Entrega final: Inteligencia Artificial paralas Ciencias e Ingenierías

Introducción

La siguiente producción académica tiene como objetivo el acercamiento a la manipulación de datos y entrenamiento de modelos con la inteligencia artificial. Todo ello en el marco de la asignatura Introducción a la inteligencia artificial para las ingenierías.

Para este caso en específico, se usó un data set extraído de Kaggle - con características especiales, las cuales serán descritas más adelante - con el fin de simular un caso real de aplicación práctica. En dicho data set el problema planteado se resume en la clasificación de información de solicitantes de crédito hipotecario, con el fin de determinar si cumplen, o no, con un margen de probabilidad suficiente de pago y devolución del préstamo. Sobre este dataset se exploraron diferentes opciones de ordenamiento de datos, filtrado, y creación de modelos predictivos.

Los modelos predictivos desarrollados fueron básicamente dos, los cuales ayudaron a comprender acerca de cómo es posible entrenar el software, alimentándolo con cientos de datos, para que pudiera arrojar probabilidades. Para el caso que se presentará a continuación, se usaron los modelos de regresión logística y el modelo de *support vector machie SVM,* ya que ambos podían resolver el problema planteado.

La regresión logística se usó para hallar relaciones entre los datos, permitiendo hacer predicciones basándose en el comportamiento de las mismas. Por su parte, el SVM se exploró para ver qué tipos de clasificaciones se generaban en diferentes regiones.

Finalmente, después de varios justes a los modelos, se logró cuantificar cuántas personas estarían aptas para recibir el crédito, y los parámetros de precisión se analizarán en las conclusiones.

Exploración descriptiva del dataset

Para este trabajo académico, se planteó que el data set debía contar con las siguientes características:

- Al menos 5000 instancias (filas, imágenes, etc.)
- Al menos 30 columnas
- Al menos el 10% de las columnas han de ser categóricas
- Al menos ha de tener un 5% de datos faltantes en al menos 3 columnas.

Por ello, se eligió uno que consta de alrededor de 108 columnas, y 48.000 instancias (filas). De allí, tenemos columnas categóricas como CODE_GENDER (masculino, femenino, 0 - 1), OWN_CAR (Y – N, 0 - 1), y también otros de carácter no discreto como INCOME_TOTAL (ingresos en integer - float), DAYS_EMPLOYED, CAR_AGE, etc. A continuación, se deja una muestra de los datos para hacerse una idea:

