Vinkos: reporte de pruebas ciencia de datos & ETL

Sergio Miguel Fernández Martínez

October 2025

Introducción

El prente reporte consta de 7 secciónes de las cuales 1-6 corresponden a la prueba de ciencia de datos, mientras que la sección 7 corresponde a la prueba ETL. Detalles sobre la implementación, código y pseudocódigo pueden hallarse en el repositorio: `https://github.com/sergfer26/ETL-Data-Science-Test`.

1. Procesamiento de bases de datos

Fueron proporcionados dos archivos info_01.csv y info_02.csv. Al hacer una carga de ambos archivos se observa que ambos cuentan 17286 registros. La tabla correspondiente a info_01.csv cuenta con las columnas id, id2, v4, v5 y v6. Mientras que la de info_02.csv cuenta con id, id2, v1, v2 v3 y c1. Se observa que hay una correspondencia entre los renglones de cada tabla por medio de la tupla de indices (id, id2). Sin embargo, en ambos casos hay registros repetidos que deben ser removidos antes realizar un *inner join*. Al tomar los registros únicos de cada tabla y realizar un *inner join* como muestra el bloque de código 1 se obtiene la tabla **info**, que cuenta con 9143 registros. Una vista previa de la tabla se encuentra en el cuadro 1.

```python
import pandas as pd

info_01 = pd.read_csv("data/info_01.csv")
info_02 = pd.read_csv("data/info_02.csv")

info_01_filtered = info_01[~info_01.duplicated(['id', 'id2', 'v4', 'v5', 'v6'], keep='first')]
info_02_filtered = info_02[~info_02.duplicated(['id', 'id2', 'v1', 'v2', 'v3', 'c1'], keep='first')]
info = info_02_filtered.set_index(['id', 'id2']).join(info_01_filtered.set_index(['id', 'id2']), how="inner")

print(info)
```

Listing 1: Filtrado de primer registro e inner join

id	id2	v1	c1	v2	v3	v4	v5	v6
1	1	426.0	1	23.2	2015-02-04 17:51:00	721.2	27.3	0.004793
2	2	429.5	1	23.1	2015-02-04 17:51:59	714.0	27.3	0.004783
3	3	426.0	1	23.1	2015-02-04 17:53:00	713.5	27.2	0.004779
4	4	426.0	1	23.1	2015-02-04 17:54:00	708.2	27.2	0.004772
5	5	426.0	1	23.1	2015-02-04 17:55:00	704.5	27.2	0.004757
					...			
5411	5411	65.7	0	19.2	NaN	423.3	30.8	0.004235
6547	6547	0.0	0	19.3	NaN	463.0	26.1	0.003606
1806	1806	0.0	0	20.2	NaN	445.0	21.2	0.003104
5606	5606	58.8	0	20.1	NaN	420.2	26.3	0.003830
3689	3689	0.0	0	19.6	NaN	449.0	19.3	0.002729

Cuadro 1: Mustras de variables v1-v6 y c1

2. Análisis exploratorio

Haciendo un análisis preliminar sobre las columnas independientes se encuentró que v1, v2, v4, v5 y v6 corresponden a variables numéricas, c1 es una variable categórica con dos clases $(0, 1)$, mientras que v3 se compone de fechas en formato AAAA-MM-DD HH:MM:SS. Adicionalmente se registraron 1000 fechas no existentes en la columna v3.

Se observó que el conjunto de datos tiene un desvalance de clases, ya que 21,23 % de las muestras corresponden a la clase 1 y el resto a la clase 0.

Para determinar si existe alguna relación de periodicidad entre el resto de columnas y v3, se decidió realizar gráficas de dispersión entre esta variable y el resto de variables numéricas. Para mayor observabilidad se decidió centrar y escalar las muestras de cada columna numérica por medio de **StandardScaler** de sklearn, que detrás utiliza la siguiente expresión,

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j},$$

dónde μ_j y σ_j representan la media y desviación estandar de la j-esima columna.

