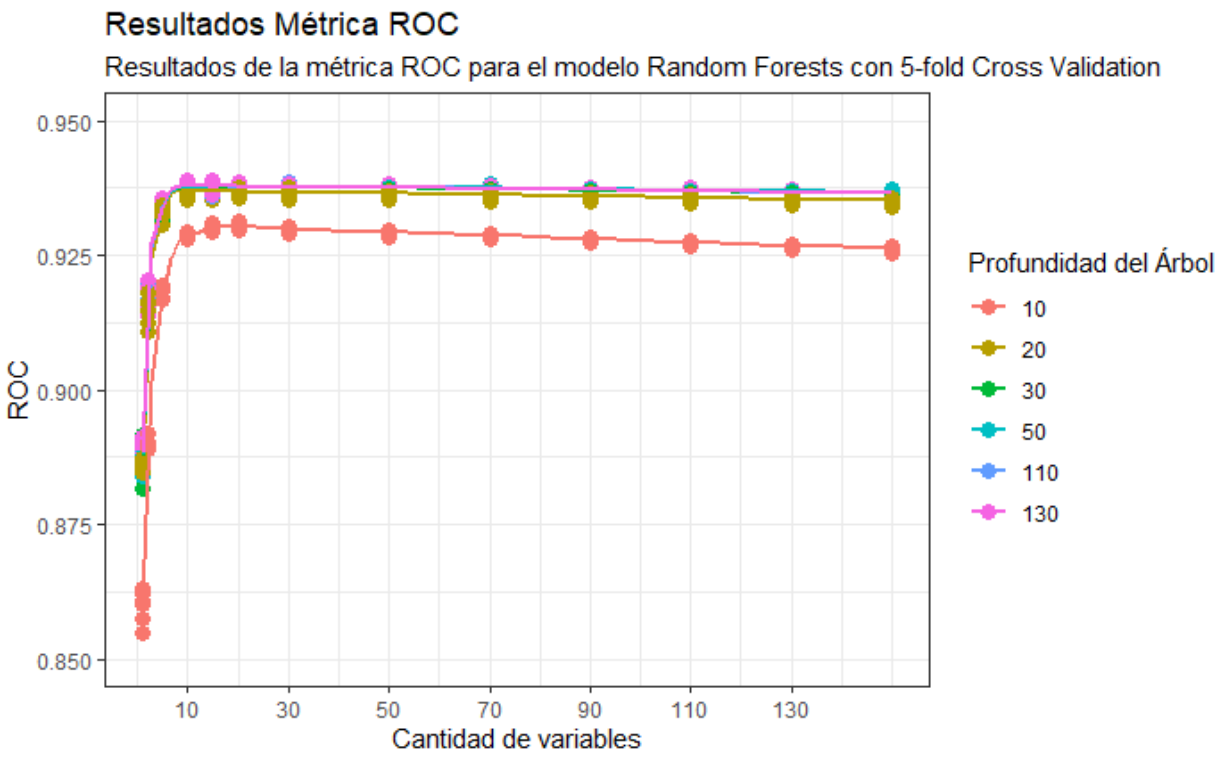
- *Min.node.size*: mínima cantidad de observaciones necesarias que quedan en el último nodo de la rama. Para Random Forests de clasificación se recomiendan valores de 1 pero se puede

entrar en un overfitting al quedar una sola observación y definir la clasificación por lo que decidimos ensayar amplios valores.

- *Max.depth*: profundidad máxima del árbol, "cuántas veces consecutivamente se puede separar una rama hasta llegar a una hoja". Relacionado al hiper parámetro anterior, si se extienden los árboles quedan pocas observaciones en las últimas hojas pudiéndose dar un overfitting.
- *Mtry*: Cantidad de variables "m" que se seleccionan aleatoriamente para cada árbol de clasificación. Seleccionar pocas variables elimina la covarianza entre los árboles volviendolos "independientes" y una gran cantidad los vuelve similares en su "desarrollo".
- *Num.trees:* Cantidad de árboles del Random Forests. Para realizar la pre-selección del modelo decidimos considerar 150 árboles para conseguir gran cantidad de resultados en el espacio muestral debido a la demanda computacional del modelo.



