Práctica de Azure

OPTIMIZACIÓN DE GRANDES VOLÚMENES DE DATOS
2023/2024

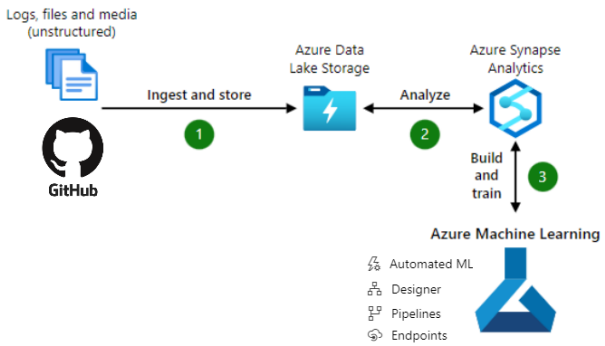
Carlos Oliva López y Christian Graf Aray
MÁSTER DE APRENDIZAJE AUTOMÁTICO Y DATOS MASIVOS

Contenido

1. Introducción

En esta práctica analizaremos las posibilidades y ventajas que nos aportan los servicios de Azure como Synapse Analytics y ML Studio. Para probar nuestro punto, utilizaremos estas dos herramientas con un dataset que recoge más de 10.000 enlaces a diferentes endpoints y sitios web, clasificándolos como phishing o legítimos. En total, por cada uno de esos enlaces se recogen 87 features, que van desde el propio enlace (la URL) hasta el número de ciertos tipos de caracteres especiales. Al ser un dataset de clasificación, la etiqueta a predecir será la columna "status".

Por lo tanto, en la primera parte de la práctica utilizaremos Azure Synapse Analytics para realizar un análisis exploratorio y tratamiento de los datos anteriormente descritos. Una vez guardado esos datos curados, los utilizaremos en Azure ML Studio para realizar su clasificación, examinando las distintas características que nos ofrece Azure. El diagrama de la arquitectura final quedará así:



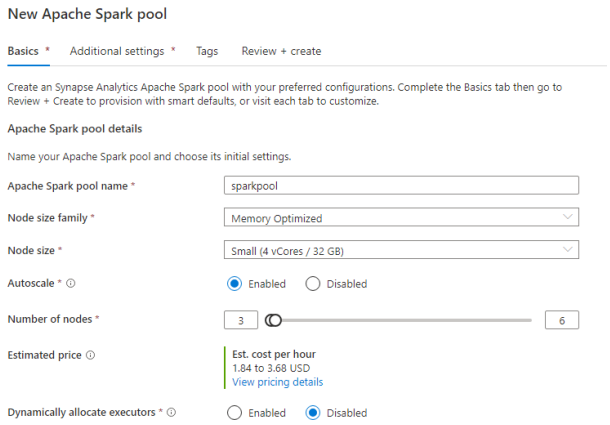
El repositorio del grupo se puede encontrar aquí, y se han dado permisos a la suscripción y los workspaces creados para que se puedan consultar los enlaces que van apareciendo a lo largo de la memoria.

2. Azure Synapse Analytics

En primer lugar, crearemos el workspace de Synapse Analytics que nos permitirá realizar el EDA y tratamiento de los datos.

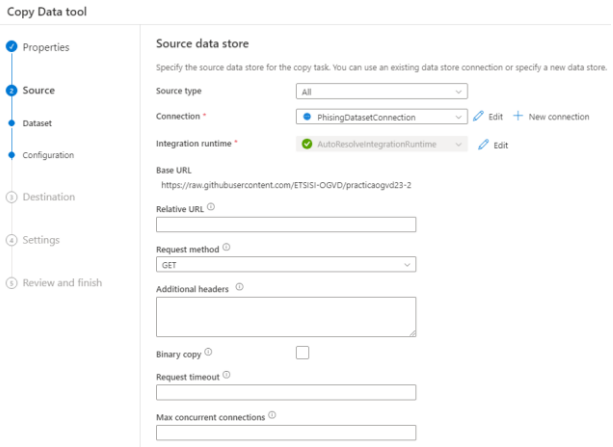
2.1. Creación del pool de Spark

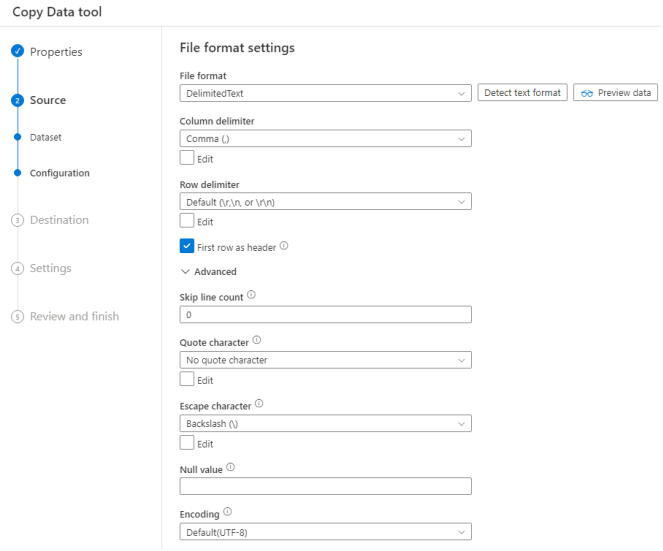
Para poder trabajar en el workspace, empezaremos creando un pool de Spark que nos permitirá lanzar notebooks utilizando PySpark. Debido a que solo se necesita tratar un dataset pequeño, escogemos pocos vCores como máximo y de la familia "Memory Optimized" para ahorrar costes.



2.2. Ingesta de datos

Posteriormente, importaremos los datos con los que vamos a trabajar. Estos se han descargado de Kaggle y se han subido al repositorio de GitHub, situándose aquí. Los pasos para la ingesta se muestran a continuación. Para que funcione correctamente la ingesta mediante HTTP, es importante utilizar el enlace tipo raw de GitHub.





Preview data

Linked service: PhisingDatasetConnection

Object: https://raw.githubusercontent.com/ETSISI-OGVD/practicaogvd23-24-grupo_christiangraf_carlosoliva/main/dataset_l...

Preview | Schema

Prop_0	url
0	http://www.progarchives.com/album.asp?id=61737
1	http://signin.eday.co.uk.ws.edayisapi.dllsign.inusingsslpuseridcopartnerid2siteid.zdfxx949xyss1pnbh0soabfdzgdh2kpp
2	http://www.avevaconstruction.com/blesstool/image.htm
3	http://www.jp519.com/
4	https://www.velocidrone.com/
5	https://support-appleid.com.secureupdate.duilawyeryork.com/ap/b5aed586dda5d21/?cmd=_update&dispatch=b5aed586dda5d219f&locale=_US
6	https://www.authpro.com/auth/ubabankng/?action=reg
7	http://littlee.com.au/alibaba/login.alibaba.com.php
8	http://www.tutorialspoint.com/dbms/

Copy Data tool

Properties
Source
Destination
Dataset
Configuration
Settings
Review and finish

Destination data store

Specify the destination data store for the copy task. You can use an existing data store connection or specify a new data store.

Destination type: Azure Data Lake Storage Gen2
Connection *: synapse-practica-ws-WorkspaceDefa... Edit New connection
Integration runtime *: AutoResolveIntegrationRuntime Edit

Folder path
If the identity you use to access the data store only has permission to subdirectory instead of the entire account, specify the path to browse.
practica-fs/ Browse

File name: phising.csv

Copy behavior: Select...

Max concurrent connections:

Block size (MB):

Metadata
New

Copy Data tool

Properties
Source
Destination
Dataset
Configuration
Settings
Review and finish

File format settings

File format: DelimitedText

Column delimiter: Comma (,)
Edit

Row delimiter: Default (\r,\n, or \r\n)
Edit

Add header to file

Advanced

Quote character: No quote character
Edit

Escape character: Backslash (\)
Edit

Copy Data tool

Properties
Source
Destination
Settings
Review and finish

Settings

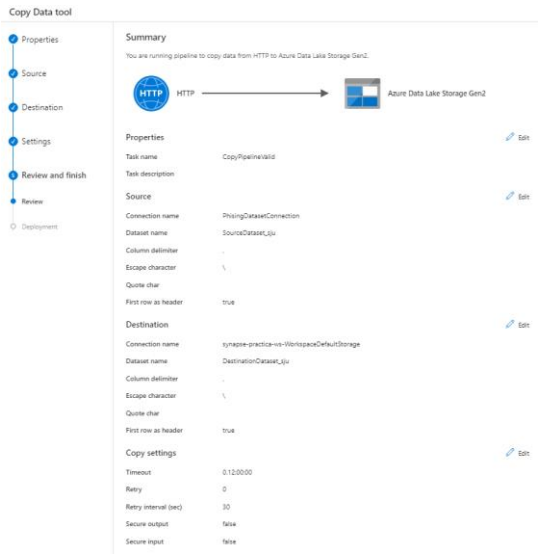
Enter name and description for the copy data task, more options for data movement

Task name *: CopyPipelineValid
Task description:
Fault tolerance:
Enable logging

Logging settings

Storage connection name *: synapse-practica-ws-WorkspaceDefa... Test connection Edit New
Integration runtime *: AutoResolveIntegrationRuntime Edit
Logging level: Info
Logging mode: Reliable Best effort
Folder path: practica-fs Browse
Enable staging

Advanced



2.3. Lectura y procesamiento de datos

Una vez creado el notebook y asignado el pool al que creamos en el apartado 2.1, podemos comenzar con el procesamiento. Se ha utilizado la función de spark.read para leer los datos previamente guardados:

```
1  path = "abfss://practica-fs@roberwido.dfs.core.windows.net/phishing.csv"
2  phishing_df = spark.read.option("header", "true").option("inferSchema", "true").csv(path)
```