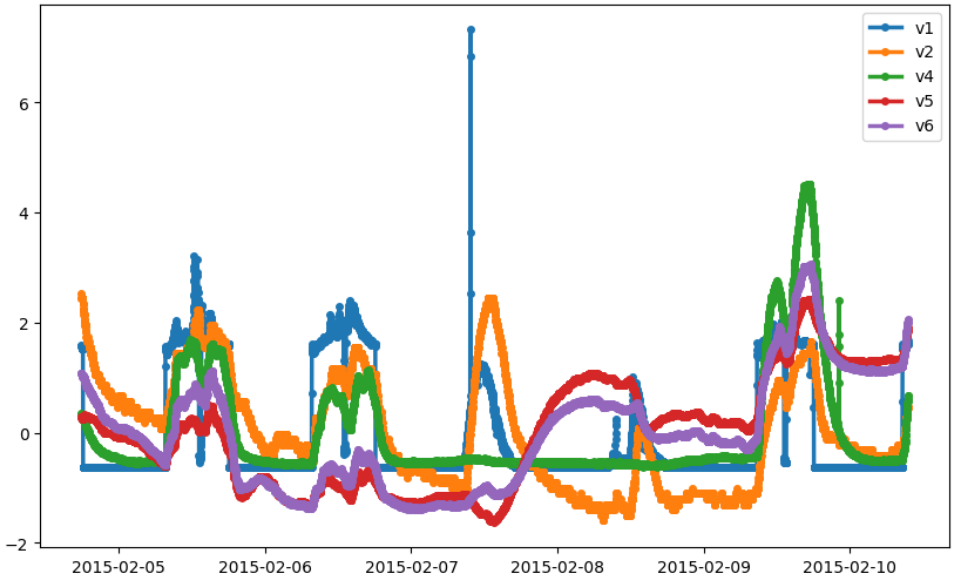

Figura 1: Dispersión de v3 vs. las variables estandarizadas v1, v2, v4, v5 y v6

Efectivamente, en la figura 1 se observa que hay una relación de periodicidad entre las columnas v3 y el resto de variables numéricas por lo que se propuso aplicar las siguientes transformaciones, considerando que la periodicidad es diaria,

$$\text{seconds\_of\_day} = \text{v3.hour} \times 3600 + \text{v3.minute} \times 60 + \text{v3.second}$$

$$\text{v3\_sin} = \sin\left(2\pi \times \frac{\text{seconds\_of\_day}}{24 \times 3600}\right)$$

$$\text{v3\_cos} = \cos\left(2\pi \times \frac{\text{seconds\_of\_day}}{24 \times 3600}\right)$$

Posteriormente, se hizo un análisis de correlación y se encontró que las variables v5 y v6 están altamente correlacionadas (ver figura 2 y 3).

3. Selección de rasgos

Para reducir la complejidad de los modelos de clustering y clasificación, así como tener las variables más relevantes para las clases en c1 se determinó hacer el **ANOVA F-test univariante** con sklearn [3], cuya es,

$$F_j = \frac{\text{Varianza entre clases}}{\text{Varianza dentro de las clases}}$$