NAME_CO	CODE_GEI	FLAG_OW	FLAG_OW	CNT_CHIL	AMT_INC	AMT_CRE	AMT_ANN	AMT_GOO	NAME_TYI	NAME_IN	NAME_ED	NAME_FA	NAME_HO	REGION_P	DAYS_BIR	DAYS_EM	DAYS_REG	DAYS_ID_I	OWN_CAF	FLAG_MOI	FLAG_EMF
Cash loans	F	N	Y	0	135000	568800	20560.5	450000	Unaccomp	Working	Higher edu	Married	House / ap	0.01885	-19241	-2329	-5170	-812		1	1
Cash loans	M	N	Y	0	99000	222768	17370	180000	Unaccomp	Working	Secondary	Married	House / ap	0.035792	-18064	-4469	-9118	-1623		1	1
Cash loans	M	Y	Y	0	202500	663264	69777	630000		Working	Higher edu	Married	House / ap	0.019101	-20038	-4458	-2175	-3503	5	1	1
Cash loans	F	N	Y	2	315000	1575000	49018.5	1575000	Unaccomp	Working	Secondary	Married	House / ap	0.026392	-13976	-1866	-2000	-4208		1	1
Cash loans	M	Y	N	1	180000	625500	32067	625500	Unaccomp	Working	Secondary	Married	House / ap	0.010032	-13040	-2191	-4000	-4262	16	1	1
Cash loans	F	Y	Y	0	270000	959688	34600.5	810000	Unaccomp	State serv	Secondary	Married	House / ap	0.025164	-18604	-12009	-6116	-2027	10	1	1
Cash loans	M	Y	Y	2	180000	499221	22117.5	373500	Unaccomp	Working	Higher edu	Married	House / ap	0.0228	-16685	-2580	-10125	-241	3	1	1
Cash loans	M	N	Y	0	166500	180000	14220	180000	Unaccomp	Working	Higher edu	Single / no	With parer	0.005144	-9516	-1387	-5063	-2055		1	1
Cash loans	F	N	Y	0	315000	364896	28957.5	315000	Unaccomp	State serv	Higher edu	Married	House / ap	0.04622	-12744	-1013	-1686	-3171		1	1
Cash loans	F	Y	Y	1	162000	45000	5337	45000	Family	Working	Higher edu	Civil marri	House / ap	0.018634	-10395	-2625	-8124	-3041	5	1	1
Cash loans	F	N	Y	0	67500	675000	25447.5	675000	Unaccomp	Pensioner	Secondary	Married	House / ap	0.003122	-23670	365243	-7490	-4136		1	0
Cash loans	F	N	Y	0	135000	261621	16848	216000	Unaccomp	Working	Secondary	Widow	House / ap	0.008019	-15524	-3555	-7833	-3985		1	1
Cash loans	F	N	Y	0	247500	296280	23539.5	225000	Unaccomp	Working	Secondary	Civil marri	House / ap	0.018634	-12278	-929	-6031	-4586		1	1
Cash loans	F	Y	Y	0	90000	360000	18535.5	360000	Unaccomp	Working	Secondary	Married	House / ap	0.01452	-19687	-3578	-10673	-3183	3	1	1
Revolving	M	N	Y	0	180000	157500	7875	157500	Spouse, pa	Working	Secondary	Civil marri	House / ap	0.031329	-12091	-1830	-1042	-4221		1	1
Cash loans	M	Y	Y	0	180000	296280	21690	225000	Unaccomp	Working	Secondary	Civil marri	House / ap	0.032561	-13563	-1007	-5719	-4044	14	1	1
Cash loans	F	Y	Y	0	202500	407520	26041.5	360000	Unaccomp	Working	Secondary	Single / no	House / ap	0.00702	-19375	-4739	-3035	-2895	11	1	1
Cash loans	M	Y	Y	0	90000	499221	22117.5	373500	Unaccomp	Pensioner	Secondary	Married	House / ap	0.003122	-22705	365243	-5107	-5990	15	1	0
Cash loans	F	Y	Y	1	225000	431280	23526	360000	Unaccomp	Commerci	Higher edu	Civil marri	With parer	0.025164	-10962	-1883	-99	-1721	8	1	1
Cash loans	F	Y	Y	0	175500	478498.5	46741.5	454500	Family	Working	Secondary	Married	House / ap	0.010147	-17955	-305	-10087	-1516	10	1	1
Cash loans	F	N	Y	0	99000	225000	19242	225000	Unaccomp	Commerci	Secondary	Civil marri	House / ap	0.007274	-10507	-2780	-5072	-2729		1	1
Cash loans	F	N	Y	0	157500	266652	16443	202500	Unaccomp	Working	Higher edu	Widow	House / ap	0.011657	-22557	-13294	-13154	-4187		1	1
Cash loans	F	N	Y	0	135000	540000	27702	540000	Family	Pensioner	Higher edu	Separated	House / ap	0.072508	-22784	365243	-5103	-5229		1	0
Cash loans	M	N	Y	1	337500	1313213	42493.5	1012500	Unaccomp	Commerci	Incomplet	Married	House / ap	0.02461	-11948	-1415	-5611	-4052		1	1
Cash loans	M	Y	Y	0	157500	539100	29245.5	450000	Unaccomp	Working	Secondary	Married	House / ap	0.031329	-13172	-3518	-6207	-4243	20	1	1
Cash loans	F	N	Y	0	76500	225000	12334.5	225000	Unaccomp	Working	Secondary	Civil marri	House / ap	0.010966	-15394	-967	-5389	-5433		1	1
Cash loans	F	N	Y	0	112500	256032	10759.5	180000	Unaccomp	Pensioner	Secondary	Widow	House / ap	0.019689	-21150	365243	-4157	-4658		1	0
Cash loans	F	N	Y	0	225000	501363	25726.5	418500	Family	Commerci	Higher edu	Married	House / ap	0.031329	-21040	-2467	-918	-3663		1	1

El data set completo puede ser explorado directamente desde este hipervínculo, o copiando y pegando este enlace: https://www.kaggle.com/competitions/home-credit-default-risk/overview

Problema predictivo a resolver:

Una entidad bancaria desea predecir el nivel de riesgo al que se encuentra expuesta con cada crédito que otorga a sus usuarios. Esto es importante, debido a que un grueso de la población no tiene acceso a los servicios bancarios. Por ende, no posee historial crediticio, o datos que corroboren su intención y capacidad de pago.