✓ - Command executed in 31 sec 52 ms on 4:16:28 PM, 3/19/24

Utilizando el atributo dtypes podemos verificar que la mayoría de las columnas de los datos son ya numéricas

```
1  # vemos los tipos de datos cargados
2  phishing_df.dtypes
```

✓ - Command executed in 183 ms on 4:03:22 AM, 3/19/24

```
('domain_with_copyright', 'string'),
('whois_registered_domain', 'int'),
('domain_registration_length', 'int'),
('domain_age', 'int'),
('web_traffic', 'int'),
('dns_record', 'int'),
('google_index', 'int'),
('page_rank', 'int'),
('status', 'string')]
```

Las columnas “domain_with_copyright” y “status” son de tipo string, por lo que vamos a convertirlas en números enteros.

```
1  # Vamos a comprobar los valores únicos para la columna 'domain_with_copyright'
2  unique_values = phishing_df.select('domain_with_copyright').distinct().collect()
3  unique_values
```

[7] ✓ - Command executed in 1 sec 783 ms on 4:03:24 AM, 3/19/24

```
[Row(domain_with_copyright='One'),
 Row(domain_with_copyright='0'),
 Row(domain_with_copyright='one'),
 Row(domain_with_copyright='zero'),
 Row(domain_with_copyright='Zero'),
 Row(domain_with_copyright='1')]
```

Después de analizar todos los posibles valores de la columna, se crea una función y un UDF para transformar los datos.

```
# Como vemos que hay 4 tipos de valor para 1 y 0, transformamos la columna para que solo acepte valores enteros
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

def transform_domain_with_copyright(x):
    if x in ['one', 'One', '1']:
        return 1
    else:
        return 0

# Definir un UDF con la transformacion
transform_udf = udf(transform_domain_with_copyright, IntegerType())

phishing_df = phishing_df.withColumn('domain_with_copyright', transform_udf(phishing_df['domain_with_copyright']))
```
✓ - Command executed in 144 ms on 4:16:47 PM, 3/19/24

```
# confirmamos que la transformacion se ha hecho correctamente
unique_values = phishing_df.select('domain_with_copyright').distinct().collect()
unique_values
```
✓ - Command executed in 8 sec 532 ms on 4:03:33 AM, 3/19/24

```
[Row(domain_with_copyright=1), Row(domain_with_copyright=0)]
```

Se puede verificar que los valores de la columna son ahora 1 o 0. Luego hacemos lo mismo con la columna “status”:

```
# Comprobamos la columna que utilizaremos como label, ¿qué valores únicos tiene?
unique_labels = phishing_df.select('status').distinct().collect()
unique_labels
```
✓ - Command executed in 514 ms on 4:03:34 AM, 3/19/24

```
[Row(status='phishing'), Row(status='legitimate')]
```

```
# Como vemos que hay 4 tipos de valor para 1 y 0, transformamos la columna para que solo acepte valores enteros
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

def transform_label(x):
    if x == 'phishing':
        return 1
    else:
        return 0

# Definir un UDF con la transformacion
transform_udf_label = udf(transform_label, IntegerType())
phishing_df = phishing_df.withColumn('status', transform_udf_label(phishing_df['status']))
```
✓ - Command executed in 151 ms on 4:16:53 PM, 3/19/24

```
# confirmamos que la transformacion se ha hecho correctamente
unique_values = phishing_df.select('status').distinct().collect()
unique_values
```
✓ - Command executed in 506 ms on 4:03:35 AM, 3/19/24

```
[Row(status=1), Row(status=0)]
```

2.5. Visualización de los datos

Se generará una matriz de correlación para buscar columnas con un alto potencial de ser predictivas.

```
# Matriz de correlacion
correlation_matrix = df.corr()
correlation_matrix
```
✓ - Command executed in 497 ms on 4:03:40 AM, 3/19/24

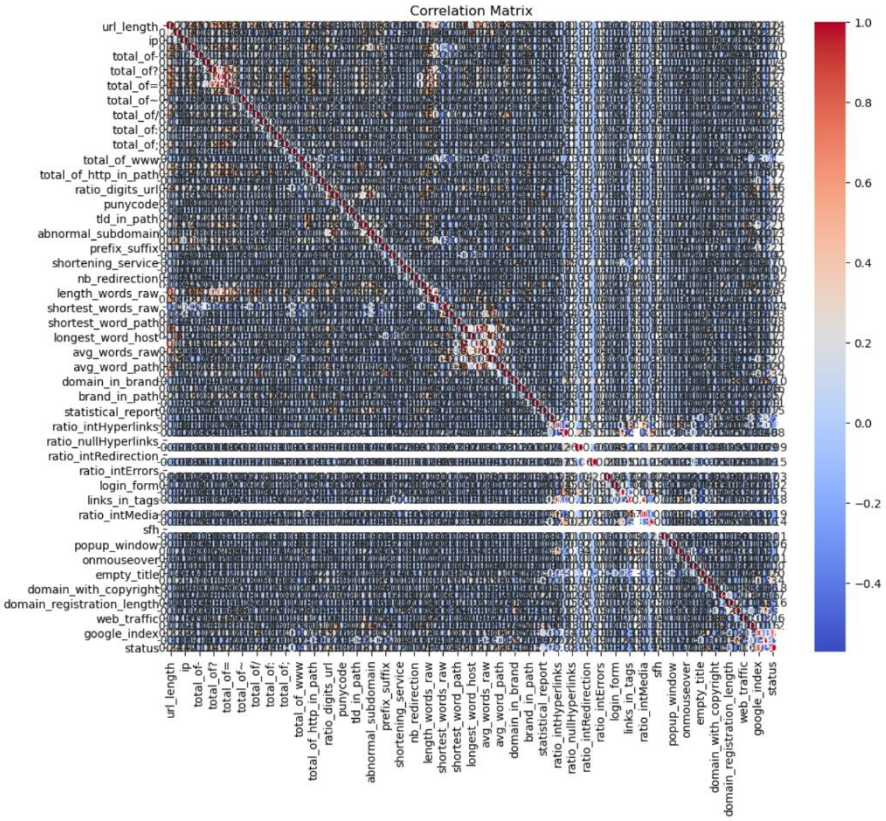
```
/tmp/ipykernel_7757/2805616672.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning
  correlation_matrix = df.corr()
```

	url_length	hostname_length	ip	total_of.	total_of-	total_of@	total_of?	total_of&	total_of=	total_of_	...	domain_in_title	domain_with_copyright	whois_registered_domain	domain_registration_length	domain_age	web_traffic	dns_record	google_index	page_rank	status
url_length	1.000000	0.217585	0.139063	0.447198	0.406951	0.140334	0.523172	0.587309	0.531574	0.443290	...	0.124224	-0.004201	-0.004760	0.032400	-0.006400	0.072773	0.024539	0.233061	-0.099900	0.244348
hostname_length	0.217585	1.000000	0.077150	0.406834	0.059480	0.070095	0.164129	0.142347	0.179735	0.080048	...	0.218850	0.073107	0.100543	0.047856	0.011504	0.161418	0.027895	0.216019	0.160621	0.240681
ip	0.139063	0.077150	1.000000	0.086433	0.034718	0.020823	0.127186	0.115839	0.134242	0.064511	...	0.045277	0.004624	0.018370	-0.033900	-0.024471	0.050773	0.041608	0.084218	-0.069010	0.098335
total_of.	0.447198	0.406834	0.086433	1.000000	0.049303	0.264505	0.353113	0.443665	0.423621	0.344767	...	0.108442	0.057120	0.060681	-0.029240	-0.010303	0.086857	0.132368	0.205764	-0.098752	0.205302
total_of-	0.406951	0.059480	0.034718	0.049303	1.000000	0.019011	0.053958	0.130734	0.115807	0.051177	...	0.009843	0.020914	0.036718	0.059414	0.080796	0.041968	0.031282	0.018283	0.104676	0.102849
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
web_traffic	0.072773	0.161418	0.050773	0.086857	0.041968	0.014132	0.142186	0.167480	0.185492	0.090583	...	0.082745	0.087961	0.025569	0.114610	0.059957	1.000000	0.011527	0.033618	0.211007	0.060531
dns_record	0.024539	-0.022895	0.041608	0.132368	-0.031282	0.031405	0.010299	0.079153	0.065150	0.061630	...	0.062821	0.042675	0.219171	-0.061376	-0.072481	-0.011527	1.000000	0.095822	-0.145565	0.123634
google_index	0.233061	0.216019	0.084218	0.200764	-0.010285	0.113746	0.202097	0.131298	0.169296	0.071718	...	0.265933	-0.144499	0.040005	-0.130972	-0.218101	0.033618	0.095822	1.000000	-0.386721	0.730684
page_rank	0.099900	0.160621	0.069010	0.098752	0.104676	0.065521	0.123151	0.128435	0.131230	0.008699	...	0.352742	0.067127	0.158069	0.350234	0.591275	0.211007	0.145563	0.386721	1.000000	0.509761
status	0.244348	0.240681	0.098335	0.205302	-0.102849	0.140987	0.293920	0.171001	0.233048	0.041028	...	0.339519	-0.175469	0.067720	-0.160737	-0.332586	0.060531	0.123634	0.730684	-0.509761	1.000000

85 rows × 85 columns

```
1  plt.figure(figsize=(12, 10))
2  sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
3  plt.title('Correlation Matrix')
4  plt.show()
```

[18] ✓ - Command executed in 8 sec 715 ms on 4:03:49 AM, 3/19/24

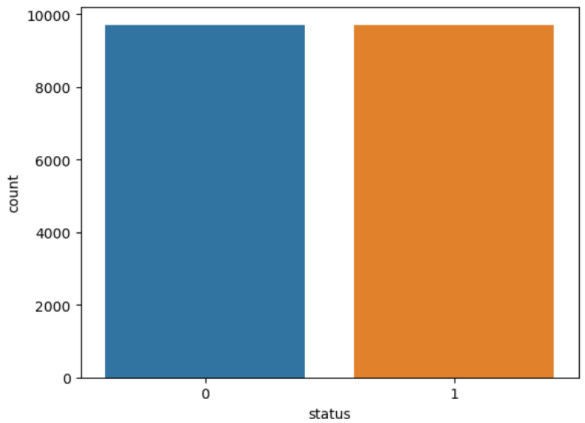


Podemos ver, por ejemplo, que la variable “google_index” tiene un alto nivel de correlación con la columna status. Esta columna será una de las que posteriormente se elegirán como las más relevantes en el siguiente apartado.