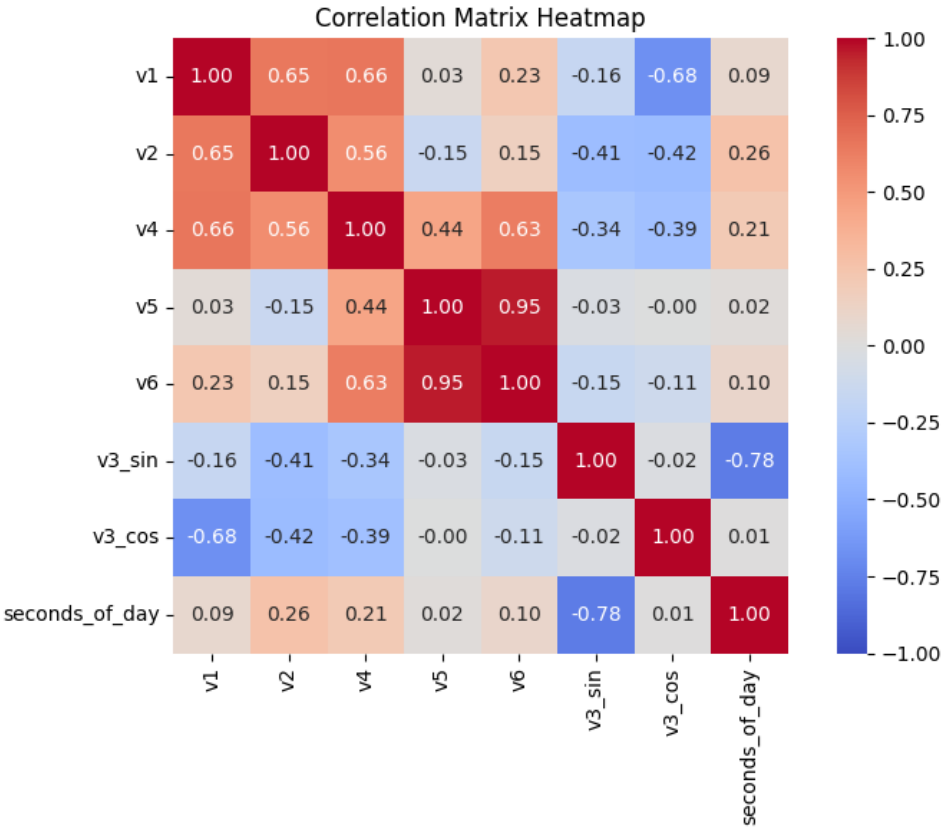

Figura 2: Correlación entre las variables v1, v2, v3_sin, v3_cos, seconds_of_day, v4, v5 y v6.

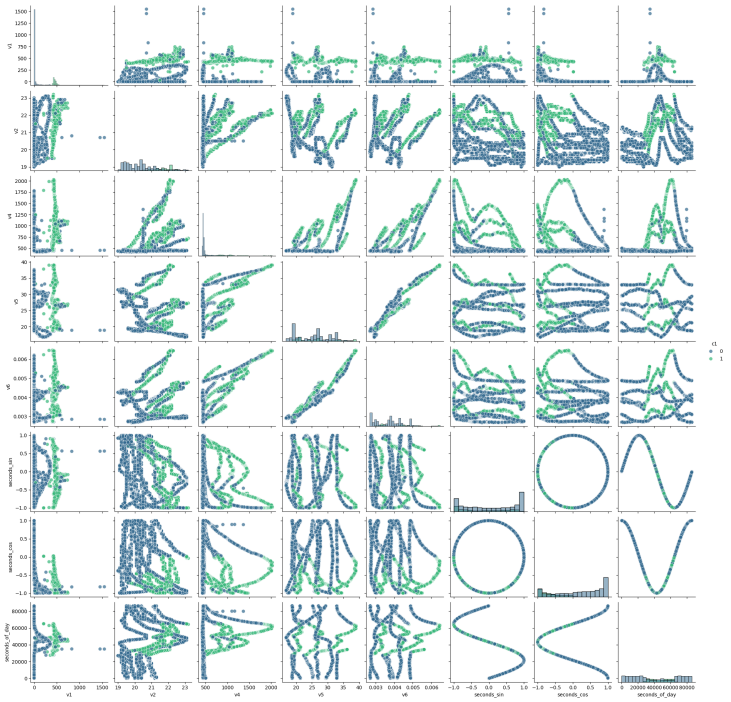

Figura 3: Dispersión entre variables dos a dos considerando la variable categórica c1

Un alto valor de F_j indica que el rasgo discrimina bien entre clases, en otro caso no hay diferencias significativas. Se el conjunto para trabajar únicamente con muestras con rasgos completos para poder realizar la prueba. En el cuadro 3 se puede observar los resultados de la prueba donde se detalla que las

variables v1, v2, v4 y v3_cos son las más significativas para c1.

Entonces el conjunto de datos utilizado en las de clustering y clasificación esta dado por la siguiente expresión,

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^4 | \mathbf{x} = [x_1, x_2, x_3, x_4]^\top : x_i \in \text{vi } \forall i \in \{1, 2, 4\},\ x_3 \in \text{v3\_cos} \right\}, \tag{1}$$

mientras que el conjunto de etiquetas esta dado por,

$$\mathcal{Y} = \{ y \in \{0, 1\} \}$$

Rasgo	**Valor F**
v1	37928.320
v2	3312.339
v4	8381.539
v5	146.309
v6	806.816
v3_sin	172.850
v3_cos	3322.118
seconds_of_day	52.700

Cuadro 2: Valor F de los ragos y transformaciones del tiempo.