Podemos ver del gráfico que cuando la profundidad del árbol es igual a 20 se estabiliza el ROC y viendo los tamaños del último nodo(colores de las observaciones para cada una de las bandas de cantidad de variables, determinada por la forma) los valores más pequeños, cercanos a 1 se encuentran por encima de los más grandes para todas las cantidades de variables. A simple vista se puede observar que a mayor cantidades de variables, determinadas por la forma, crece la ROC.



Evaluando la ROC en función de la cantidad de variables, esta se estabiliza alrededor de 10 donde llega a un máximo y lentamente comienza a decaer.

Para la pre-selección del modelo probamos 838 Random Forests con 150 árboles debido a la elevada demanda computacional que conlleva crear los distintos árboles multiplicado por la cantidad de "folds" o particiones del cross-validation. Para seleccionar el mejor modelo de Random Forests seleccionamos a los mejores modelos obtenidos y los volvimos a procesar con 1000 arboles para finalmente obtener el mejor Random Forests

num.trees	mtry	max.depth	min.node.size	ROC	Spec	Sens
1000	10	130	1	0.9401	0.9138	0.8506

Gradient Boosting

Al igual que al modelo Random Forest, el modelo Gradient Boosting puede ser considerado un ensamble de modelos simples. En este caso los modelos de árboles de clasificación son "ensamblados" de manera secuencial, de tal forma que el árbol siguiente busca mejorar los errores de las predicciones anteriores. El aporte del árbol siguiente será ponderado por un paso de aprendizaje o learning-rate.

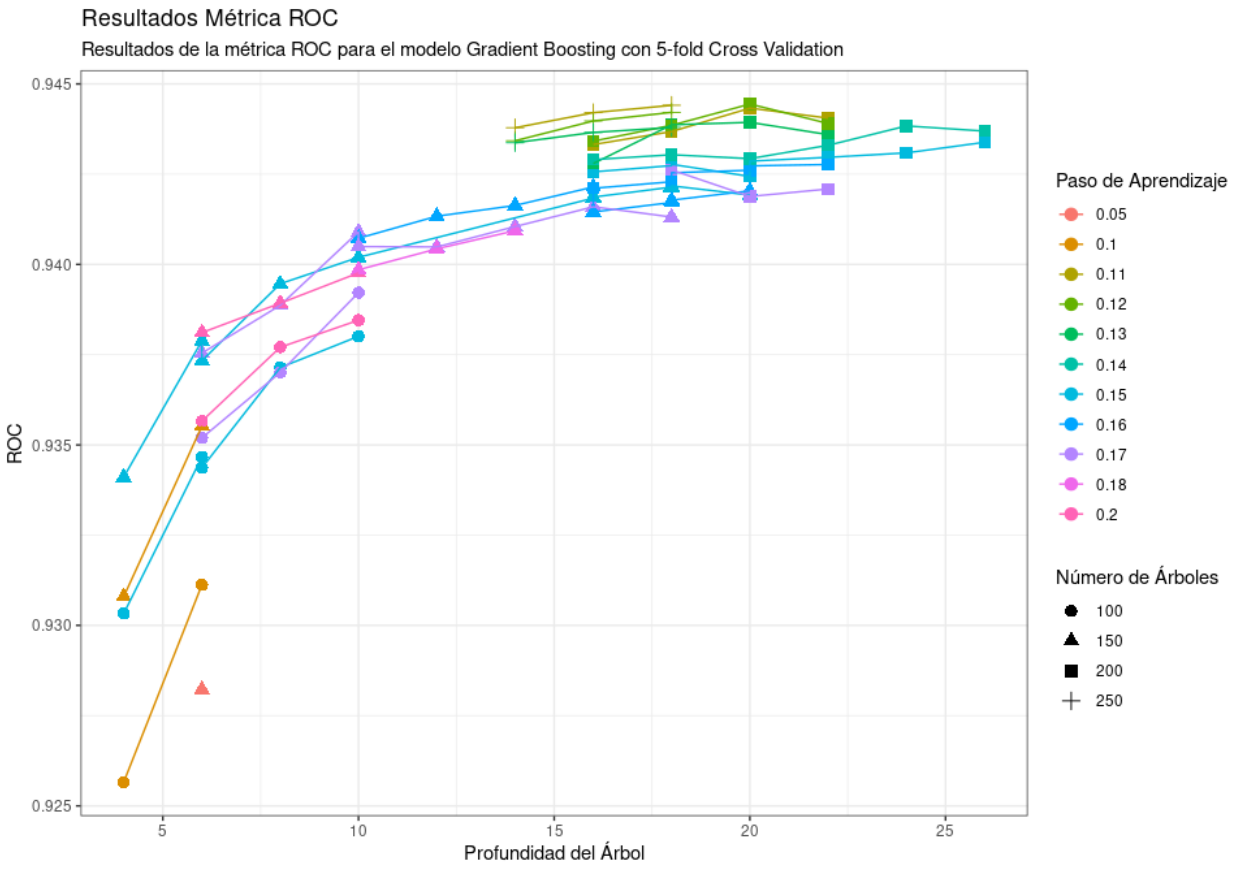
El primer árbol realizará una primera y seguramente errada predicción. El segundo árbol usará las variables explicativas para modelar y predecir los residuos de la predicción del árbol anterior. De manera secuencial se agregan los árboles para mejorar la predicción hasta un punto de corte.

