La regresión Lineal

Máster de ciberseguridad orientado a IAS

Ildefonso González Sánchez

Indice

Introducción

El documento contiene la solución asociada ala práctica propuesta de regresión linea en el módulo 2 del Máster de ciberseguridad orientado a IAS.

En esta nos centraremos en la creación de un Modelo de regresión lineal estudiando los conceptos de Machine Learning, donde el modelo desarrollado será capaz de aprender de un dataset de entrada para realizar sus propias predicciones.

Durante la práctica se explicará el desarrollo personal del alumno del diseño basado en modelo de IA predictiva de una API web desarrollada en python con Flask, que será capaz de predecir si una solicitud es lícita o maliciosa en función de sus features o atributos de entrada. La práctica solicita la creación de la API para le uso del modelo con al menos 2 endpoints:

```
- model_info
- score
```

El primero de los endpoints “`model_info`”, se debe de encargar de mostrar información interna del modelo.

El segundo de los endpoints “`score`”, se debe de encargar de predecir un resultado en función de una solicitud de entrada

Descripción del modelo

El modelo regresión lineal, está basado en un método estadístico de aprendizaje automatizado. Este método matemático trata de modelar la relación entra una variable dependiente en nuestro caso será la Y con una o varias variables independientes en nuestro caso el array X[].

Donde Y representará el valor a predecir o resultado de la la operación:

```
valor en 0-1
¿Los valores de entrada que me acaban de proporcionar se trata de un ataque?
```

Donde X representarán los valores de entrada en cada petición

```
Matriz de valores numéricos, que representan cada feature []
```

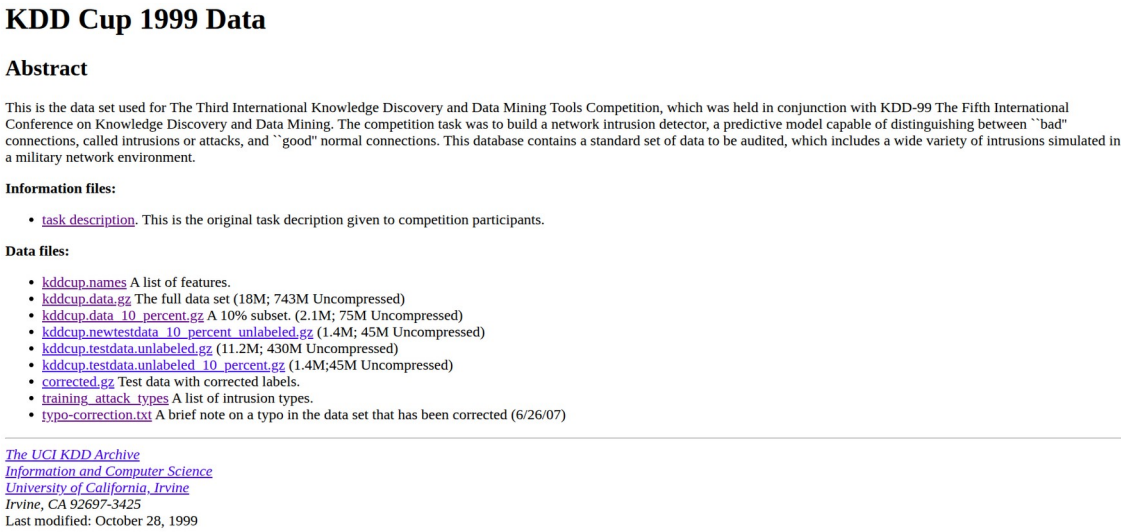
El objetivo final del modelo estadístico es encontrar una línea que describa lo mejor posible cómo cambia la variable objetivo Y en función de las variables explicativas X[].

Para ello, estima una función lineal que minimiza la diferencia entre los valores observados y los valores predichos.

Datos utilizados

Para la práctica nos han propuesto el uso del siguiente Dataset

https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html



KDD Cup 1999 Data

Abstract

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

Information files:

- task description. This is the original task decription given to competition participants.

Data files:

- kddcup.names A list of features.
- kddcup.data.gz The full data set (18M; 743M Uncompressed)
- kddcup.data_10_percent.gz A 10% subset. (2.1M; 75M Uncompressed)
- kddcup.newtestdata_10_percent_unlabeled.gz (1.4M; 45M Uncompressed)
- kddcup.testdata.unlabeled.gz (11.2M; 430M Uncompressed)
- kddcup.testdata.unlabeled_10_percent.gz (1.4M;45M Uncompressed)
- corrected.gz Test data with corrected labels.
- training_attack_types A list of intrusion types.
- typo-correction.txt A brief note on a typo in the data set that has been corrected (6/26/07)

The UCI KDD Archive
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
Last modified: October 28, 1999

Dataset

Dado que no es permitido el uso de BD, para el almacenamiento se utilizará el de 10% de los datos. Este dataset que utilizaremos de entrenamiento contiene 494022 lineas, incluyendo las cabeceras.