4. Clustering

4.1. Reducción de dimensión

Antes de proceder al modelo de clustering, aplicó una reducción lineal de dimensión sobre los rasgos estandarizados $\hat{\mathbf{x}}$ usando la implementación de **PCA** de sklearn para pasar de cuatro a dos dimensiones, gráficamente esta transformación se puede ver en la figura 4. Se puede observar dos regiones elípticas traslapadas. El conjunto de muestras transformadas por el proceso descrito anteriormente se denota por,

$$\hat{\mathcal{X}}_{PCA} = \left\{ \hat{\mathbf{x}}_{PCA} \in \mathbb{R}^2 | \hat{\mathbf{x}}_{PCA} = [x_1, x_2]^\top : x_1, x_2 = PCA(\hat{\mathbf{x}}) \ : \hat{\mathbf{x}} = \text{StandarScaler}(\mathbf{x}) \ \forall \mathbf{x} \in \mathcal{X} \right\}. \tag{2}$$

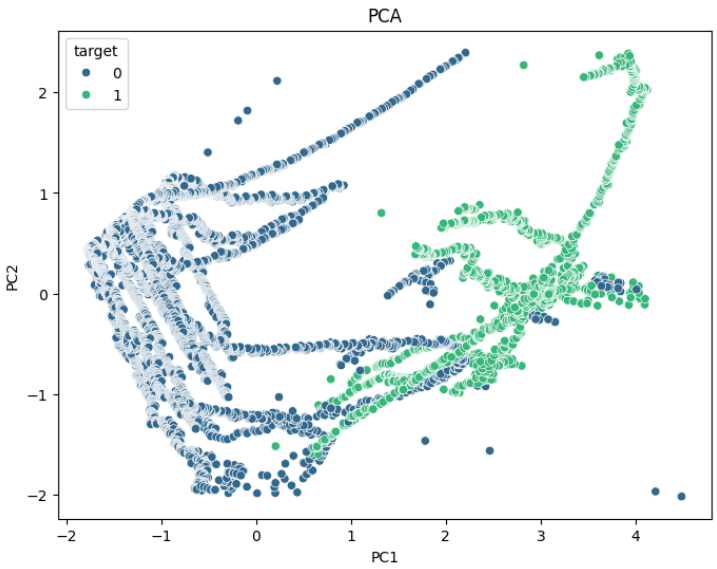

Figura 4: Reducción de dimensiones de v1, v2, v3_cos y v4 considerando la variable categórica c1

4.2. Modelo de Mezclas Gaussianas (GMM)

Debido a que las regiones encontradas en la reducción de dimensiones son elípticas y no separables, se propuso utilizar el algoritmo de mezclas gaussianas para clustering, debido a que este modelo permite definir limites suaves entre regiones formadas por el cluster de cada distribución en la mezcla.

Un modelo de mezclas gaussianas (GMM) es una superposición lineal de distribuciones gaussianas con la forma [1],

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3}$$

donde $\pi_k = p(z_k = 1)$ es la probabilidad de que la k-esima entrada de un vector binario latente $\mathbf{z}$ sea 1.

El objetivo es encontrar los parámetros $\boldsymbol{\mu}_k$ y $\boldsymbol{\Sigma}_k$ de la distribución latente, así como el coeficiente de proporción π_k cada mezcla k que mejor describa las muestras observadas. Para encontrar estos parámetros se utiliza el método EM.

Con la finalidad de facilitar el entrenamiento se fijó como hipótesis dos clusters, al igual que el número de clases disponibles en el conjunto. El conjunto $\hat{\mathbf{x}}_{PCA}$ se utilizó para optimizar el modelo de mezclas. La optimización de los parámetros de las mezclas fue realizada con la implementación del método EM para GMM de sklearn.

