

| Documentación | **Alumno: Ferreyra Azul Rocio**  **Tutor : German Rodriguez**  **Profesor: Ignacio Russo Locati**  **Coderhouse**  [**Comisión 52295**](https://plataforma.coderhouse.com/cursos/52295/data-science) |
| --- | --- |
| El desafío propuesto consiste en primer lugar en la realización de : Carga de datos | Exploración | Limpieza| Visualización | Entrenar un modelo adecuado | Observar posibles signos de Overfitting y Underfitting . CARGA DE DATOS Cargamos los datos proporcionados por <https://www.kaggle.com/> , los cuales se encuentran en formato de archivos Csv , estos archivos se cargaron previamente al drive donde procedí con la carga en google colab, brindando permiso para poder hacer la lectura de los mismos.  Procedo a hacer un análisis exploratorio de los datos, y la carga de las librerías correspondientes que estaré utilizando para este proyecto . INTRODUCCIÓN **Objetivo analitico**  El objetivo principal es desarrollar un algoritmo de predicción de seguridad de URLs que pueda identificar con precisión los enlaces que podrían representar riesgos para los usuarios, minimizando los errores de predicción y mejorando la seguridad en línea. En el ámbito de la seguridad informática, la capacidad de predecir con precisión la seguridad de una URL es crucial para proteger la privacidad y la integridad de los usuarios, así como para prevenir ataques cibernéticos y mantener la confianza en línea. El aumento en la cantidad de enlaces maliciosos representa un desafío significativo al exponer a los usuarios a diversos riesgos, como phishing, malware y fraudes en línea, lo que puede resultar en pérdida de datos sensibles y violación de la privacidad. En este contexto, la implementación de técnicas avanzadas de aprendizaje automático y análisis de patrones en grandes conjuntos de datos relacionados con la seguridad en línea nos proporciona la capacidad de identificar patrones y características que sugieren la presencia de URLs maliciosas, permitiendo a los usuarios y sistemas de seguridad tomar medidas preventivas de manera anticipada y proactiva. EXPLORACIÓN DE DATOS |(Data Exploration) Durante la exploración de datos puede identificar que no había presencia de datos null y se observan datos duplicados. Nuestro conjunto de datos consta de 10.000 filas y 50 columnas, de las cuales contamos con 3 columnas de datos de tipo flotante y 47 columnas con datos de tipo entero.  **La información que contiene cada columna es la siguiente:**   1. **id: Identificador único para cada entrada en el conjunto de datos.** 2. **NumDots: Número de puntos (.) en la URL.** 3. **SubdomainLevel: Número de subdominios en la URL.** 4. **PathLevel: Nivel de profundidad del camino en la URL.** 5. **UrlLength: Longitud total de la URL.** 6. **NumDash: Número de guiones (-) en la URL.** 7. **NumDashInHostname: Número de guiones (-) en el nombre de host.** 8. **AtSymbol: Presencia del símbolo "@" en la URL.** 9. **TildeSymbol: Presencia del símbolo "~" en la URL.** 10. **NumUnderscore: Número de guiones bajos (\_) en la URL.** 11. **NumPercent: Número de símbolos de porcentaje (%) en la URL.** 12. **NumQueryComponents: Número de componentes de consulta en la URL.** 13. **NumAmpersand: Número de símbolos de ampersand (&) en la URL.** 14. **NumHash: Número de símbolos de almohadilla (#) en la URL.** 15. **NumNumericChars: Número de caracteres numéricos en la URL.** 16. **NoHttps: Indica si la URL no utiliza HTTPS.** 17. **RandomString: Presencia de una cadena aleatoria en la URL.** 18. **IpAddress: Presencia de una dirección IP en la URL.** 19. **DomainInSubdomains: Presencia del dominio en los subdominios de la URL.** 20. **DomainInPaths: Presencia del dominio en los caminos de la URL.** 21. **HttpsInHostname: Presencia de HTTPS en el nombre de host.** 22. **HostnameLength: Longitud del nombre de host en la URL.** 23. **PathLength: Longitud del camino en la URL.** 24. **QueryLength: Longitud de la consulta en la URL.** 25. **DoubleSlashInPath: Presencia de doble barra diagonal (//) en el camino de la URL.** 26. **NumSensitiveWords: Número de palabras sensibles en la URL.** 27. **EmbeddedBrandName: Presencia de un nombre de marca incrustado en la URL.** 28. **PctExtHyperlinks: Porcentaje de hiperenlaces externos en la URL.** 29. **PctExtResourceUrls: Porcentaje de URLs de recursos externos en la URL.** 30. **ExtFavicon: Presencia de un favicon externo en la URL.** 31. **InsecureForms: Presencia de formularios inseguros en la URL.** 32. **RelativeFormAction: Acción de formulario relativa en la URL.** 33. **ExtFormAction: Acción de formulario externo en la URL.** 34. **AbnormalFormAction: Acción de formulario anormal en la URL.** 35. **PctNullSelfRedirectHyperlinks: Porcentaje de hiperenlaces nulos de redirección propia en la URL.** 36. **FrequentDomainNameMismatch: Frecuencia de desajuste en el nombre de dominio en la URL.** 37. **FakeLinkInStatusBar: Presencia de enlace falso en la barra de estado.** 38. **RightClickDisabled: Función de clic derecho deshabilitada en la URL.** 39. **PopUpWindow: Presencia de ventana emergente en la URL.** 40. **SubmitInfoToEmail: Envío de información a través de correo electrónico en la URL.** 41. **IframeOrFrame: Presencia de iframe o frame en la URL.** 42. **MissingTitle: Falta de título en la URL.** 43. **ImagesOnlyInForm: Presencia de imágenes únicamente en el formulario de la URL.** 44. **SubdomainLevelRT: Relación temporal del nivel de subdominio en la URL.** 45. **UrlLengthRT: Relación temporal de la longitud de la URL.** 46. **PctExtResourceUrlsRT: Relación temporal del porcentaje de URLs de recursos externos en la URL.** 47. **AbnormalExtFormActionR: Relación temporal de la acción de formulario externo anormal en la URL.** 48. **ExtMetaScriptLinkRT: Relación temporal de metadatos, script o enlace externo en la URL.** 49. **PctExtNullSelfRedirectHyperlinksRT: Relación temporal del porcentaje de hiperenlaces nulos de redirección propia en la URL.** 50. **CLASS\_LABEL: Etiqueta de clase que indica la seguridad de la URL (0 para segura, 1 para no segura).** | HERRAMIENTAS  * Python * Pandas * Numpy * Seaborn * Matplotlib * Scikit-learn   Archivo Csv RECONOCIMIENTOS [**https://matplotlib.org/stable/gallery/index.html**](https://matplotlib.org/stable/gallery/index.html)  [**https://seaborn.pydata.org/examples/index.html**](https://seaborn.pydata.org/examples/index.html)  [**https://www.kaggle.com/**](https://www.kaggle.com/)  [**https://www.virustotal.com**](https://www.virustotal.com)  [**https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning/data**](https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning/data)  [**https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls**](https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls)  **Un reconocimiento a mi tutor German rodríguez por los tutoriales compartidos a través de Youtube:** |