Podemos también verificar que tenemos un dataset balanceado pues hay un número similar de ejemplos para cada opción de “status”.

```
1  sns.countplot(data=df, x='status')
2  plt.show()
```

[22] ✓ - Command executed in 145 ms on 4:03:51 AM, 3/19/24



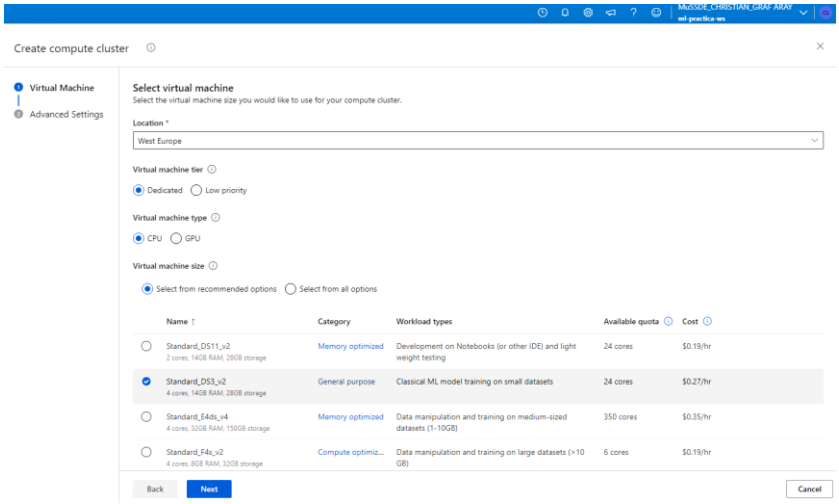
3. Azure ML Studio

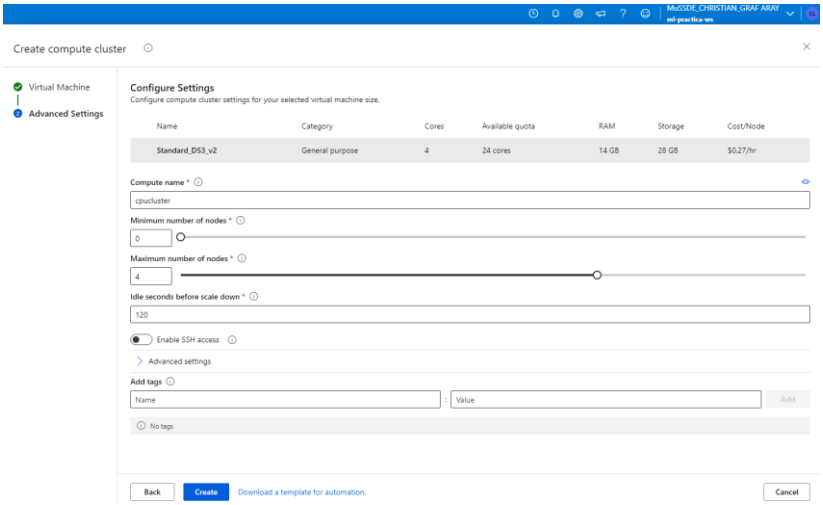
Como se comentaba en la introducción, la idea es entrenar el modelo para ver si es capaz de predecir si un enlace es phishing o no. Para explorar las características que nos ofrece Azure ML Studio, vamos a:

- Lanzar un Automated ML para determinar el mejor modelo para clasificar el dataset.
- Crear en Designer una Pipeline que encuentre los mejores parámetros para ese tipo de modelo, personalizando más el entrenamiento.
- Finalmente, con este último modelo, crear en Designer una Pipeline de inferencia y desplegarlo para ver como funcionaría si tuviéramos una API.

3.1. Creación del clúster

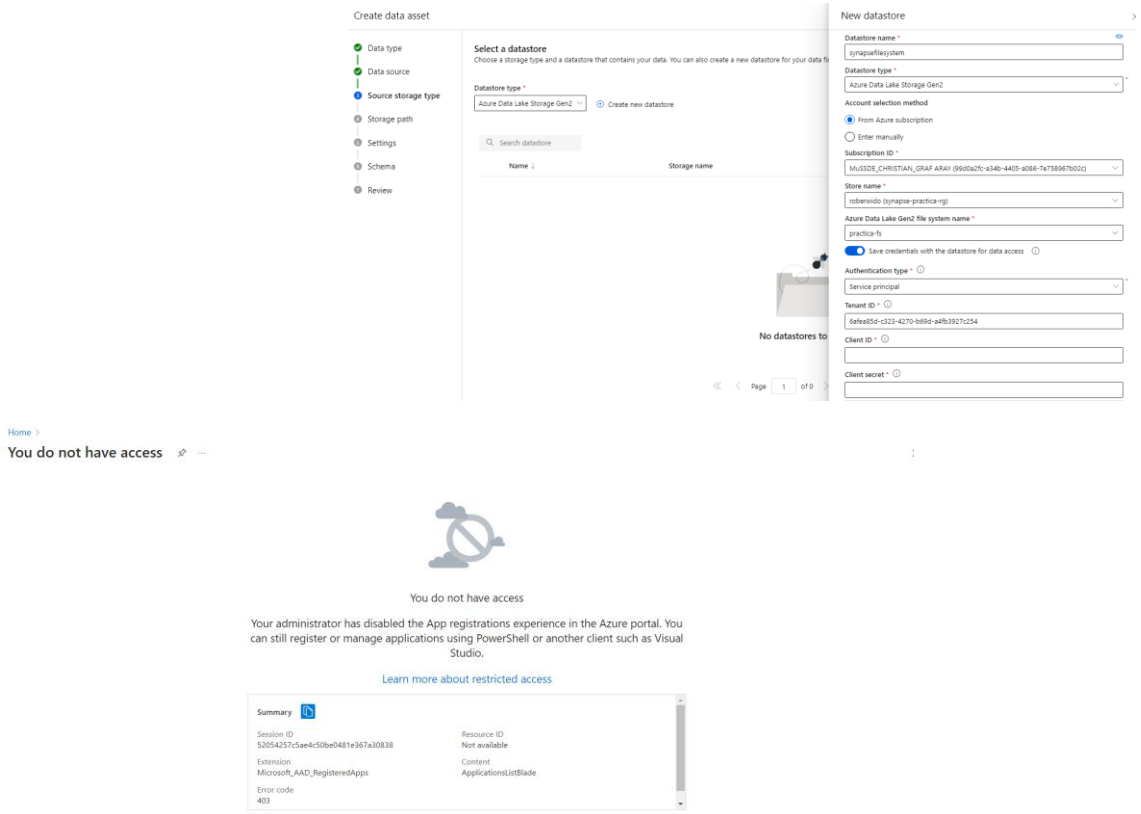
En primer lugar se ha creado un clúster de computación para ejecutar los pasos descritos anteriormente. En la sección de costes se discutirá si el clúster es adecuado para las necesidades, pero se ha decidido elegir una Standard_DS3_v2, configurando el rango de nodos de 0 a 4. En las siguientes capturas se puede ver los pasos de creación.



