Documentación Prueba Técnica DataKnow

# Sebastian Mariño

# 1. Carga de Información

La realización de este ejercicio fue desarrollada mediante el lenguaje de programación Python. En un primer acercamiento al problema, se utilizó la función open para abrir el archivo OFEI1204.txt ubicado en la carpeta '../Datos3'. El modo 'r' especifica que el archivo se abrirá en modo lectura. El contenido del archivo se leyó y almacenó en la variable txt\_data. Posteriormente, se imprimió el contenido del archivo para identificar las particularidades de la Data, en donde se notó que cada tabla de información venía delimitado por el titulo agente y líneas vacías.

Luego, se procedió a la extracción de información relevante del texto. Se dividió el contenido del archivo en secciones utilizando el método split con el delimitador "AGENTE:". Cada sección resultante contenía información sobre un agente específico. Se recorrieron estas secciones, extrayendo el nombre del agente y la tabla de datos asociada. La información de cada agente se agregó a la lista agentes, y las tablas de datos se procesaron y almacenaron en la lista dfs.

Después de completar la extracción de información, se utilizó la biblioteca pandas para manipular y analizar los datos de manera eficiente. Se concatenaron todas las tablas de datos en un único DataFrame llamado df\_final. Se filtraron las filas del DataFrame donde la columna "AGENTE" era igual a "EEPPM" para verificar la correcta lectura del archivo ya que se comparó esta salida con el archivo .txt original. Luego, se filtraron las filas donde la columna con índice 1 era igual a "D" y se reorganizaron las columnas. Se eliminó la columna 'TIPO' y se renombraron las columnas de acuerdo con los requisitos del problema.

Finalmente, se realizaron ajustes en el formato de algunas columnas, convirtiendo las columnas que representan horas a tipo de dato float. El DataFrame resultante se guardó en un archivo CSV llamado 'tabla\_punto\_1.csv' en la carpeta '../resultados'.

En resumen, el código realiza la extracción, procesamiento y manipulación de datos de un archivo de texto, utilizando Python y pandas, para obtener un DataFrame final que satisface los requisitos específicos del problema planteado.

# 2. Manipulación de datos

La implementación de este código en Python se realizó con el objetivo de manipular y analizar datos contenidos en un archivo Excel ('Datos Maestros VF.xlsx') y un archivo de texto ('dDEC1204.TXT'). Aquí se detalla paso a paso el procedimiento:

En primer lugar, se seleccionaron las columnas relevantes del archivo Excel utilizando la biblioteca pandas. Se leyó el archivo 'Datos Maestros VF.xlsx' y se extrajeron las columnas especificadas en la lista columnas\_seleccionadas. El DataFrame resultante se llamó df\_master\_data.

Luego, se filtraron las filas de df\_master\_data para obtener información específica sobre la empresa "EMGESA" y sobre centrales de tipo hidroeléctrico ("H") o térmico ("T"). El DataFrame resultante se denominó df\_master\_data\_emgesa.

A continuación, se leyó el archivo de texto 'dDEC1204.TXT' con la función pd.read\_csv, especificando el delimitador y la codificación adecuados. Se realizaron ajustes en el DataFrame resultante (df\_dec\_1204), eliminando la última fila y renombrando las columnas con nombres descriptivos.

Se llevó a cabo una fusión (pd.merge) entre los DataFrames df\_master\_data\_emgesa y df\_dec\_1204 utilizando la columna 'CENTRAL (dDEC, dSEGDES, dPRU…)' como clave de unión, y se especificó un tipo de fusión de tipo 'left'. El DataFrame resultante se llamó df\_merged\_emgesa.

A continuación, se seleccionaron las columnas que representan las horas ('Hora\_1', 'Hora\_2', ..., 'Hora\_24') y se calculó la suma de estas columnas para obtener la columna 'Total\_Horas'. Se filtraron las filas donde 'Total\_Horas' era mayor que 0, y el DataFrame resultante se llamó df\_tabla\_punto\_2.

Finalmente, se guardó el DataFrame df\_tabla\_punto\_2 en un archivo CSV llamado 'tabla\_punto\_2.csv' en la carpeta '../resultados'.

En resumen, este código realiza una serie de manipulaciones de datos utilizando pandas, fusionando información de un archivo Excel y un archivo de texto, y generando un DataFrame final que satisface los requisitos establecidos en el problema.