# FORMULACIÓN DE HIPÓTESIS

* **Hipótesis 1: Impacto del Contenido y Origen de la URL en su Seguridad**
  + Las URLs que conducen a contenido sensible o de baja reputación tendrán una mayor probabilidad de ser identificadas como no seguras.
* **Hipótesis 2: Relación entre Longitud y Estructura de la URL con su Seguridad**
  + Se plantea que la longitud y complejidad estructural de una URL pueden estar relacionadas con su nivel de seguridad.
  + Las URLs extremadamente largas o con una estructura compleja pueden ser indicativos de intentos de ocultar contenido malicioso, lo que las hace más propensas a ser clasificadas como no seguras.
  + Las URLs más cortas y simples podrían estar asociadas con contenido legítimo y, por lo tanto, ser clasificadas como seguras con mayor frecuencia.
* **Hipótesis 3: Influencia de Variables Externas en la Seguridad de las URLs**
  + Se plantea que factores externos, como la presencia de redirecciones, el uso de HTTPS, y la presencia de palabras clave sensibles, pueden influir en la seguridad percibida de una URL.

# LIMPIEZA DE DATOS| (Data Cleaning)

Se realiza la eliminación de datos duplicados (658) y eliminación de las siguientes columnas, las cuales no son requeridas para este proyecto:

'id', 'RandomString', 'IpAddress', 'MissingTitle', 'NumDots', 'NumHash', 'NumPercent', 'PctExtHyperlinks', 'PctExtResourceUrls', 'ExtFavicon', 'PctNullSelfRedirectHyperlinks', 'FrequentDomainNameMismatch', 'FakeLinkInStatusBar', 'RightClickDisabled', 'PopUpWindow', 'SubmitInfoToEmail', 'IframeOrFrame', 'ImagesOnlyInForm', 'SubdomainLevelRT', 'UrlLengthRT', 'PctExtResourceUrlsRT', 'AbnormalExtFormActionR', 'ExtMetaScriptLinkRT', 'PctExtNullSelfRedirectHyperlinksRT'

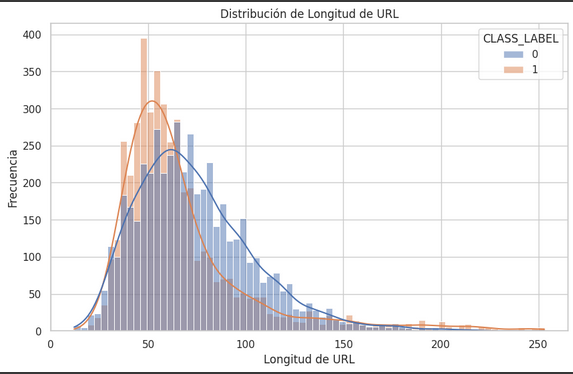
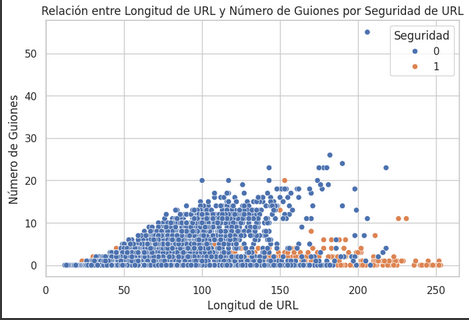
Esto nos deja con un total de 10.000 filas y 26 columnas de las cuales todas son del tipo entero.