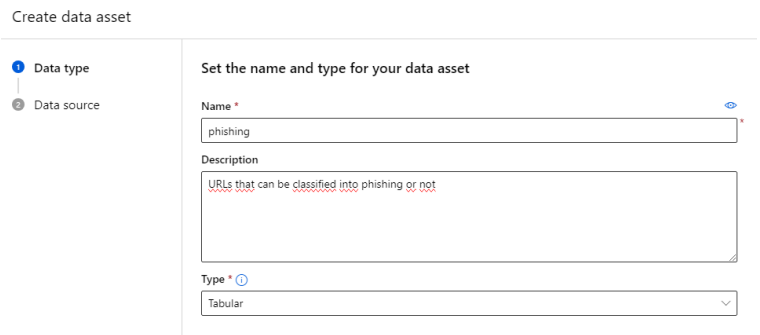


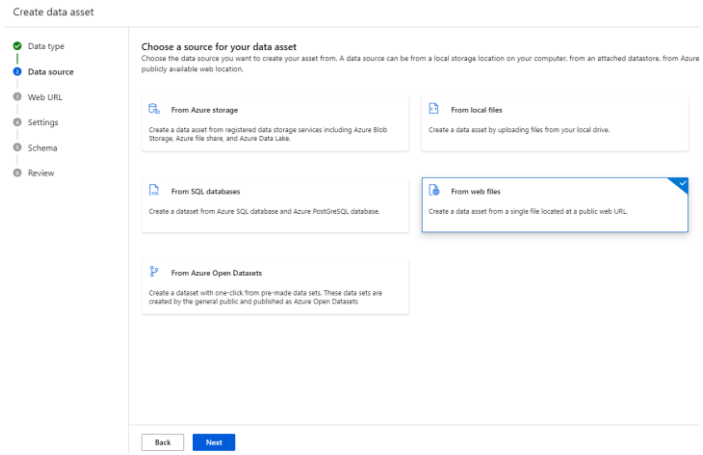
3.2. Importación del dataset

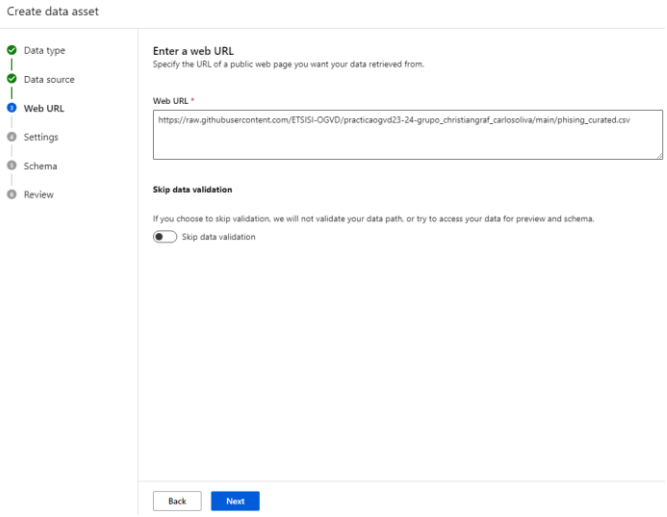
Nuestra primera idea era poder importar directamente desde el filesystem creado para el primer apartado el dataset curado. Para ello, había que crear un datastore de tipo Data Lake Gen 2 y enlazar con el filesystem del usuario utilizado anteriormente. Como se ve en las siguientes dos capturas, nos pedían 2 campos especiales: Client ID y Client secret. Lamentablemente, la única forma de acceder a ellos está restringida por la organización (UPM) a nuestras cuentas de usuarios, y al no poder acceder al servicio "App Registrations" para obtener esos campos, hemos tenido que subir los datos desde GitHub como en el apartado anterior.

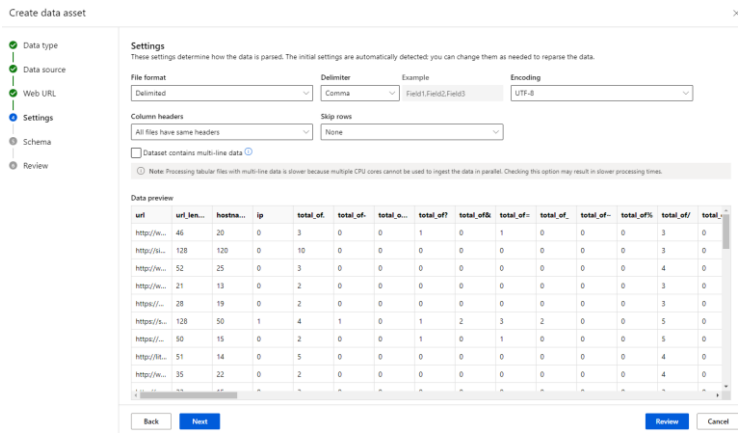


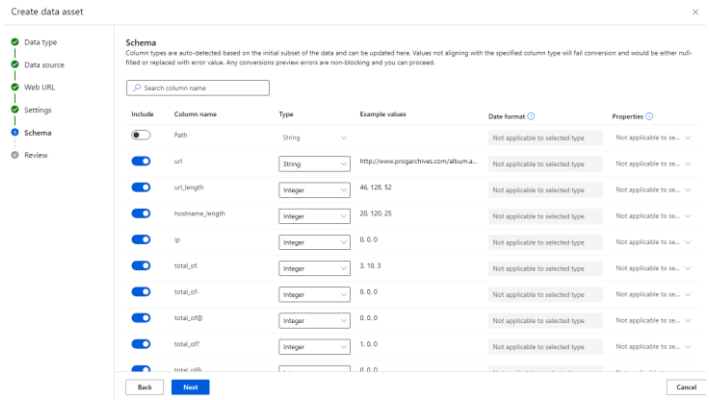
Así que, en las siguientes capturas se puede apreciar cómo se importa el dataset curado desde GitHub. Para que funcione correctamente en Azure, es importante utilizar el enlace que ofrece GitHub de tipo raw. Así, el formato y los tipos de datos se detectan automáticamente.

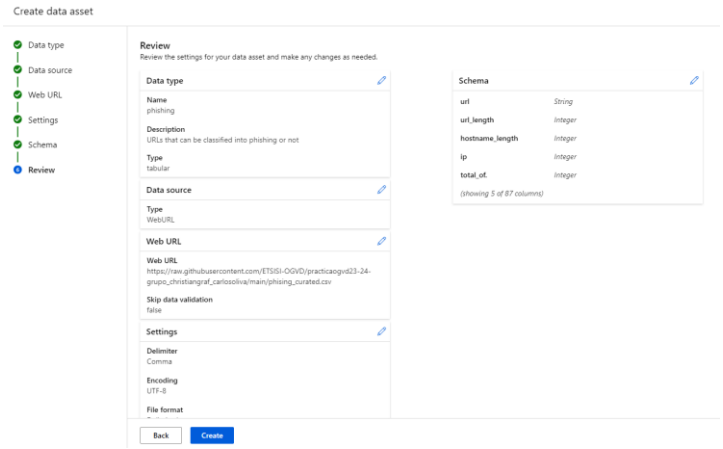



Create data asset
Data type
Data source
Web URL
Settings
Schema
Review
Choose a source for your data asset
Choose the data source you want to create your asset from. A data source can be from a local storage location on your computer, from an attached datastore, from Azure publicly available web location.
From Azure storage
Create a data asset from registered data storage services including Azure Blob Storage, Azure file share, and Azure Data Lake.
From local files
Create a data asset by uploading files from your local drive.
From SQL databases
Create a dataset from Azure SQL database and Azure PostGreSQL database.
From web files
Create a data asset from a single file located at a public web URL.
From Azure Open Datasets
Create a dataset with one-click from pre-made data sets. These data sets are created by the general public and published as Azure Open Datasets
Back
Next


Create data asset
Data type
Data source
Web URL
Settings
Schema
Review
Enter a web URL
Specify the URL of a public web page you want your data retrieved from.
Web URL *
https://raw.githubusercontent.com/ETSISI-OGVD/practicaogvd23-24-grupo_christiangraf_carlosoliva/main/phising_curated.csv
Skip data validation
If you choose to skip validation, we will not validate your data path, or try to access your data for preview and schema.
Skip data validation
Back
Next

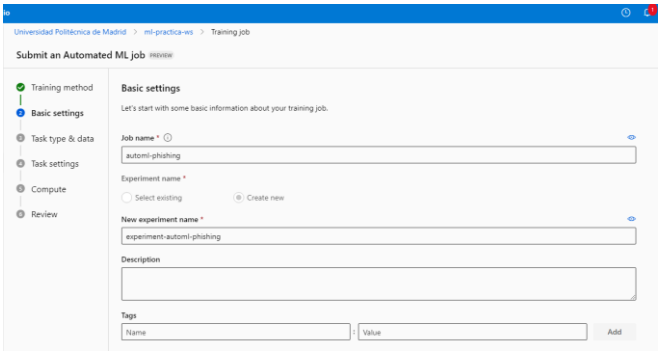

Create data asset
Data type
Data source
Web URL
Settings
Schema
Review
Settings
These settings determine how the data is parsed. The initial settings are automatically detected, you can change them as needed to reparse the data.
File format
Delimited
Delimiter
Comma
Example
Field1,Field2,Field3
Encoding
UTF-8
Column headers
All files have same headers
Skip rows
None
Dataset contains multi-line data
Note: Processing tabular files with multi-line data is slower because multiple CPU cores cannot be used to ingest the data in parallel. Checking this option may result in slower processing times.
Data preview
Back
Next
Review
Cancel

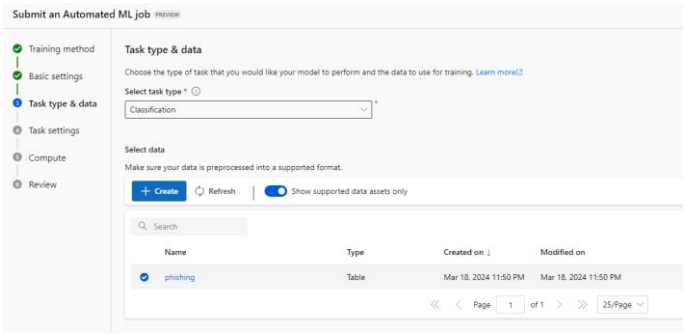


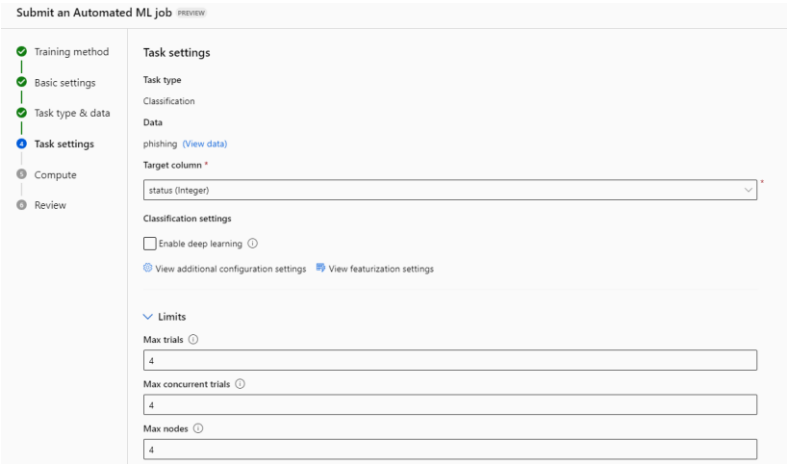


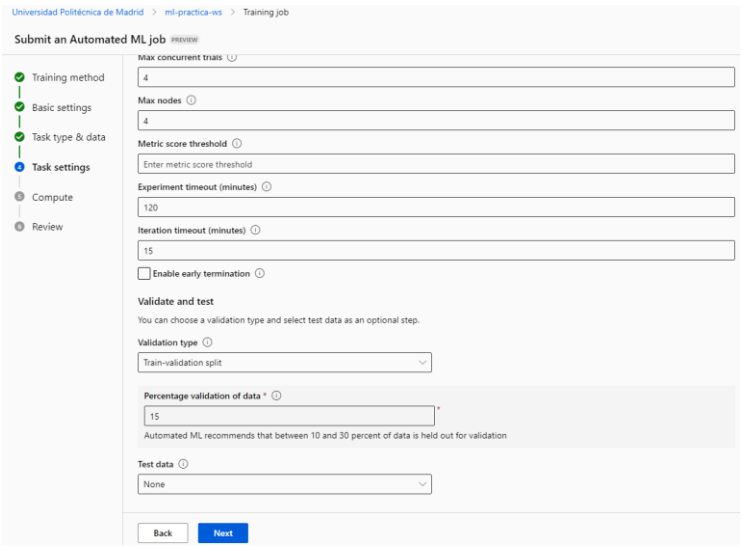
3.3. AutoML: encontrar el mejor modelo de clasificación