# 3. Prueba de SQL

Como primer paso se corrió la Query otorgada, la cual crea las tablas EMPLEADO,VACACIONES y se lleno con la información especificada.

Texto

Descripción generada automáticamente

Figura 1. Creación Tablas

A continuación, se muestran todas las Querys realizadas para obtener los datos solicitados y los respectivos resultados.

1. **Seleccionar nombre, apellido y salario de todos los empleados:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Figura 2. Nombre apellido y salario

1. **Seleccionar nombre, apellido y salario de todos los empleados que ganen más de 4 millones:**

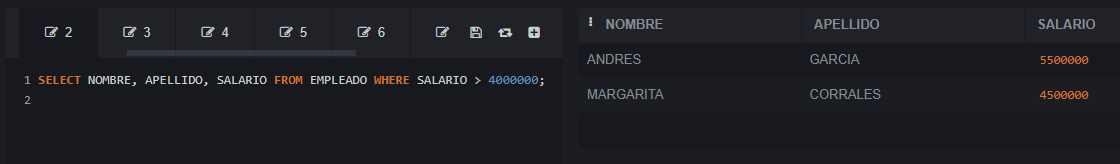
****

Figura 3. Empleados que ganas más de 4M

1. **Contar los empleados por sexo:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 4. Conteo por sexo

1. **Seleccionar los empleados que no han hecho solicitud de vacaciones:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 5. Empleados que no han solicitado vacaciones

1. **Seleccionar los empleados que tengan más de una solicitud de vacaciones y mostrar cuántas solicitudes tienen:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Figura 6. Empleados que han solicitado vacaciones más de una vez

1. **Determinar el salario promedio de los empleados:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 7. Salario promedio

1. **Determinar la cantidad de días promedio solicitados de vacaciones por cada empleado:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Figura 8. Dias poromedio solicitados por empleado

1. **Seleccionar el empleado que ha solicitado la mayor cantidad de días de vacaciones:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 9. Empleado que más ha solicitado vacaciones en total

1. **Consultar la cantidad de días aprobados y rechazados por cada empleado, mostrar 0 en caso de no tener solicitudes:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 10. Tabla de días aprobados y rechazados de vacaciones

# 4. Prueba de AWS

***1. Configuración roles en AWS***

En la Figura 11 se observan los roles creados para las dos instancias requeridas para el desarrollo de este ejercicio, estas son s3 y redshift

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 11. Roles

***2. Configure un Clúster de pruebas de Redshift con los requerimientos mínimos necesarios***

Se crea el Clúster de prueba para ingresar los datos necesarios.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 12.Clúster

***3. Copie la información a Redshift usando algún editor de queries como SQL Workbench/J - Home (sql-workbench.eu)***

Mediante SQL workbench se conecta al clúster creado como se muestra en la siguiente Figura.