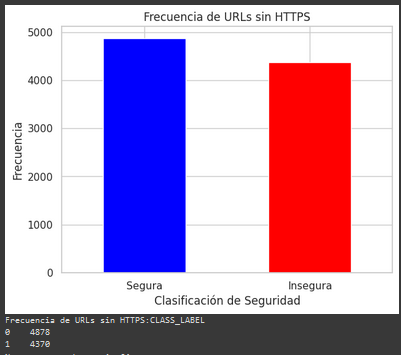
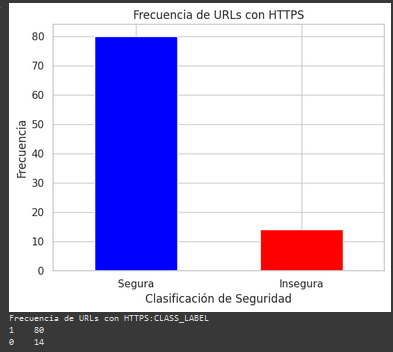
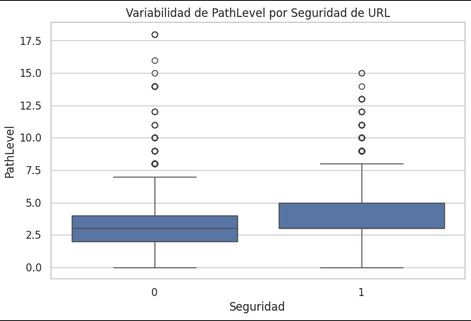
# VISUALIZACIONES DE DATOS|(Data Visualization)

## Descripción

Utilice la biblioteca **matplotlib** junto con **seaborn** para crear un conjunto de visualizaciones que exploran diferentes aspectos de los datos :

1. **Distribución de longitud de URL:**
   * Este gráfico muestra la distribución de la longitud de las URL En el eje horizontal (x), tiene diferentes longitudes de URL, mientras que en el eje vertical (y) tienes la frecuencia con la que aparecen esas longitudes. El gráfico utiliza diferentes colores para representar dos categorías diferentes de URL (seguras y no seguras), lo que permite ver cómo se distribuyen las longitudes de URL dentro de cada categoría.
   * En este conjunto de datos podemos observar que la frecuencia en la que aparecen las URL inseguras es superior que las URL seguras . Tenemos URL entre 10 a 250 caracteres para las URL inseguras , llegando a picos de 400 en la frecuencia, mientras que las URL seguras tienen una longitud menor a 250, y picos en frecuencia de 300, por lo que son URL más cortas en caracteres y menos frecuentes en nuestros datos.
2. **Presencia de HTTPS:**
   * Contamos con dos gráficos, ambos contienen la frecuencia de las URLs y su clasificación de seguridad. Podemos observar que el total de URLs que **no cuentan con presencia de HTTPS** es de un total de 4878 para la clase segura y de 4370 para la clase insegura.
   * Por otro lado contamos con datos de un total de 80 para la clase segura, y 14 para la clase insegura, **con presencia de protocolo HTTPS** en cada URLs en nuestro conjunto de datos.
   * Por lo cual, no contamos con un gran número de URLs que tengan presencia de protocolo HTTPS en nuestro conjunto de datos. Esto nos indica que claramente la presencia de HTTPS está asociada con un mayor nivel de seguridad percibida. El bajo número de URLs con HTTPS en la clase insegura respalda la idea de que HTTPS es un factor importante en la evaluación de la seguridad de una URL.
3. **Variabilidad del nivel de la ruta:**
   * Este gráfico permite comparar la dispersión de los niveles de ruta entre las URL seguras e inseguras, lo que ayuda a comprender cómo varía la estructura de la ruta en función de la seguridad de la URL.
   * La variabilidad de la ruta de una URL se refiere a la capacidad de la URL para cambiar su estructura y contenido en función de diferentes parámetros, variables o contextos. Podemos observar entonces, una diferencia en la estructura o la complejidad de las rutas de las URLs , en este caso las URL seguras tienden a ser más complejas que las URL inseguras.
4. **Relación entre longitud de URL y número de guiones:**
   * Este gráfico es un diagrama de dispersión que muestra la relación entre la longitud de las URL y el número de guiones, distinguiendo entre la seguridad de las URLs. En el eje x, se representa la longitud de las URL, mientras que en el eje y se muestra el número de guiones presentes en las URL.
   * En este gráfico podemos observar entonces que las URLs seguras contienen más guiones que las URLs inseguras, pero son las URLs inseguras las que contienen mayor longitud.

En resumen, estas visualizaciones nos proporcionan una exploración visual de varias características y relaciones en nuestro conjunto de datos de URL, lo que puede ayudarnos a entender mejor la distribución y las tendencias. 