A raíz de esta situación, muchos prestamistas malintencionados se han aprovechado de ello, y han otorgado créditos falsos, o con altos intereses, a la población, haciendo que la brecha por acceder a servicios financieros formales sea más alta.

Es allí donde un modelo confiable y seguro se hace indispensable. Esta entidad bancaria no puede comprometer sus operaciones, por lo que tampoco puede asumir un gran riesgo al otorgar créditos. Si llegase a desembolsar sumas de dinero a personas con altas probabilidades de no pagar a tiempo, podría caer rápidamente en bancarrota.

Con el ánimo de obtener más clientes certeros (que paguen o con poca tendencia a no pagar), y evitar evaluar mal a un prospecto a crédito, se crea este modelo.

Dado que es un caso

El dataset a utilizar posee información en diferentes aspectos de cada uno de los prospectos del crédito. Allí se puede ver información personal tal como:

- Edad
- Género
- Nivel de estudios
- Si tiene vehículo propio
- Número de hijos
- Total de ingresos anuales
- Estado civil
- Días laborados o con trabajo fijo
- Sector en el que trabaja

Entre otros datos de interés para el banco o entidad crediticia. Dado que este es un caso donde los datos de entrada son varias de estas categorías, y el dato de salida pretende ser una evaluación específica en “apto para crédito” o “rechazado para crédito”, el caso se vuelve clasificatorio.

En resumen, se puede decir que el problema a resolver y evaluar es: dadas las condiciones de un usuario, como su nivel de ingresos, educación, bienes, etc, vamos a predecir si es apto para un crédito, o no, según el riesgo que puede asumir x entidad financiera que presta dinero.

Dado que se necesita predecir dos condiciones (si es apto, o no), se puede decir que es un problema clasificatorio o de clasificación binaria. De hecho, esta fue la razón por la que se usó un una regresión clasificatoria, y un Support Vector Machine, SVM, visto en los videos de clase.

Esta última no se consideró tan apta, ya que, según lo visto en el contenido del curso, es mucho más efectiva cuando la frontera de decisión entre tus dos clases es lineal, es decir, una línea recta, plano, o hiperplano puede separar tus clases. Como existe una cantidad considerable de características, es probable que no sea tan fácil separar a los individuos en pocas clases. Aun así, se realizaron algunas pruebas con este método.

Para las iteraciones, tanto de la regresión clasificatoria, como del SVM, se usó una matriz modificada de la original, ya que esta es muy grande y contiene demasiados datos. Se usó en su lugar una muestra de 1000 filas para entrenamiento, 5000 filas para predicciones, y las columnas más representativas del dataset original, las cuales son:

“CAR_OWN_AGE', = posee vehículo

'CNT_CHILDREN' = cantidad de hijos

'AMT_INCOME_TOTAL' = ingresos totales

'DAYS_EMPLOYED' = días como trabajador

'CNT_FAM_MEMBERS' = cantidad de miembros de la familia

'BASEMENTAREA_AVG' = área del inmueble

'YEARS_BUILD_AVG' = años de construido el inmueble

'COMMONAREA_AVG' = cantidad de áreas comunes

'FLOORSMAX_AVG' = Cantidad de pisos

'CODE_GENDER' = género masculine o femenino

'FLAG_OWN_CAR' = Si es propietario de un vehículo

'FLAG_OWN_REALTY' = Si posee bienes

'NAME_INCOME_TYPE' = Tipo de ingresos o labor

'NAME_EDUCATION_TYPE' = nivel educativo

'NAME_FAMILY_STATUS' = estado civil

'OCCUPATION_TYPE'] = cargo de trabajo

De aquí tenemos que se tuvo que separar la información en variables categóricas y no categóricas, quedando así:

No categóricas: CAR_OWN_AGE', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'DAYS_EMPLOYED',

'CNT_FAM_MEMBERS', 'BASEMENTAREA_AVG', 'YEARS_BUILD_AVG',

'COMMONAREA_AVG', 'FLOORSMAX_AVG']

Categóricas : ['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_INCOME_TYPE',

'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'OCCUPATION_TYPE']

Iteraciones de desarrollo

Preprocesado de datos - Depuración y limpieza

El data set seleccionado contenía varias casillas sin información (vacías), y oras con diferentes tipos de datos, como los mencionados líneas arriba (discretos, no discretos, categóricos, no categóricos). Para ello, se realizó una conversión y clasificación de datos, con el fin de depurarlos para el entrenamiento del modelo. Esto se logró con las propiedades que se le puede dar a los datos con Python:

```
# Asignar 0 a CAR_OWN_AGE donde FLAG_OWN_CAR es 'N'
data.loc[data['FLAG_OWN_CAR'] == 'N', 'CAR_OWN_AGE'] = 0

# Rellenar los valores faltantes
num_columns = ['CAR_OWN_AGE', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'DAYS_EMPLOYED',
               'CNT_FAM_MEMBERS', 'BASEMENTAREA_AVG', 'YEARS_BUILD_AVG',
               'COMMONAREA_AVG', 'FLOORSMAX_AVG']
cat_columns = ['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_INCOME_TYPE',
               'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'OCCUPATION_TYPE']

for col in num_columns:
    data[col].fillna(0, inplace=True)

for col in cat_columns:
    data[col].fillna('No aplica', inplace=True)

return data[num_columns + cat_columns], num_columns, cat_columns
```

Modelos supervisados

Modelo supervisado de regresión logística

1. **Imports:** se importaron las librerías pertinentes

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

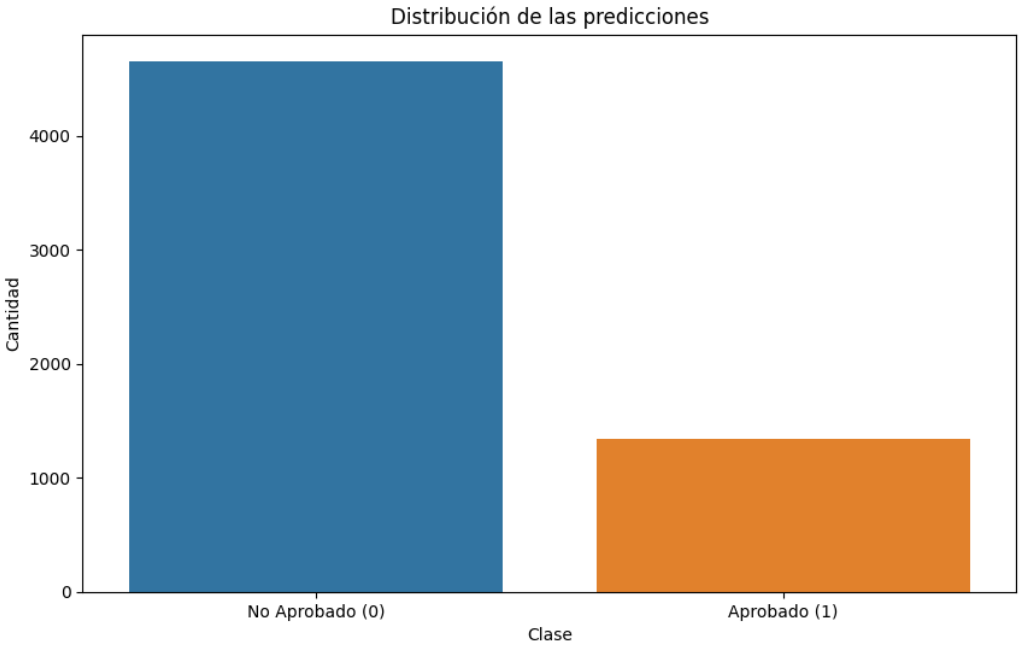
2. **Clasificación de datos y entrenamiento:** en este punto se limpiaron los datos, especialmente aquellos que arrojaron erro de NaN (not a number), y se hizo el casting respectivo, o el llenado de valores.

```python
# Función para preprocesar los datos
def preprocess_data(data):
    # Convertir DAYS_EMPLOYED a positivo
    data['DAYS_EMPLOYED'] = data['DAYS_EMPLOYED'].abs()

    # Asignar 0 a CAR_OWN_AGE donde FLAG_OWN_CAR es 'N'
    data.loc[data['FLAG_OWN_CAR'] == 'N', 'CAR_OWN_AGE'] = 0

    # Rellenar los valores faltantes
    num_columns = ['CAR_OWN_AGE', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'DAYS_EMPLOYED',
                   'CNT_FAM_MEMBERS', 'BASEMENTAREA_AVG', 'YEARS_BUILD_AVG',
                   'COMMONAREA_AVG', 'FLOORSMAX_AVG']
    cat_columns = ['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_INCOME_TYPE',
                   'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'OCCUPATION_TYPE']

    for col in num_columns:
        data[col].fillna(0, inplace=True)

    for col in cat_columns:
        data[col].fillna('No aplica', inplace=True)

    return data[num_columns + cat_columns], num_columns, cat_columns

# Cargando el dataset de entrenamiento y preprocesamiento
dataset_train = pd.read_csv('datasetv2train.csv')
X_train, num_columns, cat_columns = preprocess_data(dataset_train)
y_train = dataset_train['target']

# Preprocesador y modelo
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_columns),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), cat_columns)
    ]
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=0))
])
```