Variables del modelo

Definiremos como variables del modelo cada una de las columnas que contiene estos datos, estas son conocidas como features o características del Dataset, que a la hora de implementar el modelo de regresión lineal serán X e Y:

Los valores del Dataset que asociaremos a nuestro array de variables independientes X[]:

```
duration
protocol_type
service
flag
src_bytes
dst_bytes
```

```
land
wrong_fragment
urgent
hot
num_failed_logins
logged_in
num_compromised
root_shell
su_attempted
num_root
num_file_creations
num_shells
num_access_files
num_outbound_cmds
is_host_login
is_guest_login
count
srv_count
serror_rate
srv_serror_rate
rerror_rate
srv_rerror_rate
same_srv_rate
diff_srv_rate
srv_diff_host_rate
dst_host_count
dst_host_srv_count
dst_host_same_srv_rate
dst_host_diff_srv_rate
dst_host_same_src_port_rate
dst_host_srv_diff_host_rate
dst_host_serror_rate
dst_host_srv_serror_rate
dst_host_rerror_rate
dst_host_srv_rerror_rate
```

Los valores del Dataset que asociaremos a nuestra variable dependiente Y, sera:

```
attack_type
```

El cual reemplazaremos por attack_number, en función de si el valor que contiene es un 1 o un 0:

```
request_data['attack_num'] = np.where(request_data['attack_type'].str.strip() ==
"normal.", 0, 1)
```

Implementación del modelo

El modelo ha sido desarrollado en python3 con una API de invocación en Flask.

Por lo tanto los requisitos mínimos para la ejecución del proyecto, serán:

```
python --version
Python 3.12.3
```

librerías requeridas:

```
pyclbr
pandas
numpy
matplotlib
seaborn
sklearn
Flask
json
```

Estas librerías, deben de estar instaladas como prerequisitos en el sistema, se recomienda el uso de venv.

