Reto final Álvaro López Rompinelli

Proyecto Caso de Uso , Facturación Ayuntamiento de Zaragoza

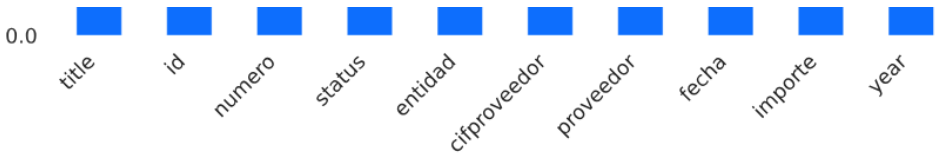
Introducción y comprensión del caso de uso

El Ayuntamiento de Zaragoza necesita optimizar la gestión y análisis de sus facturas, con el objetivo de automatizar su procesamiento. Para ello, se ha desarrollado un sistema de inteligencia artificial que permite la clasificación automática de facturas y la predicción del importe total de facturación para el próximo año.

El sistema cuenta con dos modelos de IA. El primero clasifica las facturas según su importe en tres categorías: bajo, medio o alto, lo que facilita su control y segmentación. El segundo modelo utiliza datos históricos del Ayuntamiento de Zaragoza para predecir el importe total de facturación del próximo año, proporcionando información clave para la toma de decisiones estratégicas.

El proyecto persigue varios objetivos, en primer lugar, optimiza el procesamiento de facturas, reduciendo la carga administrativa y disminuyendo el margen de error en la contabilidad. Además, automatiza la clasificación de importes, permitiendo a los responsables financieros del ayuntamiento priorizar pagos y mejorar la gestión del presupuesto. También se busca anticipar tendencias económicas mediante la predicción de ingresos futuros.

Para la realización del proyecto se ha usado una base de datos del ayuntamiento de valencia que recoge importes, proveedores y fechas entre otros.



Análisis exploratorio de datos (EDA)

El primer paso ha sido realizar un EDA (análisis exploratorio de datos),Donde se ha podido comprobar que el dataset no tiene celdas vacias o duplicadas.

Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%

El EDA también nos muestra que el dataset tiene un par de variables importantes altamente relacionadas como son el id-year o year-id. También nos enseña que la variable status esta muy desbalanceada, lo cual es entendible debido a que esta columna es categórica y este 93% de desbalance simboliza un 93% de facturas pagadas que es un valor esperado en este tipo de proyecto.

`id` is highly overall correlated with `year`	High correlation
`year` is highly overall correlated with `id`	High correlation
`status` is highly imbalanced (93.2%)	Imbalance
`id` has unique values	Unique

Ingeniería de caracteristicas

Una vez analizado el dataset se procede a la preparación de los datos.

De las diez columnas de las que dispone el dataset se procede a hacer la siguiente transformación del dataset.

Primero se elimina las variables : id, numero, status, fecha y cifproveedor, ya que no se van a utilizar en el entrenamiento de los modelos. Luego convertimos a numéricas las siguientes columnas : title, entidad y proveedor para poder operar con ellas a futuro. Finalmente se crean unas columnas nuevas : importe categoria, facturas por proveedor, importe promedio por año, importe relativo al proveedor y diferencia con media importe. Finalmente obtenemos el dataset que vamos a utilizar y lo nombramos: facturas procesadas

```
columnas_a_eliminar = ["id", "numero", "status", "fecha", "cifproveedor"]
df = df.drop(columns=columnas_a_eliminar)


encoder = LabelEncoder()
columnas_a_codificar = ["title", "entidad", "proveedor"]

for col in columnas_a_codificar:
    df[col + "_encoded"] = encoder.fit_transform(df[col])
    df = df.drop(columns=[col])

df["importe_categoria"] = pd.cut(df["importe"], bins=[0, 500, 2000, np.inf], labels=["bajo", "medio", "alto"])
df["facturas_por_proveedor"] = df.groupby("proveedor_encoded")["proveedor_encoded"].transform("count")
df["importe_promedio_por_año"] = df.groupby("year")["importe"].transform("mean")
df["importe_relativo_al_proveedor"] = df["importe"] / df.groupby("proveedor_encoded")["importe"].transform("mean")
df["diferencia_con_media_importe"] = df["importe"] - df["importe"].mean()
```