3. **Entrenamiento:** se anexó la columna “target” para que el modelo hiciera las predicciones en un nuevo archivo.

```python
# Entrenamiento del modelo
pipeline.fit(X_train, y_train)

# Cargando el dataset de predicciones y preprocesamiento
dataset_pred = pd.read_csv('datasetv2.csv')
X_pred, _, _ = preprocess_data(dataset_pred)

# Haciendo predicciones
predictions = pipeline.predict(X_pred)

# Añadiendo las predicciones al dataset
dataset_pred['target'] = predictions

# Guardando las predicciones en un nuevo archivo CSV
dataset_pred.to_csv('datasetv2_with_predictions.csv', index=False)

print("Predicciones completadas y guardadas en 'datasetv2_with_predictions.csv'")

# Gráfica de las predicciones
plt.figure(figsize=(10, 6))

# Contar la cantidad de predicciones para cada clase
sns.countplot(x='target', data=dataset_pred)

plt.title('Distribución de las predicciones')
plt.xlabel('Clase')
plt.ylabel('Cantidad')
plt.xticks(ticks=[0,1], labels=['No Aprobado (0)', 'Aprobado (1)'])

plt.show()
```

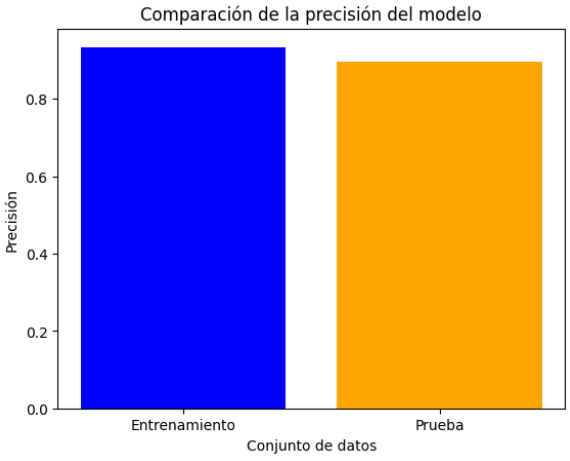
4. **Gráficos:** Por último, se generó una gráfica que mostrara, de forma básica, cuantos individuos están opcionados al crédito.