**Selección de Características y Variable Objetivo:**

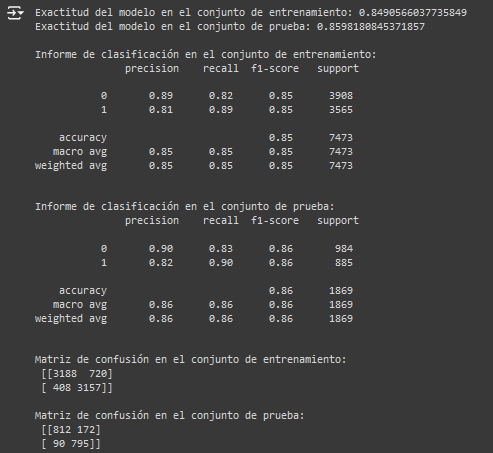
La regresión logística destaca como una opción sólida para predecir la seguridad de las URL debido a su facilidad de comprensión, rapidez y capacidad para manejar una amplia variedad de tipos de datos. Por estas razones, opté por esta alternativa.

Por lo que cree una variable X que contiene todas las columnas exceptuando la columna “CLASS\_LABEL”, para poder predecir la variable objetivo. Y por otro lado, cree la variable Y, que contiene la variable objetivo, es decir la columna “CLASS\_LABEL”. Esta variable contendrá las etiquetas que se están tratando de predecir (seguras o inseguras)

Continúe dividiendo los datos en conjuntos de entrenamiento y prueba, donde el 20%de los datos se utilizan como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento.

1. X\_train: Contiene las características del conjunto de entrenamiento.
2. X\_test: Contiene las características del conjunto de entrenamiento.
3. Y\_train: Contiene las etiquetas del conjunto de entrenamiento.
4. Y\_ test: Contiene las etiquetas del conjunto de prueba

**RESULTADOS**

****

Los resultados nos muestran que el modelo de regresión logística en el conjunto de prueba, alcanza una exactitud del 85.75%, lo que significa que aproximadamente el 85.75% de las predicciones son correctas. La matriz de confusión muestra 182 falsos positivos y 103 falsos negativos. En resumen, el modelo parece tener un buen rendimiento general, con precisiones altas para ambas clases. La matriz de confusión revela que el modelo comete más errores al clasificar la clase segura que la clase insegura.

**Overfitting**: No se observan signos claros de overfitting, ya que las métricas en el conjunto de prueba son comparables a las del conjunto de entrenamiento.

**Underfitting**: Aunque el rendimiento general es bueno, existe margen para mejoras.

**Formas de Mejorar**: Explorar la posibilidad de incluir características adicionales, podría mejorar la capacidad predictiva del modelo.

En resumen, los resultados indican que el modelo de regresión logística está funcionando adecuadamente.

El desafío propuesto en segundo lugar consiste en la obtención de datos adicionales que pueda ayudar a mejorar el rendimiento del modelo | Exploración | Limpieza | Trabajar con APIs | Unión de DataFrames.

# PREPARACIÓN DEL ENTORNO DE API

Para realizar la solicitud de datos en VirusTotal, anteriormente realice la carga de un conjuntos de datos adicional, el cual contenía URLs y su clasificación ("good"| "bad"). De esta forma seleccione algunas URLS y solicite a la API información sobre ellas. Esto nos proporcionó información detallada sobre cada URL escaneada, incluyendo :

1. **filescan\_id**: Identificador del escaneo de archivo asociado.
2. **permalink**: Enlace permanente relacionado con el recurso escaneado.
3. **positives**: Número de motores de escaneo que detectaron el recurso como malicioso.
4. **resource**: Recurso escaneado, puede ser una URL, un archivo, etc.
5. **response\_code**: Código de respuesta del escaneo.
6. **scan\_date**: Fecha en la que se realizó el escaneo.
7. **scan\_id**: Identificador único del escaneo.
8. **scans**: Resultados detallados del escaneo.
9. **total**: Número total de motores de escaneo utilizados en el escaneo.
10. **url**: URL del recurso escaneado.
11. **verbose\_msg**: Mensaje detallado sobre el resultado del escaneo.

Ya que VirusTotal tiene un cierto límite a la hora de solicitud de datos no continúe trabajando con los datos obtenidos, ya que eran escasos.

# PREPROCESAMIENTO DE DATOS|(Data Preprocessing)

Continuamos con la limpieza de los datos en el conjunto dataURL, comprobando la presencia de datos null y eliminación de datos duplicados (42150). Este conjunto de datos cuenta con dos columnas y 549346 filas las cuales ambas son del tipo object.

**Transformación de Datos (Data Transformation)**

Para realizar la unión entre los conjuntos de datos se realizan transformaciones en el tipo de datos de object a int , reducción de tamaño en ambos conjuntos de datos, renombre de la columna ( 'Label') a ( 'CLASS\_LABEL'), y reemplazamos los datos ("good","0"| "bad","1"). De esta forma obtuvimos un conjunto de datos con URLs y su clasificación segura (0) e insegura (1).