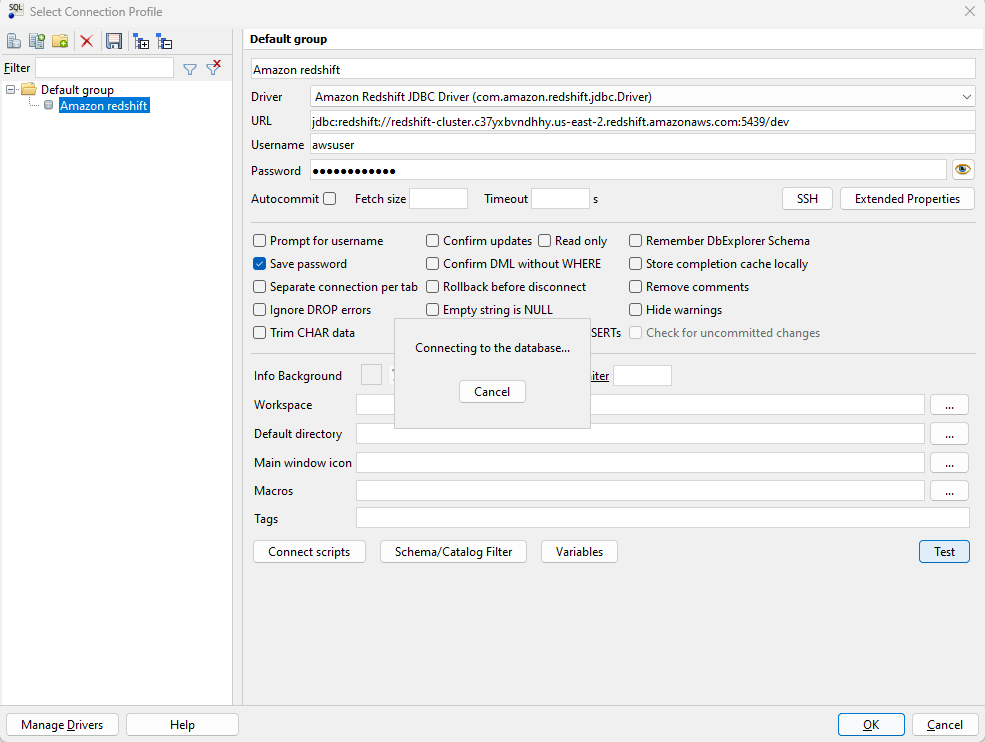
******

Figura 13. Workbench

Posteriormente se cargaron las tablas, a continuación, se muestra un ejemplo de Query SQl para cargar una tala, esto se repitió para los 7 archivos de la base de datos.

COPY events FROM 'C:\Users\{tu\_ruta}\tickitdb \allevents\_pipe.txt'

DELIMITER ',' -- Cambia a tu delimitador

IGNOREHEADER 1 -- Si hay un encabezado en el archivo

CSV; -- O el formato que estés utilizando

Ya que tickitdb es una base de dase de datos de prueba para redshift, otra forma de cargarla es cargando directamente desde la creación del Clúster del siguiente modo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14. Carga de Tickit directamente desde la creación del Clúster

A continuación, se muestra la base de datos relacional tickit.

Diagrama

Descripción generada automáticamente

Figura 15. TickitDb

***4. Responda las siguientes preguntas usando comandos de consultas SQL:***

Median el redshift query editor se realizarán las consultas, a continuación, se muestran las tablas cargadas:

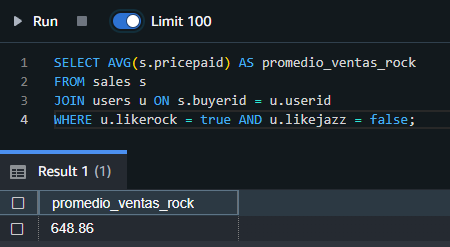
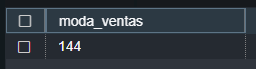
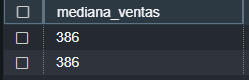
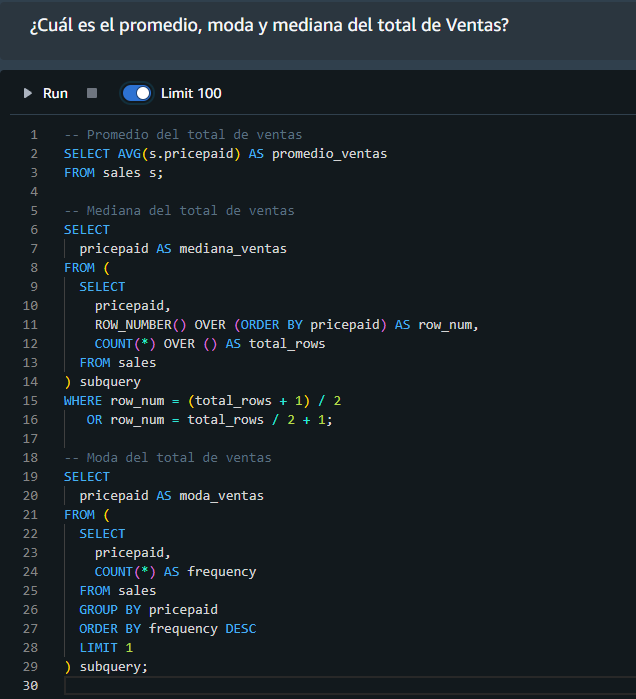
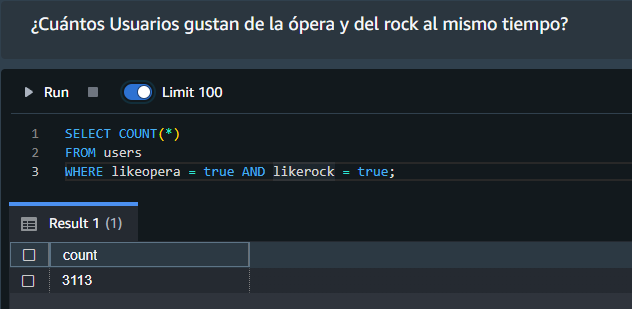
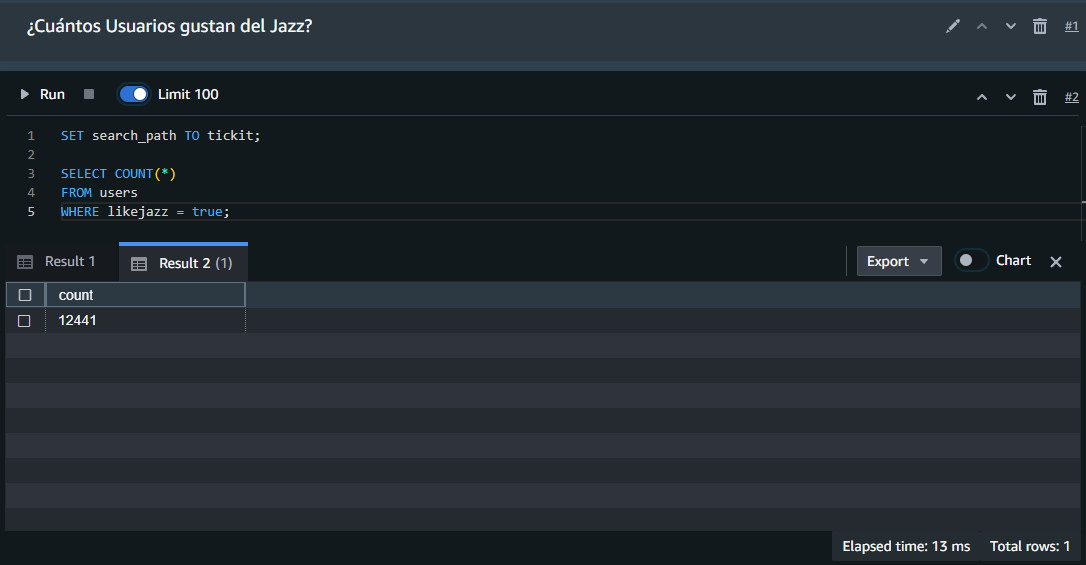
Captura de pantalla de un teléfono celular