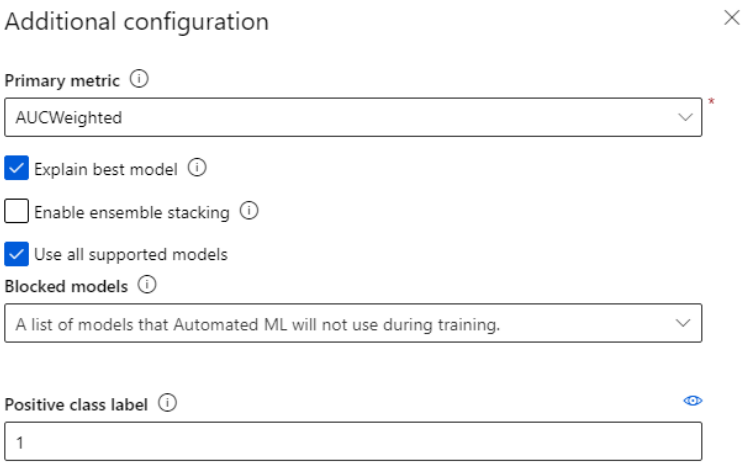
Una vez importados los datos, lanzaremos un proceso de AutoML que puede consultarse aquí. El proceso seguido se puede ver en las siguientes capturas, se ha permitido que se prueben todos los modelos de machine learning disponibles, con límites de tiempo de 15 minutos cada uno y 120 en total. Además, se ha permitido que se utilicen todos los nodos del clúster.

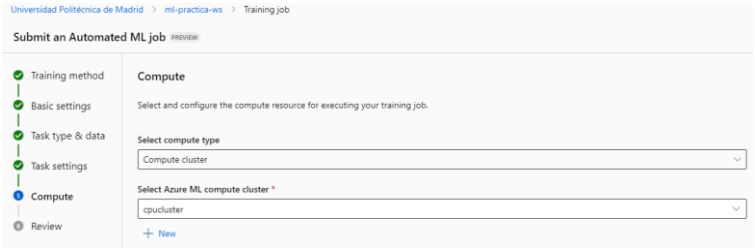


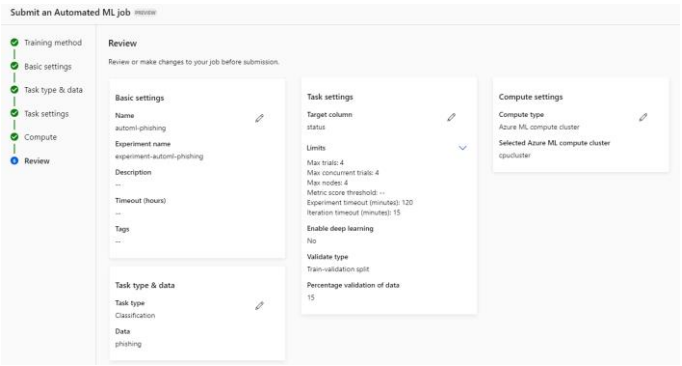

Submit an Automated ML job
Training method
Basic settings
Task type & data
Task settings
Compute
Review
Task type & data
Choose the type of task that you would like your model to perform and the data to use for training. Learn more
Select task type *
Classification
Select data
Make sure your data is preprocessed into a supported format.
Create
Refresh
Show supported data assets only
Search
Name
Type
Created on
Modified on
phishing
Table
Mar 18, 2024 11:50 PM
Mar 18, 2024 11:50 PM
Page 1 of 1
25/Page


Submit an Automated ML job
Training method
Basic settings
Task type & data
Task settings
Compute
Review
Task settings
Task type
Classification
Data
phishing (View data)
Target column *
status (Integer)
Classification settings
Enable deep learning
View additional configuration settings
View featurization settings
Limits
Max trials
4
Max concurrent trials
4
Max nodes
4

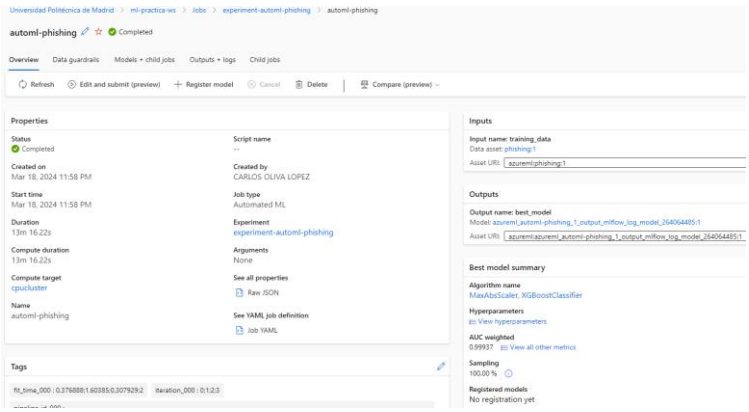

Universidad Politécnica de Madrid
ml-practica-ws
Training job
Submit an Automated ML job
Training method
Basic settings
Task type & data
Task settings
Compute
Review
Max concurrent trials
4
Max nodes
4
Metric score threshold
Enter metric score threshold
Experiment timeout (minutes)
120
Iteration timeout (minutes)
15
Enable early termination
Validate and test
You can choose a validation type and select test data as an optional step.
Validation type
Train-validation split
Percentage validation of data *
15
Automated ML recommends that between 10 and 30 percent of data is held out for validation
Test data
None
Back
Next

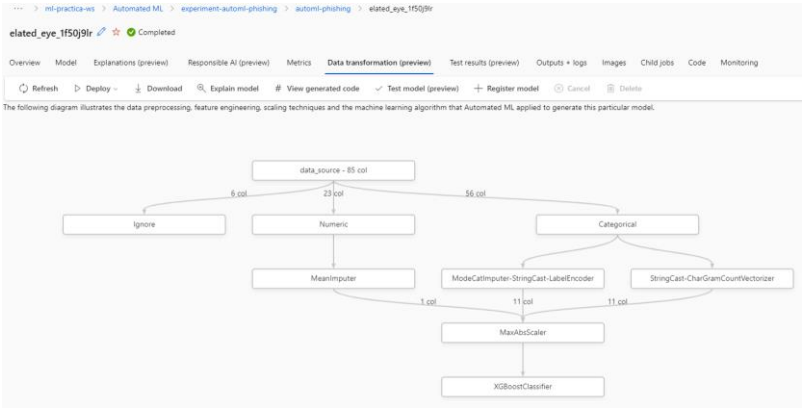


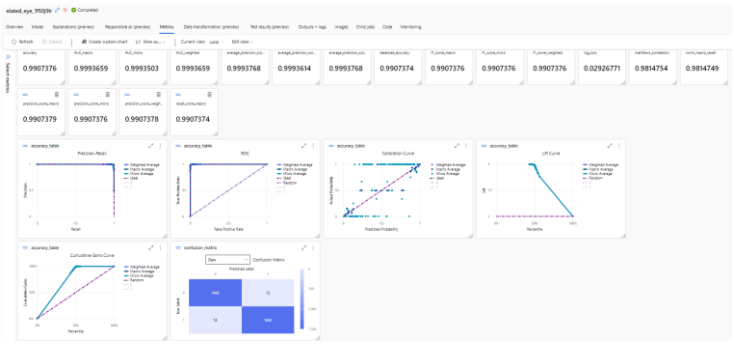


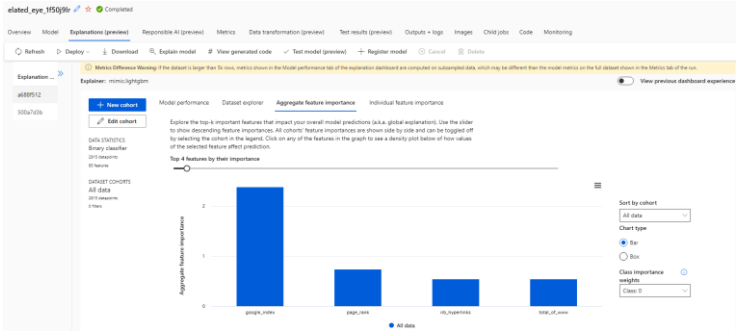


Una vez finalizado, observamos que el mejor modelo ha sido un XGBoostClassifier. La ventaja de Azure es que nos permite activar la explicabilidad, que en la última captura nos devuelve las 4 features más importantes. Además, también explica que tratamiento previo de los datos ha hecho (incluso descartando features).