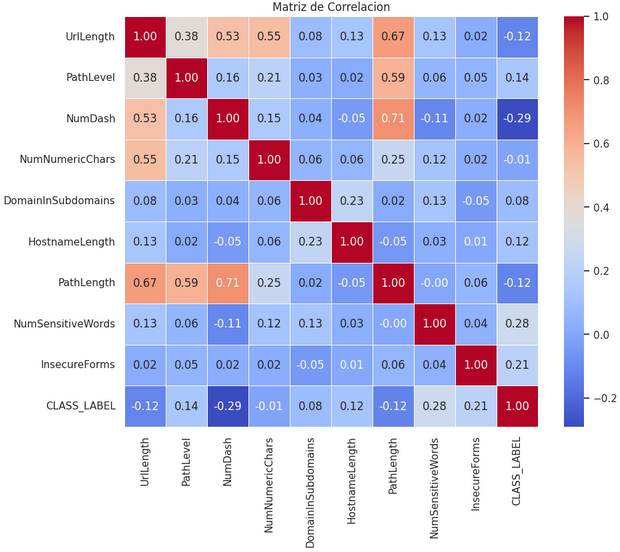
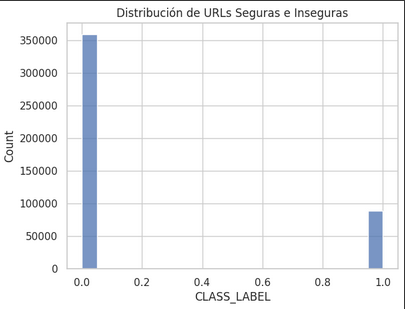
**Integración de Datos (Data Integration)**

Realizó la fusión de ambos conjuntos de datos (data y dataURL). Sólo se conservarán las filas que tienen una coincidencia en ambas tablas en la columna "CLASS\_LABEL", dejándonos un conjunto de datos llamado URL con un total de 27 columnas de las cuales solo una es de tipo object mientras que las demás son de tipo entero y 446938 filas.

El desafío propuesto en tercer lugar consiste en la realización de análisis univariado y multivariados | Realización de sobremuestreo | Normalización| Entrenamiento de modelos.

**Análisis univariado:** Como resultado obtuvimos 350 mil URLs clasificadas como seguras, mientras que la cantidad de URLs inseguras es menor a 100 mil. Esto indica un desequilibrio en los datos.

**Análisis multivariado:** Se observa poca correlación lineal positiva entre las variables seleccionadas. Esto sugiere que no están fuertemente correlacionadas entre sí. Sin embargo la variable "UrlLength" muestra una correlación lineal fuerte hasta "CLASS\_LABEL". En este contexto, significa que la longitud de la URL tiene una fuerte relación lineal con la clase de la URL (segura o no segura).Esto sugiere que la longitud de la URL puede ser significativo en la seguridad de la misma. En otras palabras, URLs más largas pueden tener una mayor probabilidad de ser clasificadas como inseguras, o viceversa.

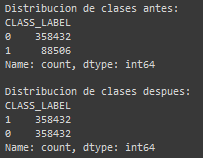


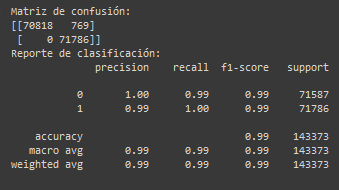
**Sobremuestreo (Oversampling):** Realizó un sobremuestreo para abordar el desequilibrio de clases .Seleccionando las 10 características que más se relacionan con una URL insegura y utilizando el método oversampler aumentamos el número de URLs inseguras para poder obtener un mejor equilibrio entre los datos. Finalmente, converti el conjunto de datos a un formato más fácil de manejar.

X\_resampled\_df : Nuestras características seleccionadas y sobremuestreadas a un nuevo conjunto de datos.

y\_resampled\_df : Convierte nuestras etiquetas sobremuestreadas a una nueva serie.

El sobremuestreo ha sido efectivo para equilibrar las clases en este conjunto de datos.



**Bosque Aleatorio (Random Forest):** Utilizamos el conjunto de características y etiquetas que obtuve anteriormente del sobremuestreo (X\_resampled\_df | y\_resampled\_df). El 20% de los datos se utilizarán para la prueba y el 80% para el entrenamiento. 

**Matriz de Confusión**: El modelo clasificó correctamente 70818 instancias para la clase segura y 71786 instancias de la clase insegura. Hubo 769 instancias de la clase segura que fueron incorrectamente clasificadas como clase insegura, y ninguna instancia de la clase insegura fue incorrectamente clasificada como clase segura.

**Reporte de Clasificación:** La precisión para la clase segura es de 1.00, indicando un 100% en predicciones clasificadas como clase seguras que son realmente clases seguras. La precisión para la clase insegura es del 0.99, indicando que el 99% de las clases clasificadas como inseguras realmente son inseguras.

**Recall:** Para la clase segura es de 0.99, indicando que el 99% de las instancias de clase seguras fueron correctamente identificadas. Mientras que para la clase insegura es de 1.00, lo que indica que el 100% de las clases inseguras fueron correctamente identificadas.

**F1-score:** Indica un buen equilibrio entre ambas la precisión y el recall .

**Accuracy:** La precisión general del modelo es del 99% .

**Conclusión** : El modelo de bosque aleatorio está funcionando correctamente, con alta precisión general y pocos errores en la clasificación.