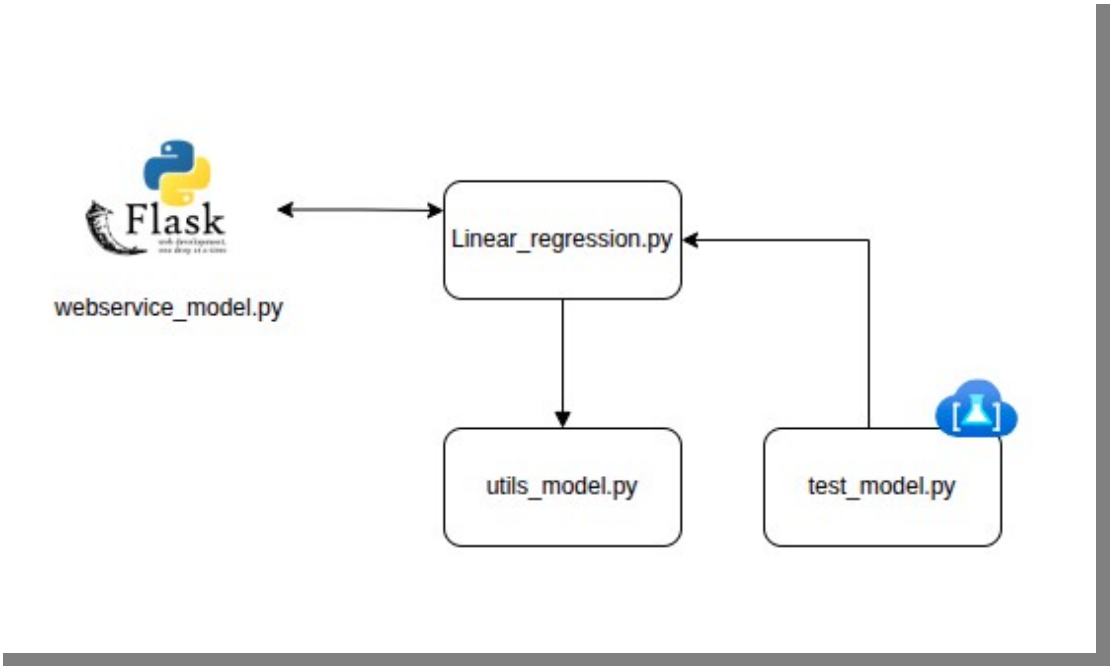
La invocación al main del proyecto, se realiza desde el fichero webservice_model.py:

```
~/MASTER_IAS/Modulo2/modelo$ python webservice_model.py
        predictions:  [ 0.99705835  0.99705835  0.99705835 ... -0.0269059  -
0.00659387
  0.01044942]

=================== Model output results: 0.9 ===============
        MAE: 0.08092916658681386
        MSE: 0.026283365426975236
        RMSE: 0.16212145270436987
        R^2 0.7726434132759052
/home/f0ns1/MASTER_IAS/Modulo2/.venv/lib/python3.12/site-packages/statsmodels/
graphics/gofplots.py:1041: UserWarning: color is redundantly defined by the 'color'
keyword argument and the fmt string "b" (-> color=(0.0, 0.0, 1.0, 1)). The keyword
argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
 * Serving Flask app 'webservice_model'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Diseño del software

El modelo proporcionado tiene tres clases fundamentales que se definen en el siguiente diagrama:



`webservice_model.py`

La función principal de esta clase, es la de exponer los dos servicios requeridos en la práctica como endpoints de la API Rest.

El primer endpoint de tipo GET /api/model_info, no tiene parámetros de entrada y se encargará de devolver los datos y métricas mas relevantes del modelo

El segundo endpoint es de tipo POST /api/score, tiene como parámetros de entrada un json que proporciona las características del modelo a evaluar devolviendo una puntuación que valora o predice si se trata de una solicitud maliciosa o legítima.

`linear_regression.py`

Este fichero, contiene la clase de regresión lineal y el grueso de la lógica de la práctica. Se encarga de crear el modelo, entrenarlo y realizar las predicciones.

`utils_model.py`

El fichero contiene los métodos auxiliares utilizados, tanto por el modelo como pro la API para la validación de datos o parámetros de entrada, normalización de objetos y creación de gráficas de los resultados del modelo.

`test_model.py`

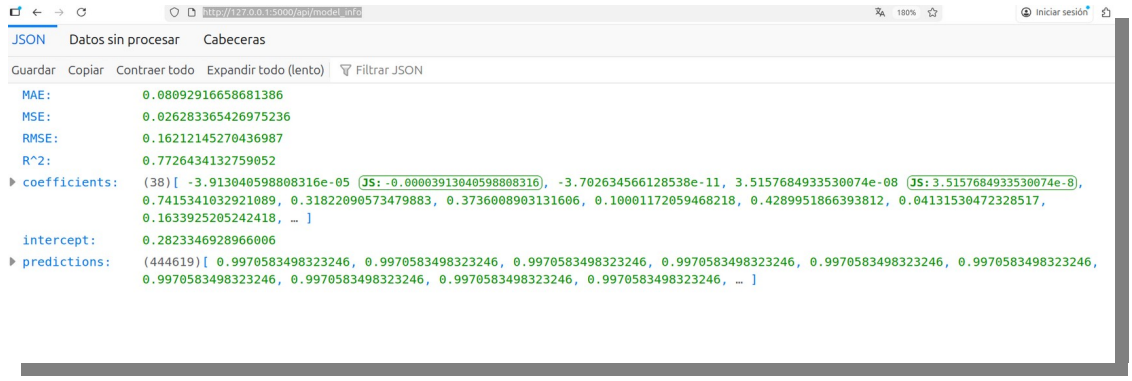
No forma parte de la solución, es utilizado para realizar pruebas sobre el modelo, verificar funcionamiento, extracción de gráficas adicionales, etc.

Consumo de la API model-info

El consumo de la API model_info puede realizarse desde el navegador en la siguiente url:

`http://127.0.0.1:5000/api/model_info`

Una respuesta de ejemplo será la siguiente:



Consumo de la API score

Para el completo funcionamiento de la API score se han creado dos archivos.

El primero tiene los parámetros de entrada para la invocación que devuelva una predicción de ataque:

```
{
  "duration": 0,
  "src_bytes": 1032,
  "dst_bytes": 0,
  "land": 0,
  "wrong_fragment": 0,
  "urgent": 0,
```

```
  "hot": 0,
  "num_failed_logins": 0,
  "logged_in": 0,
  "num_compromised": 0,
  "root_shell": 0,
  "su_attempted": 0,
  "num_root": 0,
  "num_file_creations": 0,
  "num_shells": 0,
  "num_access_files": 0,
  "num_outbound_cmds": 0,
  "is_host_login": 0,
  "is_guest_login": 0,
  "count": 511,
  "srv_count": 511,
  "serror_rate": 0.00,
  "srv_serror_rate": 0.00,
  "rerror_rate": 0.00,
  "srv_rerror_rate": 0.00,
  "same_srv_rate": 1.00,
  "diff_srv_rate": 0.00,
  "srv_diff_host_rate": 0.00,
  "dst_host_count": 255,
  "dst_host_srv_count": 255,
  "dst_host_same_srv_rate": 1.00,
  "dst_host_diff_srv_rate": 0.00,
  "dst_host_same_src_port_rate": 1.00,
  "dst_host_srv_diff_host_rate": 0.00,
  "dst_host_serror_rate": 0.00,
  "dst_host_srv_serror_rate": 0.00,
  "dst_host_rerror_rate": 0.00,
  "dst_host_srv_rerror_rate": 0.00
}
```

La invocación desde curl que verifica la predicción:

```
curl -X POST -H "Content-Type: application/json" http://127.0.0.1:5000/api/score -d
@MASTER_IAS/Modulo2/modelo/request_attack.txt
{
  "predction": [
    0.9970583498323246
  ],
  "result": "Attack_detected"
}
```

Es importante apreciar que el modelo, devuelto una predicción superior a 0.5 que el SW a considerado que se trata de un ataque. El segundo es contiene los datos para la predicción de una invocación legítima:

```
{
  "duration": 27,
  "src_bytes": 235,
  "dst_bytes": 1337,
  "land": 0,
  "wrong_fragment": 0,
  "urgent": 0,
```

```
  "hot": 0,
  "num_failed_logins": 0,
  "logged_in": 1,
  "num_compromised": 0,
  "root_shell": 0,
  "su_attempted": 0,
  "num_root": 0,
  "num_file_creations": 0,
  "num_shells": 0,
  "num_access_files": 0,
  "num_outbound_cmds": 0,
  "is_host_login": 0,
  "is_guest_login": 0,
  "count": 8,
  "srv_count": 8,
  "serror_rate": 0.00,
  "srv_serror_rate": 0.00,
  "rerror_rate": 0.00,
  "srv_rerror_rate": 0.00,
  "same_srv_rate": 1.00,
  "diff_srv_rate": 0.00,
  "srv_diff_host_rate": 0.00,
  "dst_host_count": 29,
  "dst_host_srv_count": 29,
  "dst_host_same_srv_rate": 1.00,
  "dst_host_diff_srv_rate": 0.00,
  "dst_host_same_src_port_rate": 0.03,
  "dst_host_srv_diff_host_rate": 0.00,
  "dst_host_serror_rate": 0.00,
  "dst_host_srv_serror_rate": 0.00,
  "dst_host_rerror_rate": 0.00,
  "dst_host_srv_rerror_rate": 0.00
}
```

La invocación que verifica una predicción legítima:

```
curl -X POST -H "Content-Type: application/json" http://127.0.0.1:5000/api/score -d
@MASTER_IAS/Modulo2/modelo/request_no_attack.txt
{
  "predction": [
    0.02169300335086516
  ],
  "result": "No_attack_detected"
}
```

Es importante apreciar que el modelo, devuelto una predicción inferior a 0.5 que el SW a considerado que se trata de una invocación legítima.

Evaluación del modelo

La evaluación del modelo se analiza a partir de dos métodos principales:

- Por un lado, las variables de métricas que se han obtenido tras entrenar el modelo.
- Por otro lado, las gráficas que se han creado tras entrenar el modelo.

Variables:

Definición esquemática de las variables:

Métrica	Significado	Ideal	Sensibilidad a Outliers
MAE	Error promedio	Bajo (cerca de 0)	Baja
MSE	Error cuadrático	Bajo	Alta
RMSE	Error cuadrático en mismas unidades	Bajo	Alta
R^2	coeficiente de determinación	Cercano a 1	Media

```
Variables obtenidas:
```


```
 "MAE": 0.005392411134654541,
 "MSE": 0.0017985599209257427,
 "RMSE": 0.04240943198070145,
 "R^2": 0.905903384687803,
 "intercept": 0.8558998716641331,
```


```
Descripción y características principales para la interpretación de
las variables del modelo:
```

MAE: (Mean Absolute Error) Error promedio absoluto:

El cálculo en el código fuente ha sido el siguiente:

```
MAE = metrics.mean_absolute_error(y_test, predictions)
```

- Calcula la diferencia absoluta entre cada valor real y cada predicción

- Suma todas esas diferencias
- Las divide entre la cantidad de muestras

Resultado: un valor positivo que indica cuánto se equivoca el modelo en promedio.

MSE: (Mean Squared Error) Error cuadrático medio:

El cálculo en el código fuente ha sido el siguiente:

```
MSE = metrics.mean_squared_error(y_test, predictions)
```

- utiliza los valores reales
- utiliza las predicciones del modelo
- Calcula la diferencia para cada valor
- Eleva esas diferencias al cuadrado

Resultado: cálculo del promedio

RMSE: (Root Mean Squared Error) Raíz del error cuadrático medio:

El cálculo en el código fuente ha sido el siguiente:

```
RMSE = np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

- Calcula el MSE entre valores reales y predicciones
- Toma la raíz cuadrada

Resultado: Si RMSE es un valor bajo el modelo es muy preciso, por le contrario si es un valor bajo errores grandes frecuentes

R^2: El valor adecuado para el coeficiente de determinación:

El cálculo en el código fuente ha sido el siguiente:

```
R² = self.model.score(X_test, y_test)
```

- utiliza los valores de entrenamiento X
- utiliza los valores de entrenamiento Y

Valor de R^2	Interpretación
1.0	El modelo predice perfecto
0.9 – 1.0	Ajuste excelente
0.7 – 0.9	Muy buen ajuste
0.5 – 0.7	Aceptable
0.2 – 0.5	Débil
0.0	No hay relación
< 0	El modelo es peor que predecir el promedio

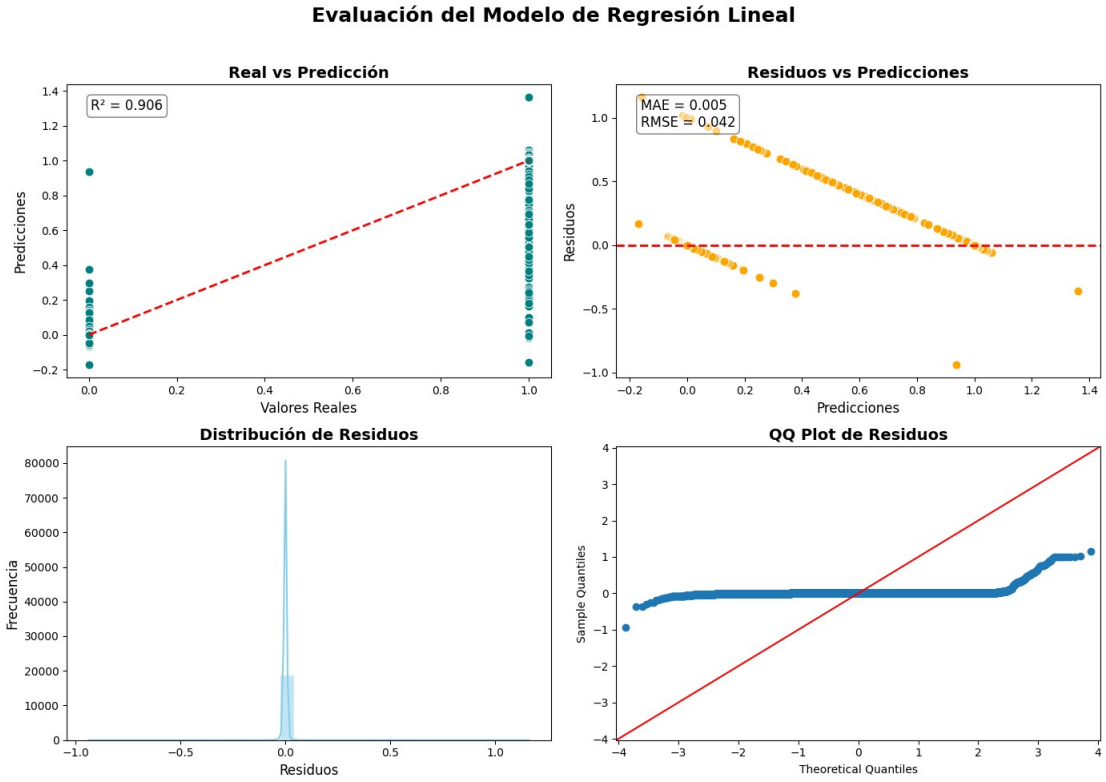
Términos que se deben entender de los valores utilizados en el modelo:

Término	Significado
Intercept	Valor de Y cuando todas las X = 0
Coeficientes	Cuánto cambia Y por cada unidad que cambia X

Gráficas

Las gráficas seleccionadas para determinar el modelo son las siguientes:

En el código fuente se puede apreciar como se han implementado las 4 gráficas en una matriz de gráficas 2x2 que ayuda compactar el resultado.



Real vs predicción

La gráfica Real vs Predicción compara los valores reales de la variable objetivo con los valores predichos por el modelo de regresión lineal. En esta representación:

- El eje X muestra los valores reales.
- El eje Y muestra las predicciones del modelo.

La línea diagonal representa un ajuste perfecto (predicción = valor real).

Si los puntos se alinean cerca de la línea diagonal, significa que el modelo realiza buenas predicciones.

residuos vs predicciones

La gráfica Residuos vs Predicciones muestra:

- en el eje Y los residuos (residuo = valor real – predicción)
- en el eje X las predicciones del modelo.

Su función principal es evaluar si el modelo lineal cumple los supuestos estadísticos y si el ajuste es adecuado.

distribución de residuos

La gráfica de distribución de residuos, se trata de un histograma que permite evaluar si los residuos del modelo siguen una distribución aproximadamente normal, uno de los supuestos fundamentales de la regresión lineal.

qq-plot-de-residuos

El QQ plot (Quantile–Quantile plot) compara los cuantiles de los residuos del modelo con los cuantiles de una distribución normal teórica.