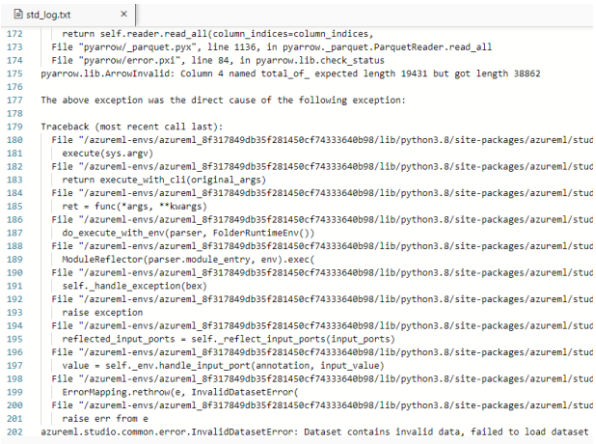

elated_eye_1f50j9lr
Completed
Overview
Model
Explanations (preview)
Responsible AI (preview)
Metrics
Data transformation (preview)
Test results (preview)
Outputs + logs
Images
Child jobs
Code
Monitoring
Refresh
Deploy
Download
Explain model
View generated code
Test model (preview)
Register model
Cancel
Delete
The following diagram illustrates the data preprocessing, feature engineering, scaling techniques and the machine learning algorithm that Automated ML applied to generate this particular model.
data_source - 85 col
6 col
23 col
56 col
Ignore
Numeric
Categorical
MeanImputer
ModeCatImputer-StringCast-LabelEncoder
StringCast-CharGramCountVectorizer
1 col
11 col
11 col
MaxAbsScaler
XGBoostClassifier


0.9907376
0.9993659
0.9993503
0.9993659
0.9993768
0.9993614
0.9993768
0.9907374
0.9907376
0.9907376
0.9907376
0.02926771
0.9814754
0.9814749
0.9907379
0.9907376
0.9907378
0.9907374


elated_eye_1f50j9lr
Completed
Explanations (preview)
Refresh
Deploy
Download
Explain model
View generated code
Test model (preview)
Register model
New cohort
Edit cohort
Model performance
Dataset explorer
Aggregate feature importance
Individual feature importance
Top 4 features by their importance
Sort by cohort
All data
Chart type
Bar
Box
Class importance weights
Class: 0
All data

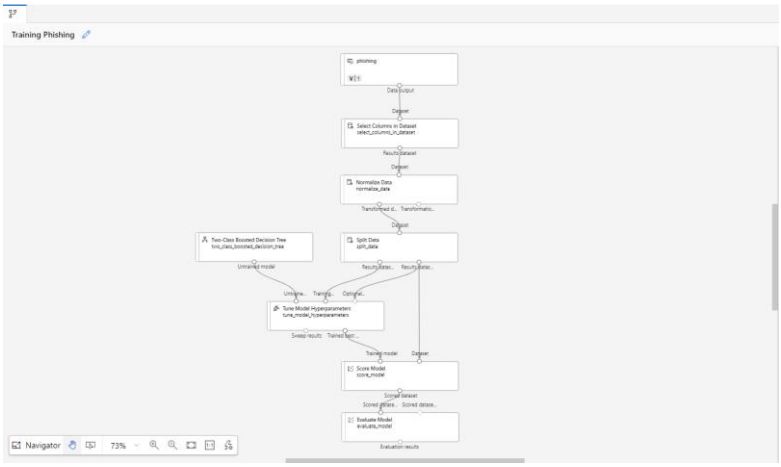
3.4. Pipelines de Designer: un entrenamiento más personalizado

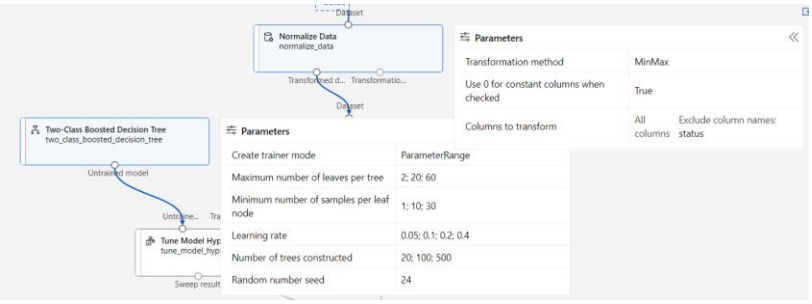
Como aclaración antes de empezar este punto, hemos encontrado que la forma de procesar un dataset de entrada no es igual cuando se utiliza AutomatedML que Designer. En AutoML el dataset curado funciona perfecto, pero para Designer hemos tenido que realizar una segunda curación (se encuentra en el notebook del EDA). Lo que se ha hecho es sustituir los nombres de columnas que tenían caracteres especiales como "total_of?" o "total_of$" por, respectivamente, "total_of_questionmarks" y "total_of_dollars" (y así con todas las columnas que cumplieran esta condición). Por lo tanto, el nuevo dataset curado para pipelines se puede encontrar aquí. Adjuntamos una captura del error que ocurría al utilizar el anterior dataset, que claramente no era soportado por esos caracteres.

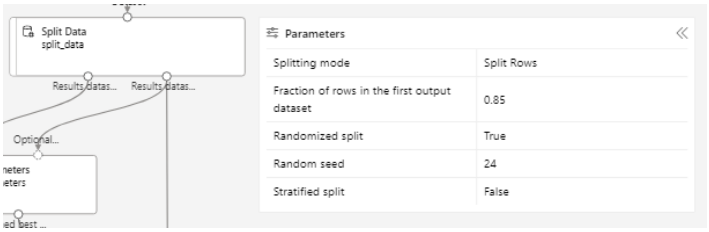


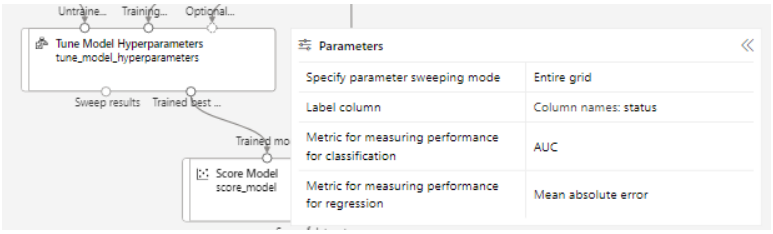
3.4.1. Pipeline de entrenamiento

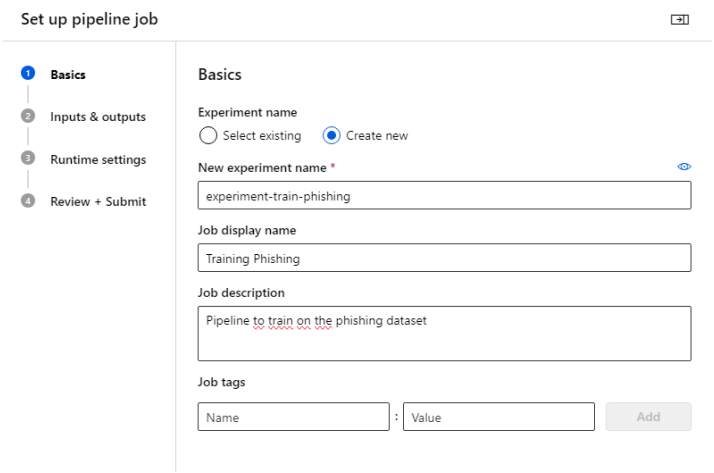
A continuación, realizaremos la última prueba con Azure ML Studio. Diseñaremos una Pipeline de entrenamiento con la herramienta gráfica Designer. En este caso, hemos aprovechado para definir cómo queremos que sea el entrenamiento. En las siguientes capturas se aprecia la configuración de cada uno de los bloques del pipeline. A destacar, la normalización aplicada y el uso de un ParameterRange a la hora de entrenar, lo que permite hacer una búsqueda de hiperparámetros en forma de grid.

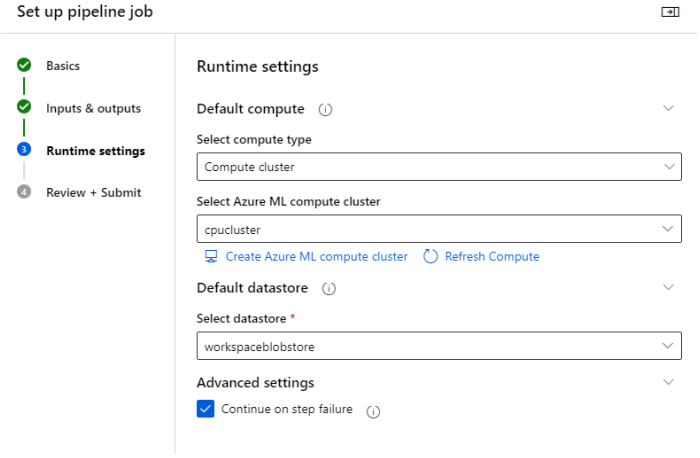



Normalize Data
normalize_data
Parameters
Transformation method
MinMax
Use 0 for constant columns when checked
True
Columns to transform
All columns
Exclude column names: status
Two-Class Boosted Decision Tree
two_class_boosted_decision_tree
Parameters
Create trainer mode
ParameterRange
Maximum number of leaves per tree
2; 20; 60
Minimum number of samples per leaf node
1; 10; 30
Learning rate
0.05; 0.1; 0.2; 0.4
Number of trees constructed
20; 100; 500
Random number seed
24