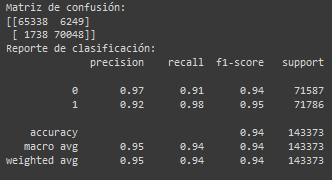
**Clasificador de Perceptrón Multicapa (Red Neuronal):** El modelo de red neuronal clasificó correctamente 65338 instancias como clase segura, sin embargo hubo 6249 instancias de la clase segura que fueron incorrectamente clasificadas como clase insegura. La precisión para la clase segura es de 0.97, lo que indica un 97% de predicción correcta para la clase segura.

Para la clase insegura el modelo de red neuronal clasificó correctamente 70048 instancias como clase insegura, hubo 1738 instancias de la clase insegura que fueron incorrectamente clasificadas como clase segura. La precisión para la clase insegura es de 0.92, indicando un 92% de predicción correcta para la clase inseguras.

**Recall**: Para la clase segura es de 0.91 , indicando un 91% de clases que fueron correctamente identificadas. Mientras que para la clase insegura fue de un 0.98%, un 98% de instancias que fueron correctamente identificadas.

**F1-score**:Para la clase segura es de 94% y para la clase insegura es del 95% .

**Accuracy**: La precisión general del modelo es de 94%.



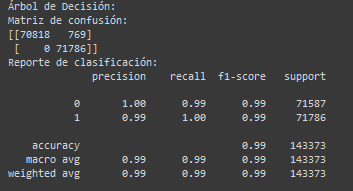
**Conclusión**: El modelo multicapa está funcionando correctamente, con una precisión del 94%, hay algunos puntos a mejorar, ya que indica una cantidad significativa de falsos negativos en la clase segura y una cantidad significativa en la clase insegura con falsos positivos.

**Árbol de Decisión:** El modelo clasificó correctamente 70818 instancias de clases seguras. Las instancias predichas como clase segura son 100% correctas . 71786 de clases inseguras , de las cuales 769 son clases seguras clasificadas incorrectamente como clase insegura. No hubo instancias de clase inseguras clasificadas incorrectamente como clase segura.

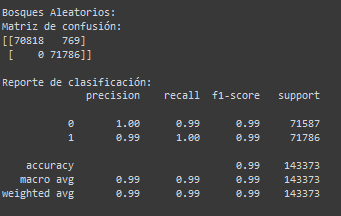
**Recall :** El 99% de las instancias de clase segura fueron correctamente identificadas mientras que el 100% de las clases inseguras fueron correctamente identificadas .

**Accuracy**: La precisión general del modelo es de 99%.

**Conclusión:** El modelo funciona extremadamente bien con una gran precisión general del 99% lo que puede llegar a indica inconsistencias, puede estar teniendo sobre ajustes de los datos entrenados, se debe realizar validación cruzada para garantizar la seguridad.



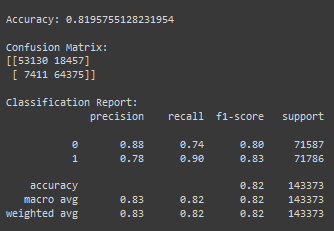
**Bosques Aleatorios:** Al igual que el modelo de árbol de decisión, se obtuvieron los mismos resultados . Funciona extremadamente bien con una precisión del 99%.



Basándose en los resultados obtenidos, el modelo de Bosques Aleatorio (Randon Forest) parece ser el más adecuado debido a su alta precisión y buen equilibrio entre la precisión y el recall, sin embargo es fundamental realizar la validación cruzada y ajustes de hiperparametros para asegurarnos.

**Regresión Logística:**

Comparando el este modelo con el primer modelo que realice, se obtuvo un buen rendimiento en términos de precisión y exactitud pero con algunas diferencia significativa. Este modelo obtuvo una mayor precisión para la clase segura del 88%, mientras que para la clase insegura fue del 78%. Se obtuvo un recall del 74% para la clase segura y un 90% para la clase insegura. Este modelo obtuvo una precisión general ligeramente mayor en el conjunto de datos de prueba.

****

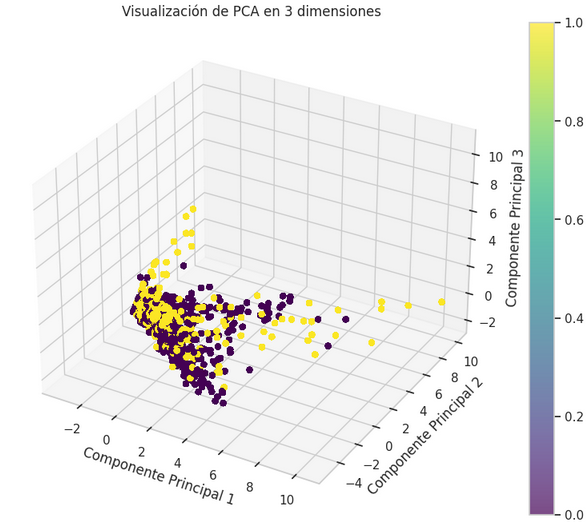
El desafío propuesto en cuarto lugar consiste en la Creación de una Variable Sintética| Implementacion de PCA | Validación de Modelos| Respondiendo a las Hipótesis.