En general los modelos con boosting usan muchos "weak learners" como árboles de clasificación de baja profundidad. Es cierto que en general tienen mejor desempeño estas configuraciones, pero por una cuestión de límites en recursos computacionales decidimos usar hasta 250 árboles con una mayor profundidad.

Los hiperparámetros que usamos para parametrizar este modelo son:

- *n.trees*: Cantidad de árboles secuenciales utilizados.
- *interaction.depth*: Profundidad máxima del árbol.
- *n.minobsinnode*: Cantidad mínima de observaciones en un nodo terminal de cada árbol.
- *shrinkage*: Es el paso de aprendizaje o learning-rate que reduce el efecto de la secuencia de los árboles en la corrección de los errores previos.

Podemos ver en un gráfico de los resultados de 5-fold Cross Validation realizado usando una grilla de valores de los distintos hiperparámetros a optimizar.



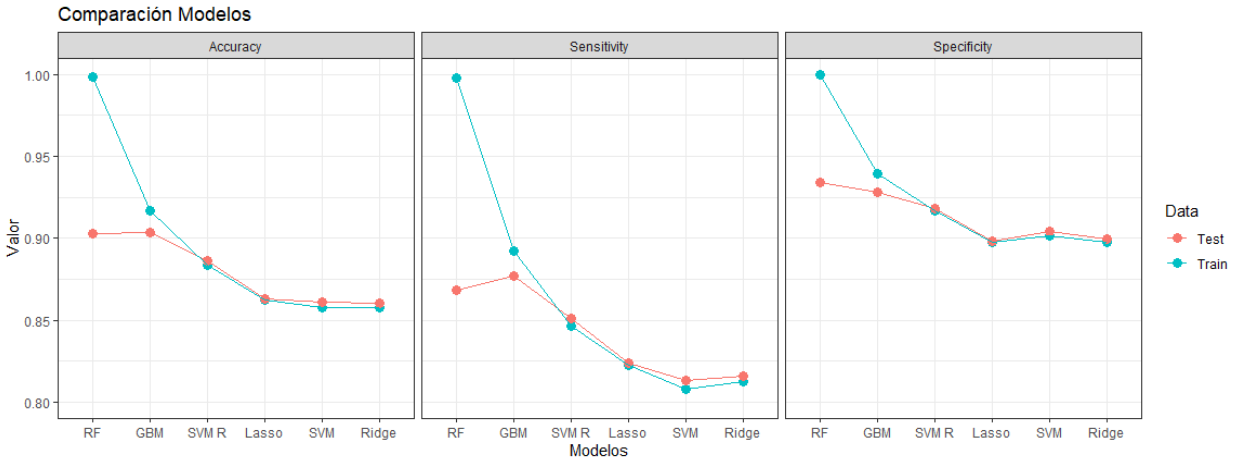
Además incluimos los valores particulares del modelo con mayor valor de la métrica ROC estimada por Cross-Validation.

shrinkage	interaction.depth	n.minobsinnode	n.trees	ROC	Spec	Sens
0.11	18	20	250	0.9444	0.9199	0.8548

Conclusión

La seguridad informática y de datos personales es hoy en día una materia de gran estudio. Además, son temas de gran interés y que tomarán aún más importancia en el futuro. Consideramos que los resultados obtenidos son prometedores, con Accuracy´s de hasta 90% y aún mayor Especificidad en modelos como Gradient Boosting y Random Forest.