5. **Hiperparámetros y confiabilidad:** También se analizó el hiperparámetro, y el grado de confiabilidad. Esto se logró dividiendo el conjunto de datos de entrenamiento y el de prueba, procesándolos, y haciendo una comparación de precisión (accuracy).

```python
# Dividir el conjunto de datos original en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(
    dataset_train.drop('target', axis=1),  # Asumiendo que 'target' es la columna objetivo
    dataset_train['target'],
    test_size=0.2,  # Porcentaje del conjunto de datos a usar como prueba
    random_state=0
)

# Preprocesar los conjuntos de entrenamiento y prueba
X_train_processed, num_columns, cat_columns = preprocess_data(X_train)
X_test_processed, _, _ = preprocess_data(X_test)

# Reentrenar el modelo con los mejores hiperparámetros encontrados
best_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=0))
])

best_pipeline.fit(X_train_processed, y_train)

# Evaluar en el conjunto de entrenamiento
train_predictions = best_pipeline.predict(X_train_processed)
train_accuracy = accuracy_score(y_train, train_predictions)
print("Precisión en el conjunto de entrenamiento:", train_accuracy)

# Evaluar en el conjunto de prueba
test_predictions = best_pipeline.predict(X_test_processed)
test_accuracy = accuracy_score(y_test, test_predictions)
print("Precisión en el conjunto de prueba:", test_accuracy)

# Diagnóstico de overfitting o bias
if train_accuracy > test_accuracy:
    if train_accuracy - test_accuracy > 0.1:  # Umbral de diferencia
        print("Posible overfitting.")
    else:
        print("Diferencia aceptable, no hay signos claros de overfitting.")
else:
    print("El modelo puede tener un alto sesgo.")

# Visualización de las métricas de rendimiento
plt.bar(['Entrenamiento', 'Prueba'], [train_accuracy, test_accuracy], color=['blue', 'orange'])
plt.xlabel('Conjunto de datos')
plt.ylabel('Precisión')
plt.title('Comparación de la precisión del modelo')
plt.show()
```

El resultado es una gráfica que nos muestra la dicha comparación de datos de entrenamiento vs datos de prueba, dando un rango por encima del 90% (cercano al 1, que sería el 100% de precisión):




```
Precisión en el conjunto de entrenamiento: 0.9336670838548186
Precisión en el conjunto de prueba: 0.895
Diferencia aceptable, no hay signos claros de overfitting.
```

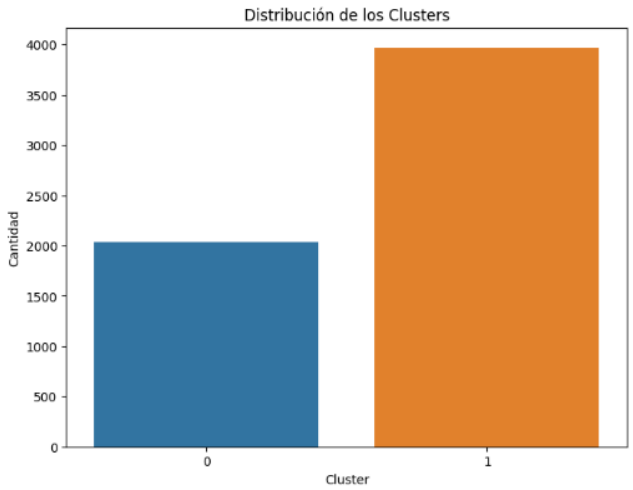
Este código puede verse en detalle en este hipervículo que lleva a Colab, o en el siguiente enlace: https://colab.research.google.com/drive/1yOs4fvB0Nd-vrvWJ7jJgzUIQVorwHb29

Regresión logística no supervisada

se intentó realizar un modelo de regresión logística no supervisada, que se mostrará a continuación. En este, se cargaron los datos como en el anterior, haciendo la respectiva depuración y limpieza de los mismos. Pero la diferencia radicó en el tipo de modelo, donde se suprimió la columna “target”, y se codificó de tal manera que se usaran clusters o agrupaciones de datos por similitud, usando el K-means implícito en Python.

Esto dio como resultado dos cluster con similitud a lo obtenido en el modelo supervisado. Pero, al no contar con tanta precisión, se concluye que el modelo supervisado es más efectivo para los objetivos de este trabajo académico.