# Ingeniería de Características (Feature Engineering)

Se crea una variable sintética llamada "Security\_Score".Esta variable se genera sumando diferentes características en el conjunto de datos, son una combinación de variables relacionadas con la estructura de la URL y otros aspectos para evaluar la seguridad. Selecciono las columnas que se utilizarán para verificar los datos y realizar la normalización. Para ello se calcula el valor máximo y mínimo para cada característica seleccionada. Procedo con la normalización aplicando el escalador MinMaxScaler de la biblioteca scikit-learn .

**Análisis PCA :**

Con el análisis de PCA hemos reducido las dimensiones a 3 componentes principales. En este caso, el primer componente principal explica aproximadamente el 33% de la variabilidad, el segundo componente principal el 20%, y el tercer componente principal el 13%. Ahora, con el gráfico de dispersión en 3D, podemos ver cómo se distribuyen los datos en función de estos tres componentes principales: Vemos una concentración alta de puntos en el centro del gráfico, especialmente en los rangos componente principal 1(-2,4),componente principal 2 (-2,6), mientras que en el componente principal 3 (-2,0). Esto nos sugiere una gran cantidad de datos que se comportan de manera similar en esas direcciones específicas. Esta concentración de puntos nos indica que hay cierta coherencia o patrón en esos aspectos de nuestros datos.

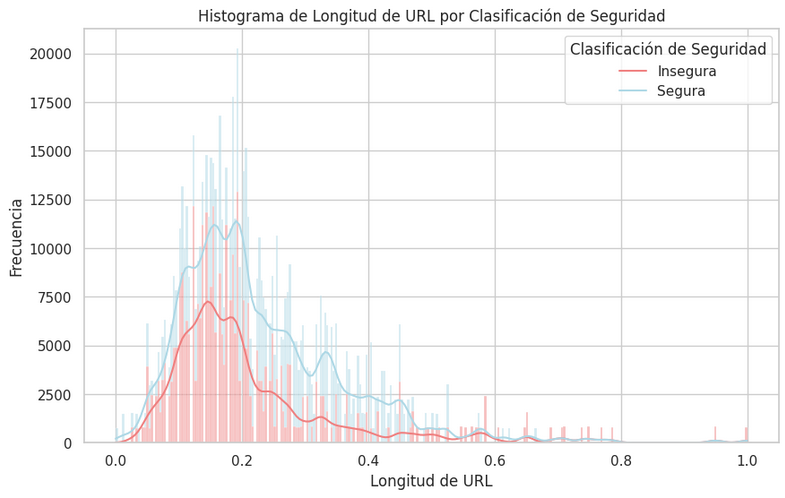


Para poder visualizar los patrones generales en nuestro conjunto de datos en figuras 2D o 3D se deberá utilizar X\_pca

**Visualizaciones:**

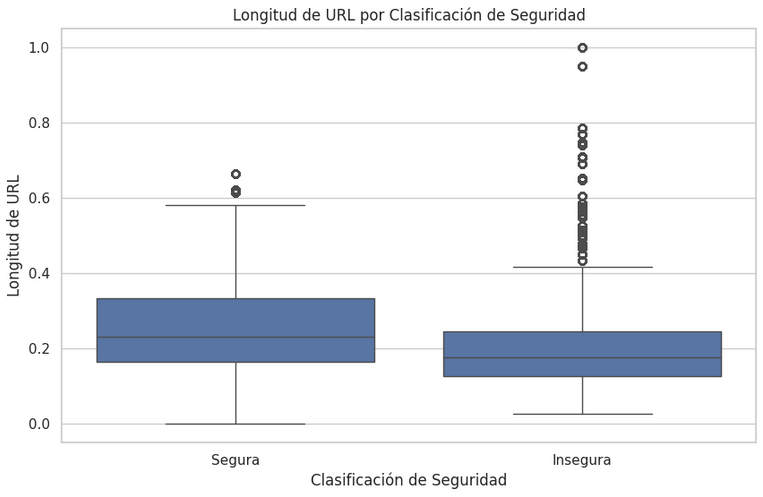
**Histplot:** En este gráfico se puede observar que las URLs seguras tienden a concentrarse en una longitud más específica, con una disminución gradual en frecuencia. Mientras las URLs inseguras, son menos frecuentes y muestran una ligera concentración a comparación con las URL seguras y tienen algunos picos dispersos en longitud.

Las URLs más frecuentes son las seguras con una distribución más clara y concentrada, mientras que las inseguras son menos frecuentes y están más dispersas.



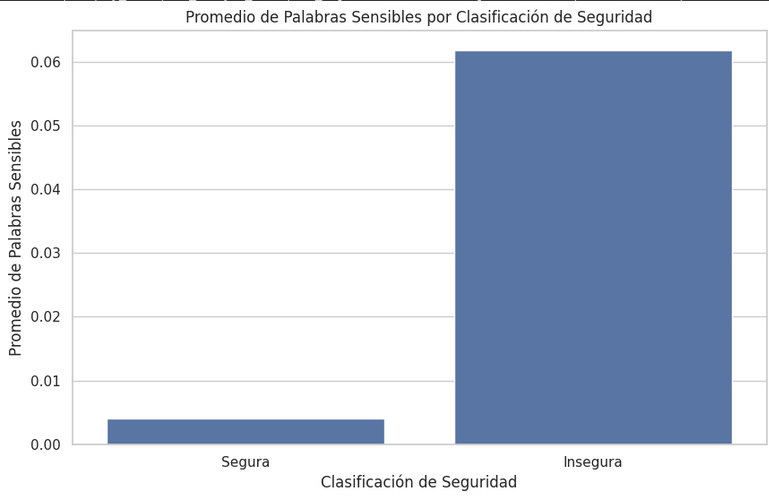
**Boxplot:**

Las URLs seguras tienen una longitud más concentrada y menos variable . Mientras que las URLs inseguras muestran una gran dispersión en la longitud son más variables, tiene más outliers que se extienden a longitudes mayores, indicando una cantidad significativa de URLs más largas.