Luego de entrenar a cada uno de los modelos con el set completo de entrenamiento y con el mejor set de hiperparámetros obtenidos realizamos la predicción tanto para la variable respuesta en el set de entrenamiento como en el de testing. Los modelos Ridge, Lasso y SVM con kernel lineal tuvieron resultados similares en las tres métricas. Los resultados con SVM con kernel radial son superiores, aunque por debajo de los valores de los métodos de ensamble de árboles como Random Forests y Gradient Boosting. El primero de ellos obtuvo mejor valor de test en Specificity, mientras que el segundo obtuvo mejor valor de Sensitivity. Algo a destacar es que el error de entrenamiento de Random Forests es prácticamente nulo, indicando que hay overfitting y por eso preferimos el modelo Gradient Boosting.

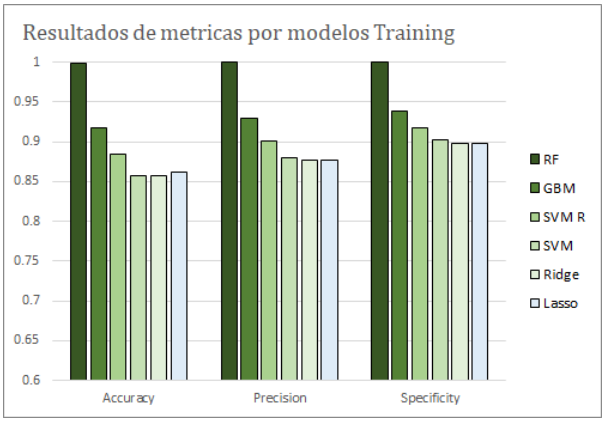


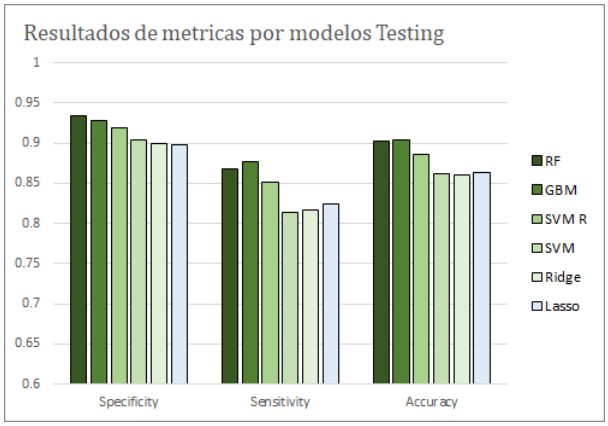
En comparación con otros estudios ya realizados disponibles en la web, donde la base de datos usada contiene variables con mayor dificultad de obtención, nuestros modelos sólamente usan variables creadas a partir del URL y obtienen relativamente buena performance para la información utilizada. Creemos que el modelo final obtenido puede ser utilizado en plataformas como Phishtank para entregar una primera probabilidad de que el URL enviado sea malicioso, pero recomendamos mantener la decisión final a cargo de la comunidad. Además consideramos que las métricas de distancias creadas le dan cierta ventaja al modelo cuando se trata de URLs maliciosas que intentan hacerse pasar por los dominios originales.

Cuando entrenamos el modelo final sin las variables de las métricas tenemos una sensibilidad de 0.95 para URLs con similitud Jaro Winkler superior a 0.70 del dominio Amazon. Cuando incluímos las métricas dicha sensibilidad asciende a 0.99. Algo similar ocurre cuando realizamos el análisis sobre URLs similares al dominio Rakuten, donde sin las métricas obtenemos una sensibilidad de 0.95 y con las métricas una sensibilidad de 0.98. De esta manera creemos que la inclusión de estas

variables ayuda al modelo a tener un mayor poder predictivo con dominios apuntados por los delincuentes.

Presentamos los resultados de las métricas tanto en entrenamiento como en testeo para todos los modelos en gráficos de columnas. Nuevamente podemos apreciar la superioridad de los modelos de ensamblaje de árboles de clasificación y se destaca el overfitting en el Random Forest.





Bibliografía

- Introducción
  - https://www.argentina.gob.ar/justicia/convosenlaweb/situaciones/phishing
  - https://phishtank.org/what_is_phishing.php
  - https://es.wikipedia.org/wiki/Phishing
  - https://superadmin.es/blog/que-es/direccion-web-url/
  - https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL
  - https://es.godaddy.com/blog/diferencia-entre-http-y-https/
  - https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

Anexo

- https://github.com/JuanCruzC97/Proyecto_CDD