Es una de las herramientas más importantes para evaluar si los residuos cumplen el supuesto de normalidad, esencial en regresión lineal clásica.

Interpretación de resultados

Una vez entendido el modo en el que se va a evaluar el modelo , es necesario interpretar los datos obtenidos en el proceso de la evaluación para llegar a tomar conclusiones sobre las predicciones realizadas

Variables

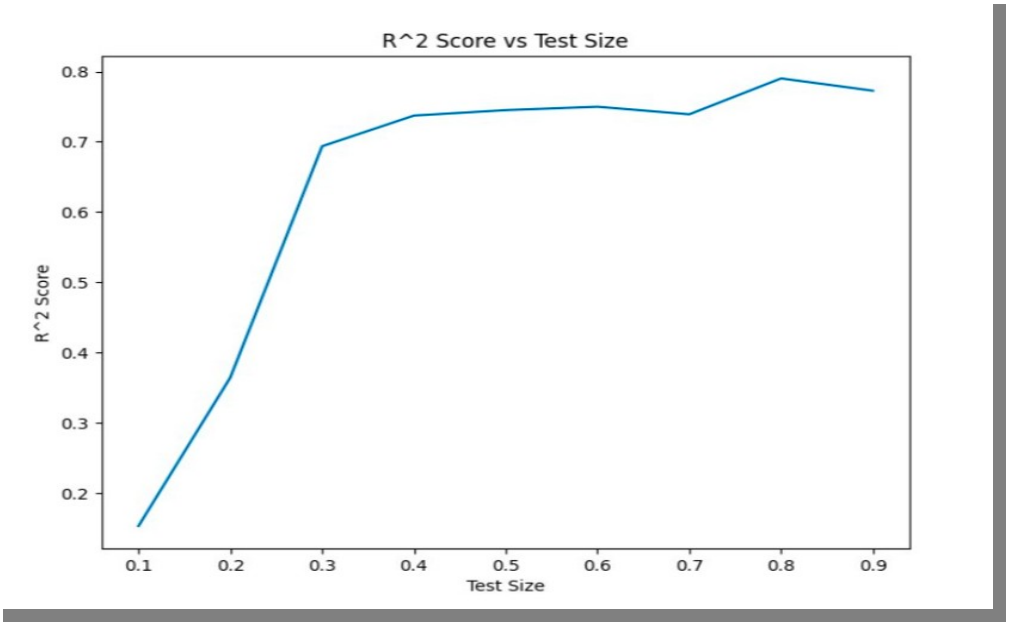
En términos generales la interpretación de los valores después de los ajustes, es bastante buena.

Métrica	Valor	Interpretación
MAE	0.00539	El error absoluto promedio es extremadamente bajo. El modelo predice con gran precisión, cometiendo en promedio un error de solo 0.005 unidades.
MSE	0.00180	El error cuadrático medio es muy pequeño; indica que los errores grandes son muy poco frecuentes.
RMSE	0.04241	El modelo tiene un error estándar del 4.2 %, lo cual es excelente para la mayoría de aplicaciones.
R^2	0.9059	El modelo explica el **90.59 % de la variabilidad** de la variable objetivo. Indica un ajuste muy alto y buen poder predictivo.
Intercepto	0.8559	Cuando todas las variables predictoras valen cero, la predicción base del modelo es ~0.856.

Ajustes realizados para la mejora del modelo:

- Modificación de el dataset utilizado
  - Se eliminan las features de tipo string:
    - protocol
    - service

        - flag
    - Se unifica el tipo de ataque en una sola feature binaria attack_num:
        - Error
        - No error
    - Se filtran los tipos de ataques del dataset, utilizando solo los de la misma tipología, que a priori tienen valores muchos mas comunes y los errores dejan de ser grandes y frecuentes
- Se evalúa el test_size del modelo determinando que porcentaje es el mejor para la tupla entrenamiento, predicción del total de los datos:



Gráficas

Bien es cierto que para al correcto entendimiento de las gráficas se han utilizado IAs para que dieran sus puntos de vista, ya que lo considero muy enriquecedor a la hora de entender la teoría matemática, a continuación presento un resumen personal sobre los puntos mas relevantes de cada una de las gráficas:

Gráfica “Real vs Predicción”

Esta gráfica compara los valores reales de la variable objetivo con los valores predichos por el modelo,muestra un alto nivel de ajuste.

- Indica que aproximadamente el 90.6% de la variabilidad de los valores reales es explicada por el modelo lineal.
- Pero se observa que muchos valores reales están concentrados en extremos (0 y 1), mientras que las predicciones presentan cierta dispersión alrededor de esos puntos.