4.2.1. GMM como clasificador

Una vez encontrados los parámetros de la ecuación 3 para cada k, se puede predecir la pertenencia de una muestra $\mathbf{x} \in \hat{\mathcal{X}}_{PCA}$ a un cluster k por medio de encontrar la k cuyo valor $p(z_k = 1|\mathbf{x})$ sea el mayor. Usando el teorema de bayes se encuentra dicho valor [1],

$$p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

En sklearn se obtiene el cluster con mayor probabilidad para cada muestra usando,

```
labels = gmm.predict(X_pca)
```

que matemáticamente se expresa como,

$$\hat{k}_i = \arg\max_k \, p(z_k = 1|\mathbf{x}^{(i)}).$$

Debido a que GMM es un método no supervisado los clusters no necesariamente corresponden a las etiquetas reales por lo que se hace una asignación por medio del siguiente código,

```
mapping = {}
for cluster in np.unique(labels):
    mask = labels == cluster
    mapping[cluster] = mode(y[mask], keepdims=False).mode

y_pred = np.array([mapping[label] for label in labels])
```

Para cada cluster se toman todas las observaciones asignadas. Se calcula la etiqueta más frecuente entre sus correspondientes etiquetas verdaderas y. Esta etiqueta se asigna a todas las demás.

En la figura 5 pueden observarse gráficamente los resultados de esta clasificación, mientras que en las tablas 3 y 4 se reportan la matriz de confusión y las métricas de clasificación, respectivamente.

Actual / Predicted	**0**	**1**
0	5692	722
1	0	1729

Cuadro 3: Matriz de confusión del clasificador basado en GMM + PCA.

4.2.2. GMM para imputar datos

Debido a que GMM es un modelo probabilista generativo, es posible usarlo para imputar rasgos faltantes en las muestras que lo requieran. A continuación se describe el proceso para lograr este fin.

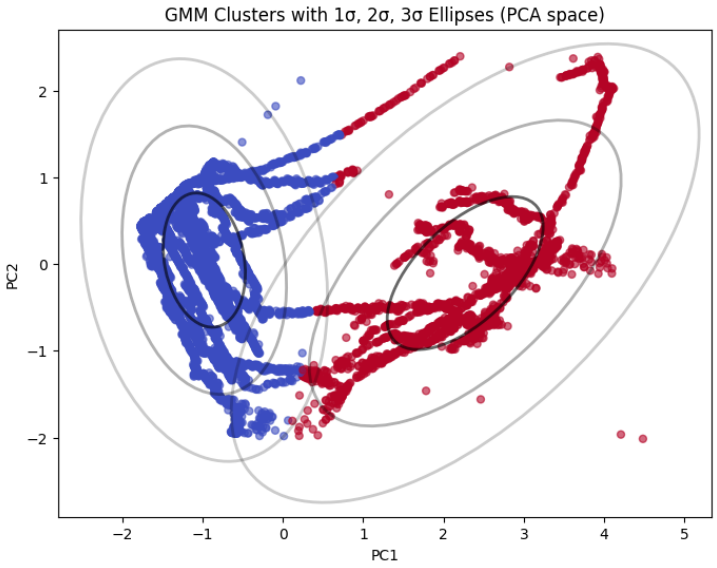

Figura 5: Muestras clasificadas y regiones elípticas asociadas a las clusters. Gráfica generada con asistencia de un LLM.

Clase	Precision	Recall	F1-score	Support
0	1.000	0.887	0.940	6414
1	0.705	1.000	0.827	1729
Accuracy	0.911			
Macro avg	0.853	0.944	0.884	8143
Weighted avg	0.937	0.911	0.916	8143

Cuadro 4: Reporte de clasificación del clasificador basado GMM + PCA.