Modelado de algoritmos

Primer modelo

El primer algoritmo cumple la función de clasificar las facturas mediante machine learning; según su importe. Para ello hemos usado las librerías de la siguiente imagen.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import pickle
from imblearn.over_sampling import SMOTE
```

Lo primero que se ha hecho es eliminar filas con valores nulos, ya que han supuesto un problema para el desarrollo del algoritmo, de seguido la columna importe_categoria se transforma en string, ya que va albergar solo tipo texto, en este caso para ser más concreto solo va tener registros de : “alto””medio””bajo”. En la siguiente imagen también se observa la selección de columnas para la clasificación.S

```
# eliminar filas nulas
df = df[df["importe_categoria"] != "nan"]

# convertir a string
df["importe_categoria"] = df["importe_categoria"].astype(str)

# seleccionar  X e y
X_clasificacion = df.drop(columns=["importe", "importe_categoria"])
y_clasificacion = df["importe_categoria"]
```

En las primeras fases del modelo el entrenamiento daba un resultado de un 100% de accuracy lo cual lejos de ser algo significativo de haber realizado un buen trabajo en IA puede significar que algo malo está sucediendo con el algoritmo, como que los datos de entrenamiento y prueba son iguales de esta manera el algoritmo no aprende, solo copia de ahí el uso de las siguientes líneas de código para así evitar este resultado.

```
# verificar correlaciones con importe_categoria para evitar informacion redundante
df_encoded = df.copy()
df_encoded["importe_categoria"] = df_encoded["importe_categoria"].astype("category").cat.codes

plt.figure(figsize=(10, 6))
sns.heatmap(df_encoded.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Mapa de Calor de Correlación")
plt.show()

# eliminar alguna columna correlacionada con "importe_categoria"
columnas_eliminar = ["importe_relativo_al_proveedor", "diferencia_con_media_importe"]
X_clasificacion = X_clasificacion.drop(columns=columnas_eliminar, errors="ignore")
```

El siguiente paso es dividir el entrenamiento y la prueba que ha sido 80 y 20% respectivamente.

Aquí se aplica smote, que es una técnica utilizada en Machine Learning para **balancear datasets desbalanceados**. Se usa principalmente en problemas de **clasificación**, cuando una clase tiene muchas más muestras que otra, lo que puede hacer que el modelo aprenda patrones incorrectos o sesgados.

El motivo de aplicar smote, es que en los primeros intentos el valor “medio” se entrenaba peor debido a menos muestras de ese valor, con smote intentamos igualar todos los valores (bajo,medio,alto) a un numero similar mediante la creación de muestras sintéticas del valor minoritario. Se puede observar en la siguiente imagen.

```python
# dividir en conjunto de entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X_clasificacion, y_clasificacion, test_size=0.2, random_state=42)

# aplicar smote en el conjunto de entrenamiento
smote = SMOTE(sampling_strategy="auto", random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# manejar valores nulos en X_train
for col in X_train_smote.columns:
    if X_train_smote[col].isnull().sum() > 0:
        if X_train_smote[col].dtype == "object":
            X_train_smote[col].fillna(X_train_smote[col].mode()[0], inplace=True)
        else:
            X_train_smote[col].fillna(X_train_smote[col].mean(), inplace=True)


y_train_smote.fillna(y_train_smote.mode()[0], inplace=True)
y_train_smote = y_train_smote.astype(str)
y_test = y_test.astype(str)

# entrenar el modelo de clasificacion
modelo_clasificacion = RandomForestClassifier(n_estimators=100, random_state=42, class_weight="balanced")
modelo_clasificacion.fit(X_train_smote, y_train_smote)

# hacer predicciones y evaluar el modelo
y_pred = modelo_clasificacion.predict(X_test)
```

Luego de usar smote procedemos a entrenar el modelo y hacer las predicciones para evaluarlo.