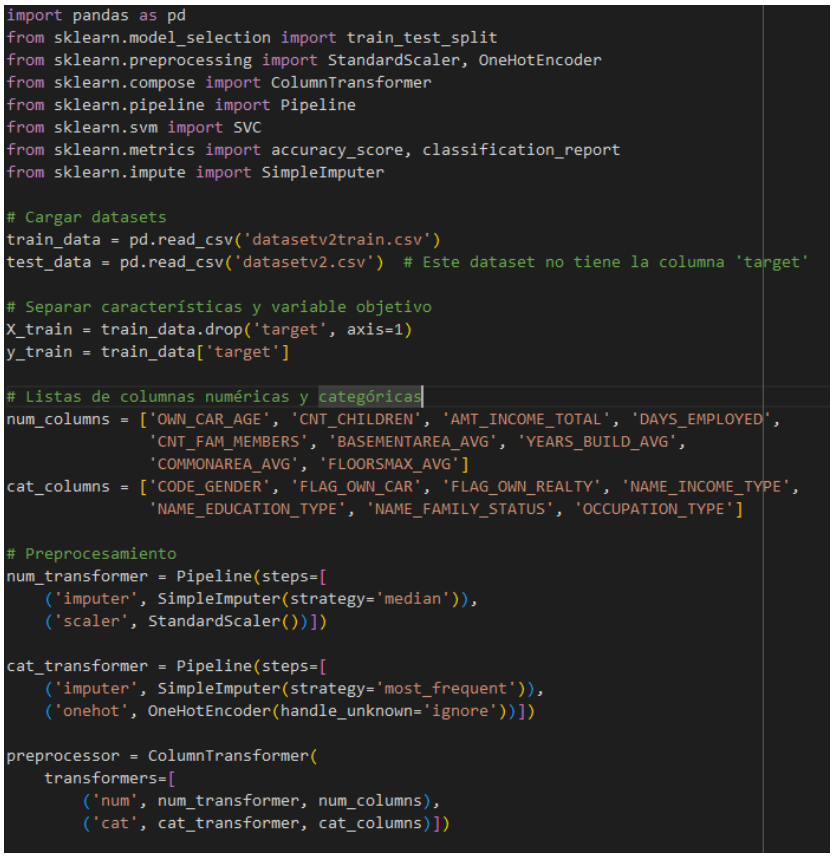
```python
# Crear un pipeline con el preprocesador
pipeline = Pipeline(steps=[('preprocessor', preprocessor)])

# Preprocesar los datos
X_processed = pipeline.fit_transform(X)

# Definir el modelo K-means con un número arbitrario de clusters
kmeans = KMeans(n_clusters=2, random_state=0)

# Ajustar el modelo a los datos preprocesados
kmeans.fit(X_processed)

# Etiquetas de cluster
clusters = kmeans.labels_

# Añadir las etiquetas de cluster al dataframe original para un análisis más detallado
dataset['Cluster'] = clusters

# Guardar el dataframe con las etiquetas de cluster en un archivo CSV
dataset.to_csv('/datasetv2_with_clusters.csv', index=False)

# Visualizar la distribución de los clusters
plt.figure(figsize=(8, 6))
sns.countplot(x='Cluster', data=dataset)
plt.title('Distribución de los Clusters')
plt.xlabel('Cluster')
plt.ylabel('Cantidad')
plt.show()

print("Clustering completado y guardado en 'datasetv2_with_clusters.csv'")
```



Modelo de Support Vector Machine SVM

Este modelo actúa mejor cuando hay grupos claramente diferenciables por una recta en el espacio trazado o en el hiperplano, de acuerdo a sus características. Aun así, se consideró una opción para contrastar resultados.