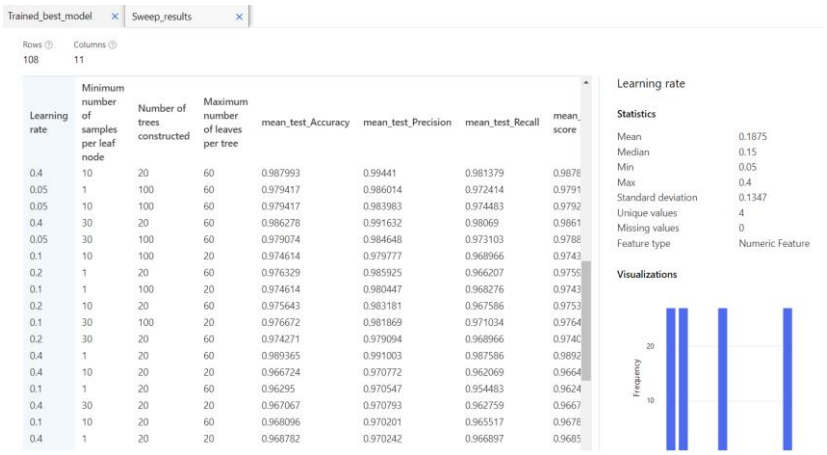

Split Data
split_data
Parameters
Splitting mode
Split Rows
Fraction of rows in the first output dataset
0.85
Randomized split
True
Random seed
24
Stratified split
False


Tune Model Hyperparameters
tune_model_hyperparameters
Parameters
Specify parameter sweeping mode
Entire grid
Label column
Column names: status
Metric for measuring performance for classification
AUC
Metric for measuring performance for regression
Mean absolute error
Score Model
score_model

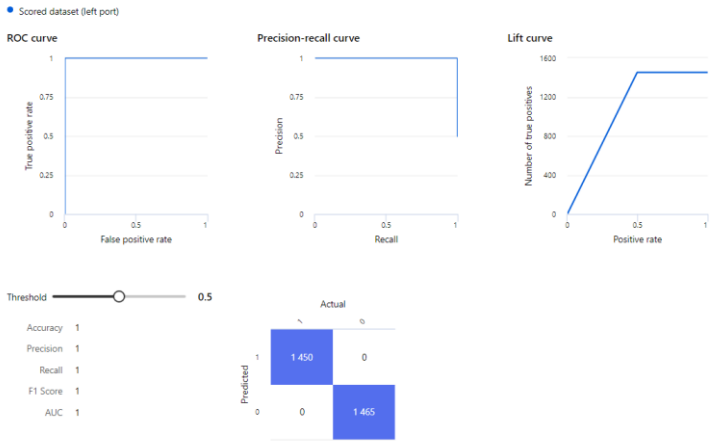

Set up pipeline job
Basics
Inputs & outputs
Runtime settings
Review + Submit
Basics
Experiment name
Select existing
Create new
New experiment name *
experiment-train-phishing
Job display name
Training Phishing
Job description
Pipeline to train on the phishing dataset
Job tags
Name
Value
Add


Set up pipeline job
Basics
Inputs & outputs
Runtime settings
Review + Submit
Runtime settings
Default compute
Select compute type
Compute cluster
Select Azure ML compute cluster
cpucluster
Create Azure ML compute cluster
Refresh Compute
Default datastore
Select datastore *
workspaceblobstore
Advanced settings
Continue on step failure

Una vez finalizado el experimento, podemos observar el valor de las distintas métricas para cada conjunto de hiperparámetros.

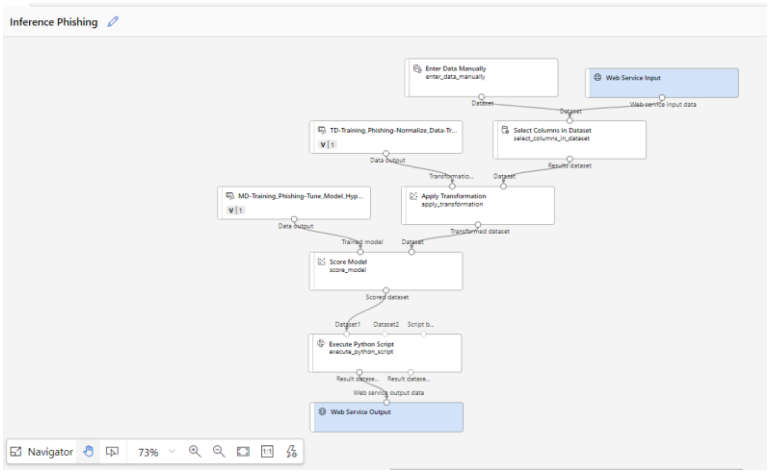


Para el mejor modelo, a pesar de haber utilizado un split del conjunto total como validación durante el entrenamiento, se ha producido overfitting ya que como vemos, los resultados son del 100%.



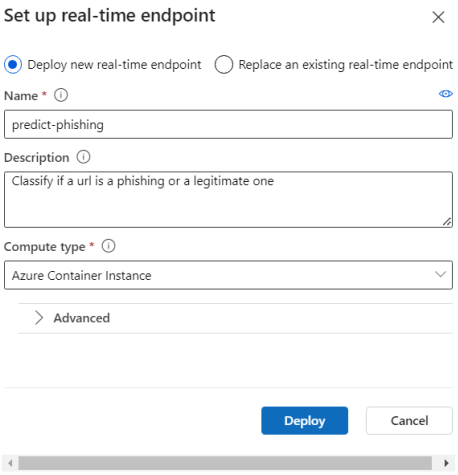
3.4.2. Pipeline de inferencia y despliegue

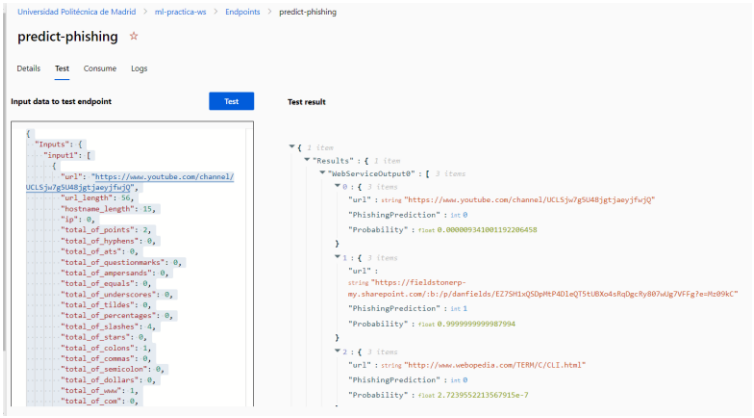
Por último, crearemos una Pipeline de inferencia en la que introduciremos unos datos manualmente y comprobaremos la salida.



A destacar, la configuración del script de Python que se encuentra aquí. Simplemente renombra las columnas de salida. Si comprobamos las salidas, y comparamos con el dataset original, veremos que ha acertado (se ha utilizado desde la línea 16 a la línea 24).

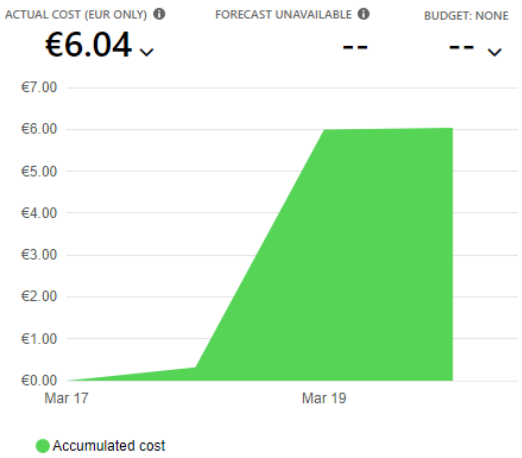
Por último, desplegaremos este modelo en un endpoint para que, hipotéticamente, alguien pudiera utilizarlo como API con sus propios datos. Se puede observar que, con estos datos en formato JSON, las salidas son correctas y el servicio responde.





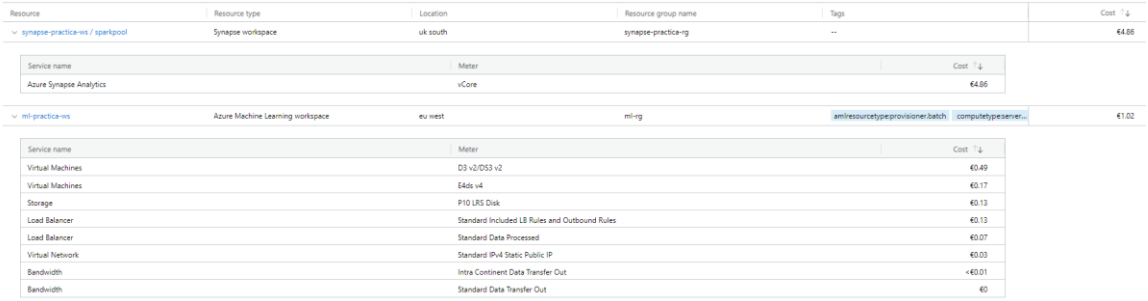
4. Costes

En esta última sección, analizaremos los costes de las operaciones realizadas en los últimos dos apartados. Lo primero es echar un vistazo a los costes acumulados a lo largo de la realización de la práctica. En total, un gasto de algo más de 6€ en 3 días.

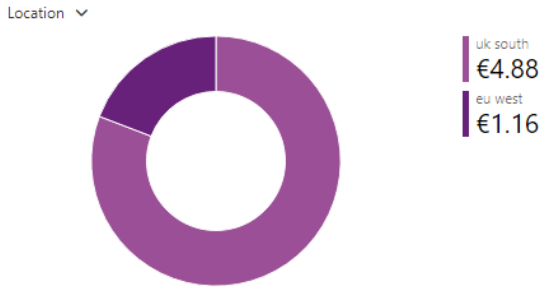


Analizando el coste diario, observamos que el grueso del coste fue el día 19 y causado por Synapse Analytics.