Descripción generada automáticamente

Figura 16. Redshift query editor v2

A continuación, se muestran las Querys para resolver las preguntas planteadas:

Figura 17. Querys para resolver las preguntas



***4. En una nueva tabla junte la información (Nombre de usuario, Apellido de usuario, Correo de usuario, Nombre del evento, lugar del evento, Fecha del evento, Cantidad y Total vendidos) y expórtela usando Redshift a un bucket predefinido de S3.:***

Se crua una instancia de S3 y se crea el bucket para almacenar una nueva tabla.

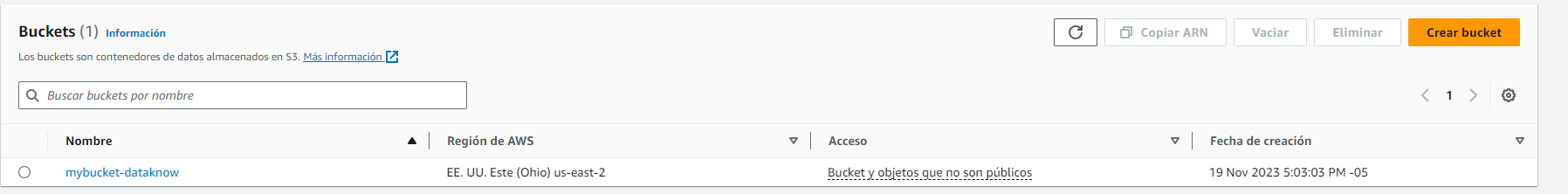


Figura 18. creación bucket

Posteriormente con la siguiente Query se crea la “nueva\_tabla”

Texto

Descripción generada automáticamente

Figura 19. Crear nueva tabla

Mediante la siguiente query se carga la nueva en la ubicación deseada dentro del Bucket

Texto

Descripción generada automáticamente con confianza media

Figura 20. Envío de tabla a S3

Se revisa dentro del bucket el archivo cargado de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 21. Carga del archivo csv