Gráfica “Residuos vs Predicciones”

Los residuos no parecen distribuirse aleatoriamente alrededor de cero:

- Se observa una tendencia descendente de los residuos conforme aumentan las predicciones.
- Esto indica un patrón sistemático, lo que sugiere que el modelo podría estar incurriendo en sesgo, posiblemente debido a una relación no estrictamente lineal o a la falta de variables explicativas relevantes.

Las métricas MAE = 0.005 y RMSE = 0.042 confirman que el error medio es reducido, pero el patrón visible indica que los errores no son puramente aleatorios.

Gráfica “Distribución de Residuos”

Definido como un histograma debería tener una forma normal o estar normalizada, peor no es exactamente así:

- Los residuos se encuentran altamente concentrados alrededor de cero, con un pico muy pronunciado.
- Esto se debe a que la variable objetivo está prácticamente binarizada (valores próximos a 0 y 1), lo cual genera residuos muy pequeños o muy específicos.
- La distribución no es exactamente normal, sino asimétrica y fuertemente concentrada, lo que refleja que el modelo trabaja con una variable de naturaleza casi categórica, lo cual afecta la forma del error.

Gráfica “QQ Plot de Residuos”

El QQ plot compara los cuantiles de los residuos con los cuantiles de una distribución normal teórica.

Visualmente se observa que:

- Los puntos se separan significativamente de la línea roja en las colas.
- Esto confirma que los residuos no siguen una distribución normal, especialmente en valores extremos.

Esta desviación suele ocurrir porque:

- Existen valores atípicos o límites estructurales (como 0 y 1).

conclusiones

La conclusión es que aunque el modelo funciona con una precisión buena y como aproximación inicial está bien, no es suficiente debido al siguiente motivo en esencia.

La variable Y que es la variable objetiva del modelo en función de X la hemos simplificado como variable binaria (0|1), lo que quier decir error o no error.

La realidad, es que esto es solo un modo de simplificar el dataset con el que se trabaja, para poder adaptarlo al modelo de regresión lineal.

Y no debería ser este el modelo a utilizar para analizar estos datos, pudiendo ser empleados alguno de los siguientes:

- Modelos no lineales,
- Transformaciones de variables,
- O incluso modelos diseñados para datos binarios o proporciones (como regresión logística o beta regresión), si corresponde.

Basándome en el siguiente script en bash, se puede determinar el numero de ataques diferentes que se identifican para cada servicio detectado en el dataset:

```
for i in `cat /home/f0ns1/MASTER_IAS/Modulo2/kddcup.data_10_percent | awk -F "," '{print $3}' | sort -u`; do

NORMAL=`cat /home/f0ns1/MASTER_IAS/Modulo2/kddcup.data_10_percent | grep $i | grep normal | wc -l`;

ATTACK=`cat /home/f0ns1/MASTER_IAS/Modulo2/kddcup.data_10_percent | grep $i | grep -v normal | wc -l`;

echo "Service $i normal: $NORMAL attack: $ATTACK";

ATTACK_TYPES=`cat /home/f0ns1/MASTER_IAS/Modulo2/kddcup.data_10_percent | grep $i | grep -v normal | awk -F "," '{print $42}' | sort -u`;

echo "          $i ---> $ATTACK_TYPES";

done
```