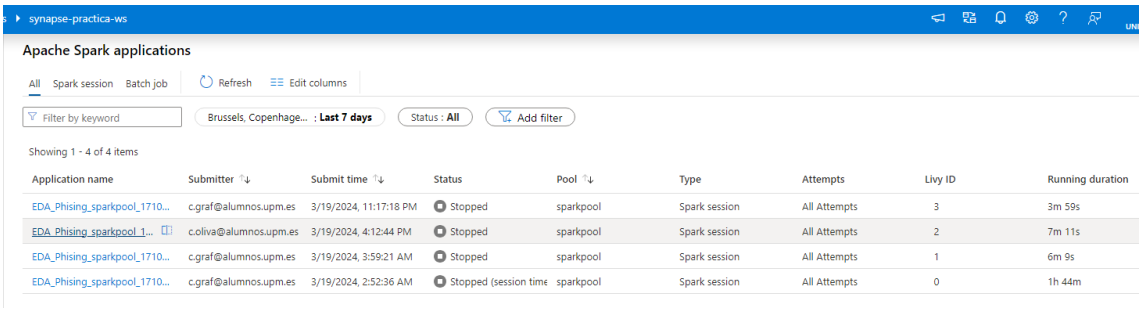
Si analizamos por recurso, el gasto de Synapse Analytics es única y exclusivamente por el uso de los vCores, es decir, por cada vez que se ha arrancado el pool de PySpark. El gasto por el filesystem del usuario creado es ínfimo (<0,01€). En cambio, los gastos de ML Studio están más repartidos entre varios componentes. Los primeros 0,49€ que aparecen son los equivalentes al clúster creado, mientras que los 0,17€ son equivalentes al endpoint del modelo desplegado. El resto de gastos, como por ejemplo el load balancer, se debe a los gastos asociados cada vez que se lanzaba un pipeline y se requería escalar el número de nodos del clúster creado. Por lo tanto, el propio reescalamiento vemos que lleva un coste asociado.



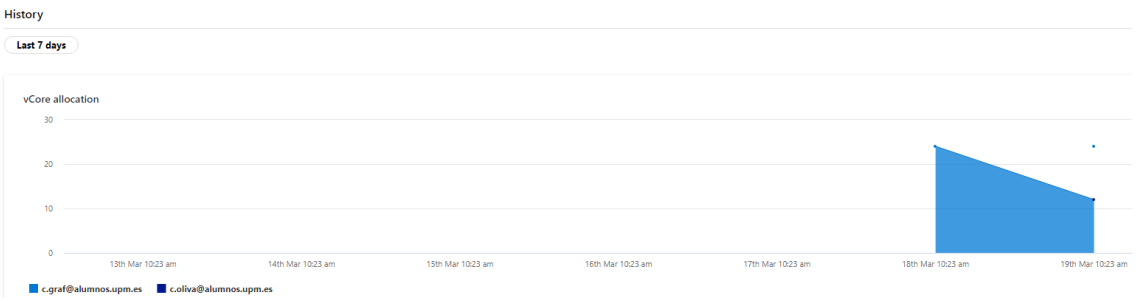
Por ultimo a nivel general, podemos distinguir los gastos de Synapse Analytics y ML Studio porque, debido a limitaciones de Azure en el momento de intentar crear el primero, no nos permitía la región West Europe. Así, dado que la latencia no era un problema, escogimos la región más cercana (South UK) y los gastos han quedado repartidos en dos zonas.



Pasando a analizar más concretamente los gastos de Synapse, podemos apreciar en el propio workspace todas las ejecuciones realizadas y el tiempo que duraron. En este caso, fueron 4.



Y los vCores utilizados por día, que encajan perfectamente con las ejecuciones de arriba.



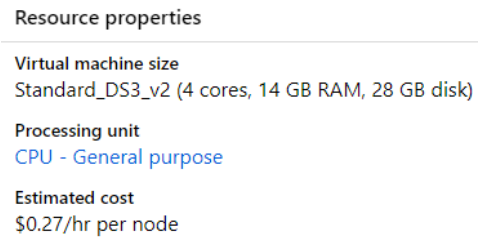
Teniendo en cuenta que el vCore por hora es de 0,56€ en la familia del pool de PySpark seleccionado, podríamos calcular de la primera sesión que se ejecutó, que fue la más larga de todas con una duración de 1 hora y 44 minutos.

$$GastoSesion1\ vCore = \left(1h + \frac{44}{60}h\right) * 0{,}56\frac{€}{h} = 0{,}97€\ vCore$$

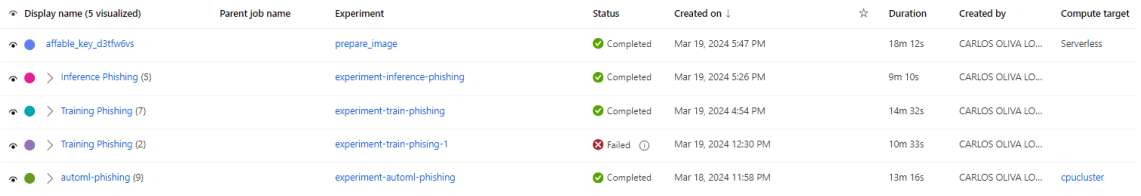
Como el pool puede utilizar entre 3 y 6 vCores, el gasto de esa primera sesión está comprendido entre 2,91€ y 5,82€. Si analizamos la grafica de costes diarios veremos que el coste estuvo cerca de los 5€. Por lo tanto, habría que analizar si es rentable el uso de Synapse Analytics. ¿Vale la pena ese gasto para un EDA de un dataset verdaderamente pequeño? Lo cierto es que esa sesión debería haberse acortado, ya que la mayoría del tiempo se utilizó para programar código y probar en tiempo real, mientras que el paradigma a adoptar en estos casos (si se quiere ahorrar) es programar todo de antemano

y solo ejecutar cuando sea necesario hacer una prueba. Es decir, que el tiempo durante el cual el pool esté activo sea única y exclusivamente utilizado para ejecución. Sería interesante poder comparar con el gasto en luz equivalente a hacer estas ejecuciones en un ordenador local, aunque lo cierto es que no se tendría la misma infraestructura ni disponibilidad de cores. Por lo que hemos visto, el gasto puede dispararse muy rápidamente si se descuidan los tiempos de ejecución, lo que podría suponer gastos innecesarios para una empresa o incluso un problema económico para pymes o pequeñas empresas emergentes.

Por último, aunque los gastos de ML Studio han sido mínimos (1,18€) conviene analizarlos también. El coste aproximado por nodo utilizado es de 0,25€ en el clúster escogido, un DS3 v2. Si recordamos lo dicho anteriormente, el gasto de todas las ejecuciones en este clúster fue de 0,49€, al que habría que sumar los costes de 0,13€ por reescalamiento automático.



Teniendo en cuenta los experimentos lanzados (pueden verse abajo), tanto los intentos acertados como fallidos, es cierto que no parece un gasto muy alto y hasta parece rentable. De hecho, hay que tener en cuenta que con ese gasto se han entrenado varios modelos (aunque nos hayamos quedado con los mejores solo) y se ha realizado una pipeline de inferencia también.



Se podrían haber ahorrado gastos si se hubiera escogido un clúster más humilde, ya que lo cierto es que los entrenamientos realizados eran de machine learning y no de deep learning con redes neuronales. Sería interesante analizar la relación de bajar los recursos de computo para ver si compensa, ya que habría que tener en cuenta que el tiempo de ejecución aumentaría. Por último, lo que sí que parece caro es levantar un endpoint con el modelo para poder utilizarlo como API. Teniendo en cuenta que estuvo levantado tan solo 18 minutos (como se aprecia en el primer job de la captura de arriba), el consumo de 0,17€ nos hace una idea de cómo de rápido subiría el precio, aproximadamente 14€ por día levantado. Al igual que se comentaba antes, también seria interesante analizar qué otros tipos de recursos endpoint ofrece Azure, ya que la inferencia de un modelo como el que hemos entrenado (básicamente un árbol de decisión) requiere muy poco cómputo.

5. Referencias

- Dataset - https://www.kaggle.com/datasets/winson13/dataset-for-link-phishing-detection
- Repositorio de GitHub - https://github.com/ETSISI-OGVD/practicaogvd23-24-grupo_christiangraf_carlosoliva/tree/main
- Suscripción de Azure - https://portal.azure.com/#@upm365.onmicrosoft.com/resource/subscriptions/99d0a2fc-a34b-4405-a086-7e758967b02c/overview
- Workspace de Synapse Analytics - https://web.azuresynapse.net/?workspace=%2fsubscriptions%2f99d0a2fc-a34b-4405-a086-7e758967b02c%2fresourceGroups%2fsynapse-practica-rg%2fproviders%2fMicrosoft.Synapse%2fworkspaces%2fsynapse-practica-ws
- Workspace de ML Studio - https://ml.azure.com/?tid=6afea85d-c323-4270-b69d-a4fb3927c254&wsid=/subscriptions/99d0a2fc-a34b-4405-a086-7e758967b02c/resourceGroups/ml-rg/providers/Microsoft.MachineLearningServices/workspaces/ml-practica-ws
- Get Started with Azure Synapse Analytics - https://learn.microsoft.com/en-us/azure/synapse-analytics/get-started
- What is automated machine learning (AutoML)? - https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml?view=azureml-api-2
- Tutorial: Train a classification model with no-code AutoML in the Azure Machine Learning studio - https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-first-experiment-automated-ml?view=azureml-api-2
- Explore classification with Azure Machine Learning Designer - https://microsoftlearning.github.io/AI-900-AIFundamentals/instructions/02b-create-classification-model.html
- El resto de enlaces que han ido apareciendo a lo largo de esta memoria.