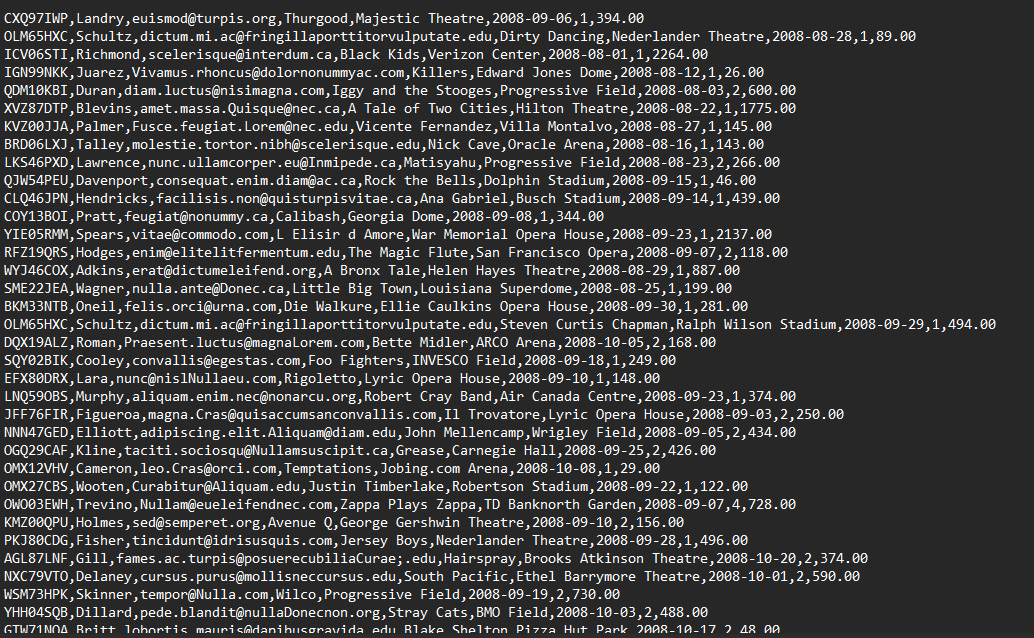


Figura 22. CSV generado

# 5. Prueba de Azure

1.Desplegar base de datos en SQL, con la base de pruebas Adventure Works

Se crea la base de datos SQL en azure como se muestra en la siguiente figura, posteriormente ya se tiene una base de datos con la dataset de pruebas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 23. Creación base de datos

2. Realizar un pipeline con Azure Datafactory, utilizando data flow, para realizar la carga de una base de datos, crear 5 indicadores.

En la siguiente figura se muestra la creación del Dataflow desde Azure:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 24. Dataflow

# 6. Prueba de modelación analítica

En el transcurso del análisis y modelado de datos, se procedió a cargar el conjunto de datos de entrenamiento desde el archivo 'train.csv'. Se identificaron varias columnas que no solo eran consideradas irrelevantes para la predicción de la variable objetivo FRAUDE, sino que también presentaban un alto número de valores nulos. La decisión de eliminar estas columnas se basó en la premisa de que su inclusión no aportaría significativamente al modelo y podría introducir ruido innecesario en el análisis. Después de remover filas con valores nulos y eliminar estas columnas, el conjunto de datos de entrenamiento no experimentó una reducción significativa en su tamaño, si no se hubiesen eliminado las columnas irrelevantes el conjunto de datos sufriría una drástica disminución de casi la mitad de los datos. Este proceso de limpieza y simplificación de características no solo contribuyó a mejorar la eficiencia computacional, sino que también permitió centrarse en las variables más relevantes para el objetivo de predecir transacciones fraudulentas.

Posteriormente, se dividió el conjunto de datos en conjuntos de entrenamiento y prueba, y se ajustaron modelos de Random Forest y SVM. La optimización de hiperparámetros se realizó mediante las técnicas de búsqueda GridSearchCV y RandomizedSearchCV. Se evaluó el rendimiento de los modelos, destacándose el Random Forest por su alta precisión, recall y f1-score. En el siguiente paso, se aplicó el modelo seleccionado a un conjunto de datos de prueba previamente cargado. Se garantizó la consistencia en el orden de las columnas y se aplicó la codificación one-hot, ajustando la columna 'COD\_PAIS' según el mapeo ISO previamente creado. Se generaron probabilidades en lugar de predicciones binarias para ofrecer flexibilidad en la interpretación de los resultados. Los resultados se guardaron en un archivo CSV, incluyendo las probabilidades de fraude. Se optó por el modelo de Random Forest con hiperparámetros optimizados mediante GridSearchCV debido a su robusto rendimiento, especialmente en la minimización de falsos positivos. Este enfoque se seleccionó con la consideración de la crítica asociada a acusar falsamente de fraude, priorizando así la precisión en la detección de transacciones legítimas. Los resultados detallados del rendimiento de los modelos se presentan en el informe, y el archivo CSV resultante se generó para su análisis y revisión adicionales.