- Paso 1: División entre rasgos observados y faltantes,

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_o \\ \mathbf{x}_m \end{pmatrix}, \quad \boldsymbol{\mu}_k = \begin{pmatrix} \boldsymbol{\mu}_{k,o} \\ \boldsymbol{\mu}_{k,m} \end{pmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{k,oo} & \boldsymbol{\Sigma}_{k,om} \\ \boldsymbol{\Sigma}_{k,mo} & \boldsymbol{\Sigma}_{k,mm} \end{pmatrix}$$

- Paso 2: Calculo de media y covarianza condicional de los datos faltantes.

$$\boldsymbol{\mu}_{k,m|o} = \boldsymbol{\mu}_{k,m} + \boldsymbol{\Sigma}_{k,mo}\boldsymbol{\Sigma}_{k,oo}^{-1}(\mathbf{x}_o - \boldsymbol{\mu}_{k,o})$$

$$\boldsymbol{\Sigma}_{k,m|o} = \boldsymbol{\Sigma}_{k,mm} - \boldsymbol{\Sigma}_{k,mo}\boldsymbol{\Sigma}_{k,oo}^{-1}\boldsymbol{\Sigma}_{k,om}$$

- Paso 3: Calculo de probabilidad posterior del componente dado las observaciones.

$$p(z_k = 1 \mid \mathbf{x}_o) = \frac{\pi_k \mathcal{N}(\mathbf{x}_o \mid \boldsymbol{\mu}_{k,o}, \boldsymbol{\Sigma}_{k,oo})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_o \mid \boldsymbol{\mu}_{j,o}, \boldsymbol{\Sigma}_{j,oo})}$$

- Imputación de datos (uso de la esperanza)

$$\mathbb{E}[\mathbf{x}_m \mid \mathbf{x}_o] = \sum_{k=1}^{K} p(z_k = 1 \mid \mathbf{x}_o)\, \boldsymbol{\mu}_{k,m|o}$$

En esta ocasión un LLM asistió en la genración de una función que realiza este proceso sobre un rasgo faltante en un cierto indice.

Debido a que el primer GMM ajustado se aplico una reducción de dimensión sobre los rasgos, no es posible retomar dicho modelo para la imputación. Por lo que se entrenó otro GMM sin aplicar PCA sobre los rasgos. Los resultados de evaluar este GMM como clasificador con respecto a las etiquetas reales se encuentran en los cuadros 5 (matriz de confusión) y 6 (métricas de clasificación).

Una vez ajustado este GMM fue posible imputar los datos faltantes en el rasgo asociado a la columna v3_cos.

Actual / Predicted	0	1
0	4936	1478
1	0	1729

Cuadro 5: Matriz de confusión de clasificador basado en GMM.

Class	Precision	Recall	F1-score	Support
0	1.000	0.770	0.870	6414
1	0.539	1.000	0.701	1729
Accuracy	0.818			
Macro avg	0.770	0.885	0.785	8143
Weighted avg	0.902	0.818	0.834	8143

Cuadro 6: Reporte de clasificación de clasificador basado en GMM.

5. Clasificación

Antes de describir el modelo utilizado para clasificación, se debe mencionar que en este caso al ser una tarea de aprendizaje supervisado se consideraron los conjuntos de entrenamiento $\mathcal{X}_{train}$ y $\mathcal{X}_{test}$ que corresponden al 80 % y 20 % del conjunto completo, respectivamente.

Se determinó usar un modelo logístico de regresión y su implementación en sklearn para la tarea de clasificación. El objetivo es entrenar un modelo tal que para $\mathbf{x}_i \in \mathcal{X}$ y la etiqueta $y_i \in \{0, 1\}$, la probabilidad de la clase positiva sea,

$$p(y_i = 1|\mathbf{x}_i; \mathbf{w}, b) = \sigma(\mathbf{w}^\top \mathbf{x}_i + b)$$

donde $\mathbf{w}$ y b son parámetros que deben ser ajustados. La optimización se realiza minimizando el siguiente costo,

$$J(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i + b) + (1 - y_i)\log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i + b))\Big] + \frac{1}{2C}\|\mathbf{w}\|_2^2$$

donde C es un parámetro utilizado en sklearn que controla la regularización inversamente. En este caso la regularización es de tipo L2, que tiene la finalidad de estabilizar los parámetros y un manejo efectivo sobre la multicolinariedad, haciendo más cercanos a cero a los parámetros correspondientes sin eliminarlos [2].

5.1. Clasificación: conjunto de datos incompleto

Actual / Predicted	0	1
0	1254	22
1	4	349

Cuadro 7: Matriz de confusión del modelo de regresión logística con datos faltantes.