```
Métricas de Clasificación:
Accuracy: 0.73
              precision    recall  f1-score   support

        alto       0.72      0.74      0.73       376
        bajo       0.85      0.80      0.83      1289
       medio       0.35      0.42      0.39       328
         nan       0.22      0.29      0.25         7

    accuracy                           0.73      2000
   macro avg       0.54      0.56      0.55      2000
weighted avg       0.74      0.73      0.73      2000
```

Finalmente se ha conseguido obtener una accuracy algo mas real aunque el valor medio sigue siendo peor, lo cual debería de mejorar mediante una futura implementación de mas datos por parte de la entidad proveedora de los datos.

Segundo Modelo

El siguiente modelo trata de predecir mediante machine learning el total de facturación que podría tener el año que viene 2026. Al igual que el anterior en la siguiente imagen se puede observar las librerías usadas.

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pickle
```

```python
# agrupar importes por año
df_agg = df.groupby("year")["importe"].sum().reset_index()

# crear nuevas clases para mejorar la prediccion
df_agg["importe_anterior"] = df_agg["importe"].shift(1)  # Importe del año anterior
df_agg["variacion_anual"] = df_agg["importe"] - df_agg["importe_anterior"]  # Cambio de un año a otro
df_agg["tasa_crecimiento"] = df_agg["variacion_anual"] / df_agg["importe_anterior"]  # % de crecimiento
df_agg = df_agg.fillna(0)
```

En la pasada imagen se puede observar la creación de diferentes clases como son importe del año anterior, la variación anual y la tasa de crecimiento.

```python
# crear nuevas variables  X e y
X_regresion = df_agg[["year", "importe_anterior", "variacion_anual", "tasa_crecimiento"]]
y_regresion = df_agg["importe"]

# dividir en conjunto de entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X_regresion, y_regresion, test_size=0.2, random_state=42)

# entrenar el modelo de regresion RandomForest
modelo_regresion = RandomForestRegressor(n_estimators=100, random_state=42)
modelo_regresion.fit(X_train, y_train)

# evaluacion del modelo
y_pred = modelo_regresion.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f" MAE: {mae:.2f}")
print(f" RMSE: {rmse:.2f}")
print(f" R² Score: {r2:.2f}")



# predecir el importe total del proximo año
proximo_anio = df_agg["year"].max() + 1
ultimo_importe = df_agg["importe"].iloc[-1]
variacion = df_agg["variacion_anual"].iloc[-1]
tasa = df_agg["tasa_crecimiento"].iloc[-1]

entrada_nueva = pd.DataFrame([[proximo_anio, ultimo_importe, variacion, tasa]],
                             columns=["year", "importe_anterior", "variacion_anual", "tasa_crecimiento"])
prediccion = modelo_regresion.predict(entrada_nueva)

print(f" Predicción de Importe Total para el año {proximo_anio}: €{prediccion[0]:,.2f}")
```

En la imagen pasada se crean las variables x e y, se dividen además al igual que el anterior modelo en 80-20%. Para este modelo hemos usado el modelo de regresión Random forest ya que suele dar resultados con buena presicion, es resistente a valores atípicos y datos faltantes además de manejar datos numéricos y categóricos sin problemas.

Luego de evaluar el modelo se procede a hacer una predicción del importe del próximo año 2026.

```
MAE: 5301872.77
RMSE: 5752853.80
R² Score: 0.13
Predicción de Importe Total para el año 2026: €13,732,951.33
```

En orden, podemos observar el error medio absoluto con un valor de 5301872.77 y la raíz del error cuadrático medio con valor de 5752853.80, ambos valores dan un resultado muy elevado debido a que los valores que se tratan en el dataset soy bastante elevados. El R2 score da un valor muy bajo debido a la poca cantidad de datos, la cual, al ir incrementando el r2 score irá aumentando y siendo mas preciso.