Output:

```
Service auth normal: 220 attack: 108

auth ---> neptune.
Service bgp normal: 0 attack: 106
bgp ---> neptune.
Service courier normal: 0 attack: 108
courier ---> neptune.
portsweep.
Service csnet_ns normal: 0 attack: 126
csnet_ns ---> neptune.
portsweep.
Service ctf normal: 0 attack: 97
ctf ---> neptune.
nmap.
Service daytime normal: 0 attack: 103
daytime ---> neptune.
portsweep.
Service discard normal: 0 attack: 116
discard ---> neptune.
satan.
Service domain normal: 5865 attack: 114
domain ---> ipsweep.
neptune.
satan.
Service domain_u normal: 5862 attack: 1
domain_u ---> satan.
Service echo normal: 0 attack: 112
echo ---> neptune.
portsweep.
Service eco_i normal: 389 attack: 1253
eco_i ---> ipsweep.
nmap.
satan.
Service ecr_i normal: 345 attack: 281055
ecr_i ---> ipsweep.
nmap.
pod.
```

```
portsweep.
smurf.
Service efs normal: 0 attack: 103
efs ---> neptune.
portsweep.
Service exec normal: 0 attack: 99
exec ---> neptune.
Service finger normal: 468 attack: 202
finger ---> ipsweep.
land.
neptune.
portsweep.
satan.
Service ftp normal: 4172 attack: 1350
ftp ---> buffer_overflow.
ftp_write.
ipsweep.
loadmodule.
multihop.
neptune.
portsweep.
rootkit.
satan.
warezclient.
warezmaster.
Service ftp_data normal: 3798 attack: 923
ftp_data ---> buffer_overflow.
ftp_write.
ipsweep.
loadmodule.
multihop.
neptune.
portsweep.
rootkit.
satan.
warezclient.
warezmaster.
Service gopher normal: 0 attack: 117
gopher ---> ipsweep.
neptune.
```

```
portsweep.
satan.
Service hostnames normal: 0 attack: 104
hostnames ---> neptune.
portsweep.
satan.
Service http normal: 61886 attack: 2506
http ---> back.
ipsweep.
neptune.
phf.
portsweep.
satan.
Service http_443 normal: 0 attack: 99
http_443 ---> neptune.
portsweep.
Service imap4 normal: 0 attack: 117
imap4 ---> imap.
neptune.
Service IRC normal: 42 attack: 1
IRC ---> satan.
Service iso_tsap normal: 0 attack: 115
iso_tsap ---> neptune.
Service klogin normal: 0 attack: 106
klogin ---> neptune.
Service kshell normal: 0 attack: 98
kshell ---> neptune.
Service ldap normal: 0 attack: 101
ldap ---> neptune.
Service link normal: 0 attack: 102
link ---> ipsweep.
neptune.
portsweep.
Service login normal: 0 attack: 210
login ---> ftp_write.
neptune.
Service mtp normal: 9598 attack: 232
mtp ---> ipsweep.
neptune.
portsweep.
satan.
Service name normal: 0 attack: 202
```

```
name ---> ipsweep.
neptune.
portsweep.
satan.
Service netbios_dgm normal: 0 attack: 99
netbios_dgm ---> neptune.
Service netbios_ns normal: 0 attack: 102
netbios_ns ---> neptune.
portsweep.
Service netbios_ssn normal: 0 attack: 107
netbios_ssn ---> neptune.
portsweep.
Service netstat normal: 0 attack: 95
netstat ---> neptune.
portsweep.
satan.
Service nnsp normal: 0 attack: 105
nnsp ---> neptune.
Service nntp normal: 0 attack: 108
nntp ---> neptune.
nmap.
satan.
Service ntp_u normal: 380 attack: 0
ntp_u --->
Service other normal: 5632 attack: 1605
other ---> neptune.
portsweep.
rootkit.
satan.
warezclient.
Service pm_dump normal: 0 attack: 1
pm_dump ---> satan.
Service pop_2 normal: 0 attack: 101
pop_2 ---> neptune.
Service pop_3 normal: 79 attack: 123
pop_3 ---> neptune.
nmap.
portsweep.
satan.
Service printer normal: 0 attack: 109
```

```
printer ---> neptune.
portsweep.
Service private normal: 7366 attack: 103527
private ---> ipsweep.
neptune.
nmap.
portsweep.
satan.
teardrop.
Service red_i normal: 1 attack: 0
red_i --->
Service remote_job normal: 0 attack: 120
remote_job ---> ipsweep.
neptune.
portsweep.
Service rje normal: 0 attack: 111
rje ---> ipsweep.
neptune.
portsweep.
Service shell normal: 1 attack: 209
shell ---> neptune.
Service smtp normal: 9598 attack: 125
smtp ---> ipsweep.
neptune.
portsweep.
satan.
Service sql_net normal: 0 attack: 110
sql_net ---> neptune.
portsweep.
Service ssh normal: 1 attack: 104
ssh ---> ipsweep.
neptune.
portsweep.
Service sunrpc normal: 0 attack: 107
sunrpc ---> neptune.
portsweep.
satan.
Service supdup normal: 0 attack: 105
supdup ---> neptune.
portsweep.
Service systat normal: 0 attack: 115
systat ---> neptune.
```

```
portsweep.
Service telnet normal: 219 attack: 294
telnet ---> buffer_overflow.
guess_passwd.
ipsweep.
land.
loadmodule.
multihop.
neptune.
nmap.
perl.
portsweep.
rootkit.
satan.
spy.
Service tftp_u normal: 1 attack: 0
tftp_u --->
Service time normal: 52 attack: 208
time ---> ipsweep.
neptune.
portsweep.
Service tim_i normal: 2 attack: 5
tim_i ---> pod.
Service urh_i normal: 14 attack: 0
urh_i --->
Service urp_i normal: 537 attack: 1
urp_i ---> satan.
Service uucp normal: 0 attack: 212
uucp ---> neptune.
portsweep.
satan.
Service uucp_path normal: 0 attack: 106
uucp_path ---> neptune.
portsweep.
Service vmnet normal: 0 attack: 106
vmnet ---> neptune.
portsweep.
satan.
Service whois normal: 0 attack: 110
whois ---> ipsweep.
```

```
neptune.
portsweep.
Service X11 normal: 9 attack: 2
X11 ---> satan.
Service Z39_50 normal: 0 attack: 92
Z39_50 ---> neptune.
portsweep.
```