Class	Precision	Recall	F1-score	Support
0	1.00	0.98	0.99	1276
1	0.94	0.99	0.96	353
Accuracy	0.98			
Macro avg	0.97	0.99	0.98	1629
Weighted avg	0.98	0.98	0.98	1629

Cuadro 8: Reporte de clasificación: modelo de regresión logística con datos faltantes.

Actual / Predicted	0	1
0	1431	19
1	1	378

Cuadro 9: Matriz de confusión del modelo dee regresión logística con datos imputados.

Class	Precision	Recall	F1-score	Support
0	1.00	0.99	0.99	1450
1	0.95	1.00	0.97	379
Accuracy		0.99		
Macro avg	0.98	0.99	0.98	1829
Weighted avg	0.99	0.99	0.99	1829

Cuadro 10: Reporte de clasificación: modelo de regresión logística con datos imputados

5.2. Clasificación: conjunto con datos imputados

6. Conclusión prueba ciencia de datos

Se hizo una limpieza, unión y procesamiento de las bases de datos en los documentos info_01.csv y info_02.csv, donde encontró que cada registro tiene los rasgos numéricos v1,v2, v4, v5 y v6, una variable de tipo fecha en v3 y una variable categórica en c1. Al hacer un análisis exploratorio se encontró 1000 fechas faltantes en v3. También se hayó un desvalance en la proporción de etiquetas 0 y 1, en favor de la clase 0. Posteriormente, se encontró que v3 guarda una relación periódica diaria con el resto de variables numéricas, por lo que se optó por transformar esta variable en una representación que plasmará esta relación (transformación basada en sin y cos). Al hacer una selección de rasgos con la implementación de ANOVA-test de sklearn se determinó que las variables más significativas para c1 son v1, v2, v4 y la variable transformada v3_cos. Para la tarea de clustering se ajusto un GMM con dos clusters sobre el conjunto de datos (incompleto) con dimensión reducida a 2 por medio de PCA. Adicionalmente se ajustó otro GMM directamente sobre los rasgos significativos con la intención de usar las propiedades de las mezclas para imputar los datos faltantes en v3_cos. Para la tarea de clasificación se entrenó un modelo de regresión logística, primero uno sobre el conjunto de datos de entrenamiento con muestras faltantes y otro sobre el conjunto de datos entrenamiento con valores imputados.

El clasificador basado en GMM + PCA muestra un accuracy global de 0.911, lo cual es un resultado aceptable para un enfoque no supervisado. Sin embargo, el F1-score para la clase 1 exhibe la notable diferencia entre el recall y precision: predice correctamente todos los casos positivos, pero genera una cantidad considerable de falsos positivos. El clasificador basado únicamente en GMM es considerablemente más deficiente (0.818 de accuracy global), la cantidad de falsos positivos es muy mayor.

Ambos modelos basados en el modelo de regresión logística muestran buenos resultados: una baja proporción de falsos positivos y negativos; un accuracy global de 0.98 y 0.99 para el modelo con datos incompletos y con datos imputados. Esto sugiere que la imputación de datos ayudó a mejorar el modelo al proporcionar más información, aunque no de forma concluyente y se recomienda probar con conjuntos de datos similares para validar esta conjetura.

7. ETL: Propuesta de solución

Para efectuar el flujo del proceso de ETL se tiene un servicio dockerizado que tiene la función principal de ejecutar un script de python con la función **etl_pipeline** 2. Esta función tiene la tarea de consultar todo el tiempo cuando hay un nuvo archivo cargado en el servidor remoto (8.8.8.8) por medio de la función **retrieve_txt_files** 3. En esta función se accede al servidor por medio de un cliente ssh con las credenciales previamente cargadas. Cuando carga un archivo lo elimina del servidor y guarda un respaldo un archivo zip. Hay un volumen dedicado a este respaldo en la instancia de docker.

```
def etl_pipeline():
    """
    ETL pipeline to process txt files and load data into the database.

    :return: None
    """
    txt_files = retrieve_txt_files()
```

```
    for file_name in txt_files:
        try:
            data = process_txt_file(file_name)
        except Exception as e:
            logger.error(f"[!] Error processing {file_name}: {e}")
            error_manager.log_error(file_name, e)
            continue
        if data is None:
            logger.warning(f"[!] Error processing {file_name}")
            continue

        correct_data = data[data["error"]]
        incorrect_data = data[~data["error"]]
        try:
            for e, row in correct_data.iterrows():
                visitor_data = transform.visitor_table(row)
                stats_data = transform.stats_table(row)

                db.count_correct_data(file_name, row)
                db.insert_visitor_data(visitor_data)
                db.insert_stats_data(stats_data)
        except Exception as e:
            logger.error(f"[!] Error processing correct data: {e}")

        try:
            for e, row in incorrect_data.iterrows():
                error_data = transform.error_table(row)
                db.count_error_data(file_name, row)
                db.insert_error_data(error_data)
        except Exception as e:
            logger.error(f"[!] Error processing incorrect data: {e}")
            error_manager.log_error(file_name, e)
            continue
```