**Barplot:**

Las URLs inseguras tienen una mayor tendencia a incluir palabras sensibles en comparación a la clase segura.



### 

### **Resumen y Conclusiones Finales**

**Hipótesis 1 : Impacto del Contenido y Origen de las URL en su Seguridad**

La hipótesis es soportada ya que las URLs que contienen palabras sensibles son más propensas a ser clasificadas como inseguras.

**Hipótesis 2 : Relación entre Longitud y Estructura de las URLs con su Seguridad**

La hipótesis es soportada ya que las URLs inseguras tienden a ser más largas y más dispersas.

**Hipótesis 3 : Influencia de Variables Externas en la Seguridad de las URLs**

Hipótesis soportada dado que la presencia de HTTPS y de palabras sensibles son factores que influyen en la seguridad percibida de una URL.

# Validation:

Red Neuronal :

Utilice Stratified K-fold Cross validation para la evaluación del modelo, definiendo un umbral de clasificación =5 para poder convertir las puntuaciones de seguridad en clases binarias. Esto me permitió un entrenamiento del modelo de red neuronal en cada iteración de la validación cruzada. Por último se calcula la desviación y media. Como **resultado** se muestra una precisión del 100%, indicando la correcta clasificación del modelo en todas las muestras del conjunto de prueba. La precisión media también clasifica correctamente todas las muestras en todos los pliegues y también observamos que la desviación estándar es de 0.0, indicando que no hay variaciones en la precisión entre los pliegues

Si comparamos el modelo de red neuronal que realice anteriormente podemos observar que se obtuvo un buen rendimiento y al igual que este modelo de red neuronal, sin embargo este modelo utiliza validación cruzada, lo que nos proporciona una evaluación más robusta del modelo al dividir el conjunto de datos en múltiples pliegues y entrenar y evaluar cada uno de ellos, mientras que el segundo enfoque lo que hacía era dividir el conjunto de datos en entrenamiento y prueba, lo que puede dar resultados más susceptibles.

Bosques Aleatorios:

Se utilizaron 5 folds para evaluar el rendimiento del modelo de manera general y robusta. Los resultados de la validación cruzada muestran que el modelo de Bosques Aleatorios alcanzó un promedio de precisiones de todos los folds del 87.44%, lo que sugiere que el modelo tiene un buen rendimiento general en la capacidad de predecir la seguridad de las URLs en el conjunto de datos.

Estos resultados indican que el modelo tiene gran rendimiento en diferentes conjuntos de datos de entrenamiento y prueba, lo que aumenta la confianza en su capacidad general para datos no vistos y realizar predicciones precisas sobre la seguridad de las URLs.

Árbol de Decisión:

Se realizaron dos conjuntos de validaciones utilizando 5 folds cada uno. La precisión promedio para la primera validación fue del 87.65%. Mientras que para el segundo conjunto de validación cruzada fue del 87.44%.

Regresión Logística:

La precisión promedio del modelo de Regresión Logística en esta validación cruzada fue del 82.28%.La desviación estándar de la precisión fue de 0.00072, indicando que las precisiones obtenidas en los diferentes pliegues fueron muy consistentes entre sí.

En resumen: Para predecir la seguridad de las URLs, el modelo de **Bosques Aleatorios** parece ser la mejor opción debido a su alta precisión y consistencia en la validación cruzada, junto con un equilibrio adecuado entre la precisión y el recall. La precisión perfecta de la Red Neuronal debe ser analizada con cautela, asegurando que no esté sobre ajustado a los datos.

**CONCLUSIÓN ….**

Con respecto a las Hipótesis:

Estas observaciones, nos permiten tomar en cuenta la importancia de considerar la longitud, el contenido y el protocolo utilizado en las URLs para evaluar su seguridad. Estas conclusiones pueden ayudar a mejorar las prácticas de navegación y la construcción de sitios web seguros, contribuyendo a un entorno digital más seguro y confiable para todos.

Con respecto a los Modelos:

Dado que la predicción de la seguridad de las URL es crucial y se desea minimizar los errores, es esencial considerar modelos con alta precisión y bajo riesgo de errores. En este caso, los modelos de Bosques Aleatorios y Árboles de Decisión destacan por su excelente desempeño en términos de precisión y capacidad para manejar conjuntos de datos desequilibrados. Ambos modelos muestran una precisión del 99% en los resultados del modelo y una precisión promedio de alrededor del 87% en la validación cruzada, lo que sugiere una consistencia y robustez en sus predicciones. Además, estos modelos son más fáciles de interpretar y entender.