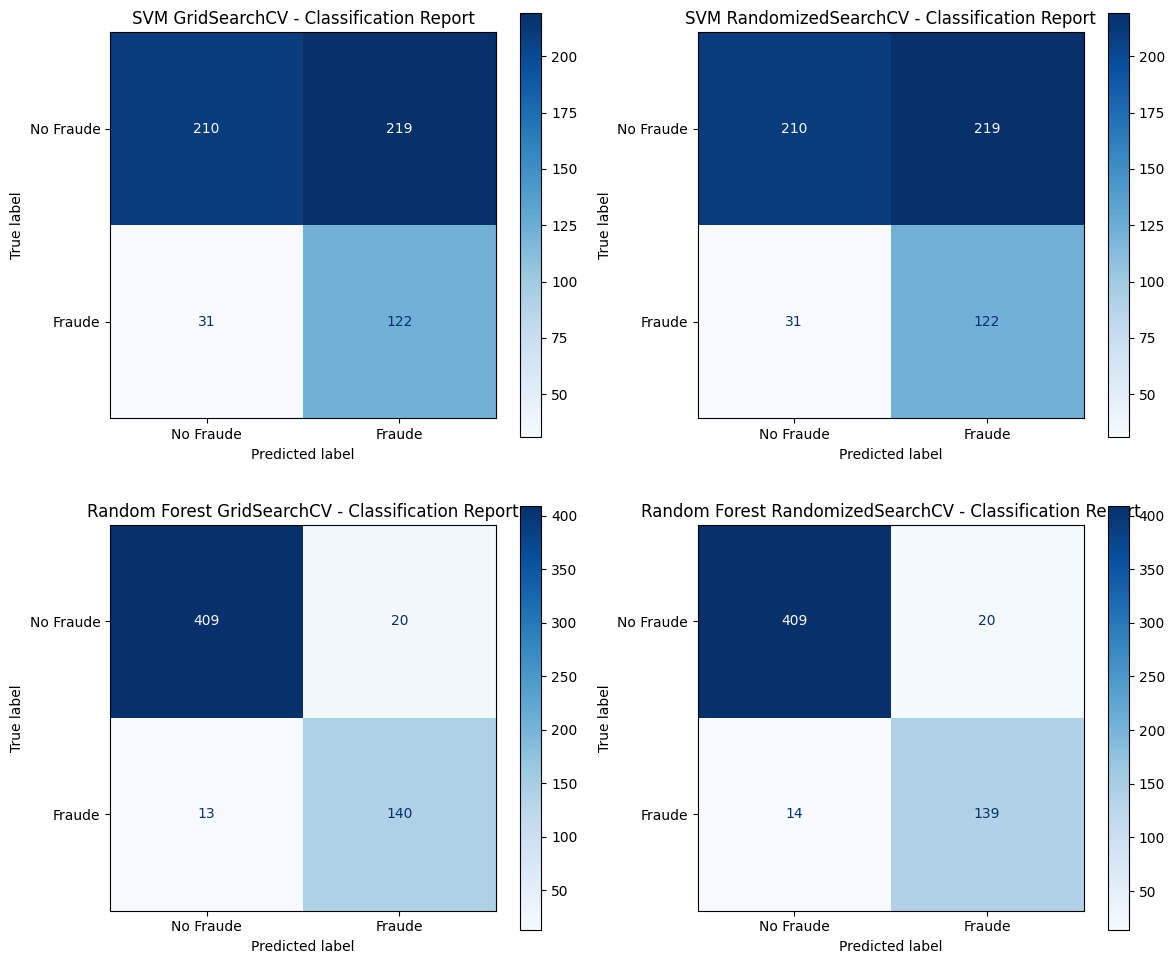


Figura 25. Matriz de confusión con modelos entrenados