Listing 2: Función principal ETL

```
import paramiko

def retrieve_txt_files():
    # Create an SSH client
    ## Suggested approach by Claude
    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    ssh_client.connect(hostname, port, username, password)
    sftp = ssh_client.open_sftp()
    txt_files = []
    try:
        for file in sftp.listdir(remote_path):
            if file.endswith('.txt'):
                try:
                    sftp.get(os.path.join(remote_path, file), os.path.join(local_path,
    file))
                    sftp.remove(os.path.join(remote_path, file))
                    txt_files.append(file)
                    zip_file(file)
                    logger.info(f"[*] Successfully downloaded and deleted {file}")
                except Exception as e:
                    logger.error(f"[!] Error processing {file}: {e}")
    finally:
        sftp.close()
        ssh_client.close()

    return txt_files
```

Listing 3: extractor de archivos

Una vez cargados todos los documentos disponibles en el servicio, se procede a procesar cada uno de ellos con la función **process_txt_file** para validar:

- El formato del texto (que contenga todas las columnas requeridas)

- Que los registros de email cumplan la expresión regular:

```
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
```

- Que las fechas cumplan el formato:

```
%d/%m/%Y %H:%M
```

Adicionalmente, se convierte el archivo de texto en un *dataframe*. Si existe algún error en las validaciones, se indica en una columna booleana **error** y se añade una descripción del error a otra columna llamada **error_description**. Si durante este paso hay un error de procesamiento, se registra en el *logger* con una descripción de la causa.

Posteriormente, se separan los registros con errores de aquellos que están correctos, ya que requieren un procesamiento distinto.

Sobre cada registro correcto, se transforma su correspondiente fila de observaciones con las funciones **transform.visitor_table** y **transform.stats_table**. En estas funciones se filtra y procesa el registro para las correspondientes tablas. Una vez procesadas, se insertan en la base de datos usando **SQLite** con los esquemas previamente creados en el servicio. Si durante esta etapa hay algún problema de procesamiento o guardado, el *logger* se encarga de registrar el error.

Similarmente, en el caso de los registros con errores, se procesa el registro por medio de la función **transform.error_table** para coincidir con el esquema de SQL; posteriormente, se inserta en la base de datos por medio de **db.insert_error_data**.

Las funciones db.count_correct_data y db.count_error_data tienen la función de llevar el control de la cantidad de archivos y registros en los casos correspondientes. Cada mes estás información es usada para generar reportes mensuales que indique la cantidad de errores observados, el total de registsros, así como los picos de uso de los usuarios.

El *logger* tiene la función de indicar cuando hay un error y notificar cada paso cumplido del flujo. El caso del error es muy importante pues aquí detalla la causa del error y notifica a la persona correspondiente.

Para estar en un ambiente productivo no sólo se tiene que gestionar el pipeline por medio de un contenedor de docker, si no que también provisionar un volumen para el alamacenamiento de la base de datos, el volumen del respaldo de documentos, el volumen dedicado a los logs.

Referencias

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[2] Adrien Payong and Shaoni Mukherjee. Mastering logistic regression with scikit-learn: A complete guide. DigitalOcean Community Tutorial, mar 2025. March 20, 2025.

[3] scikit-learn developers. sklearn.feature$_selection.f_classif.$, 2025. Accedido: 27-10-2025.