1. **Carga de datos:** al igual que el modelo anterior, haremos una carga de datos con su respectiva depuración y limpieza:



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer

# Cargar datasets
train_data = pd.read_csv('datasetv2train.csv')
test_data = pd.read_csv('datasetv2.csv')  # Este dataset no tiene la columna 'target'

# Separar características y variable objetivo
X_train = train_data.drop('target', axis=1)
y_train = train_data['target']

# Listas de columnas numéricas y categóricas
num_columns = ['OWN_CAR_AGE', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'DAYS_EMPLOYED',
               'CNT_FAM_MEMBERS', 'BASEMENTAREA_AVG', 'YEARS_BUILD_AVG',
               'COMMONAREA_AVG', 'FLOORSMAX_AVG']
cat_columns = ['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_INCOME_TYPE',
               'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'OCCUPATION_TYPE']

# Preprocesamiento
num_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

cat_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_columns),
        ('cat', cat_transformer, cat_columns)])
```

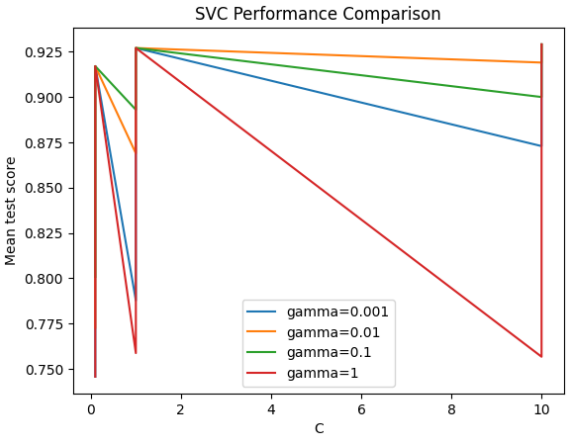
2. **Hacer predicciones**: una vez tenemos los datos cargados, se harán las predicciones de acuerdo a la naturaleza de los mismos, y se guardarán en un nuevo documento denominado “predictions”. Esto nos da como resultado una tabla, donde la columna target se llenará de ceros o unos, siendo los primeros “no aprobado para crédito”, y “aprobados para crédito” los segundos.

_FAMILY_STATUS	DAYS_EMPLOYED	OCCUPATION_TYPE	CNT_FAM_MEMBERS	BASEMENTAREA_AVG	YEARS_BUILD_AVG	COMMONAREA_AVG	FLOORSMAX_AVG	target
/ not married	-185	Sales staff	1	0.1012	0.7484		0.1667	1
/	-11883		1	0.1158	0.7144		0.1667	1
d	-5664		4	0.0564	0.762	0.0044	0.3333	1
d	-4516	Sales staff	4					0
d	-680	Sales staff	4	0.084	0.9252	0.0034	0.3054	0
/ not married	-240	Managers	1					1
d	-410	Laborers	3	0.0935	0.7348		0.1667	0
d	-5025	Medicine staff	2					0
d	-837	Sales staff	2					0
d	-2899	Drivers	3	0.1336			0.125	0

Show 10 per page 1 10 19 20 21 30 100 500 590 600

3. **Hiperparámetros y precisión:** para este caso también se verificó este apartado, con valores *gamma* de diferente rango. Como se explica en los videos de clase, un gama más alto no es necesariamente el mejor.

```python
# Cargar datasets
train_data = pd.read_csv('datasetv2train.csv')
test_data = pd.read_csv('datasetv2.csv')  # Este dataset no tiene la columna 'target'

# Separar características y variable objetivo
X_train = train_data.drop('target', axis=1)
y_train = train_data['target']

# Crear el pipeline del modelo
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', SVC(probability=True))])

# Definir el espacio de búsqueda de hiperparámetros
param_grid = {
    'classifier__C': [0.1, 1, 10],
    'classifier__gamma': [0.001, 0.01, 0.1, 1],
    'classifier__kernel': ['rbf', 'linear']
}

# Realizar la búsqueda de hiperparámetros
grid_search = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Mejores hiperparámetros
best_params = grid_search.best_params_
print('Mejores hiperparámetros:', best_params)

# Resultados en una DataFrame
results = pd.DataFrame(grid_search.cv_results_)

# Visualizar los resultados
# Podemos graficar, por ejemplo, cómo el parámetro C afecta la precisión media de validación
for gamma in param_grid['classifier__gamma']:
    subset = results[results['param_classifier__gamma'] == gamma]
    plt.plot(subset['param_classifier__C'], subset['mean_test_score'], label=f'gamma={gamma}')

plt.xlabel('C')
plt.ylabel('Mean test score')
plt.legend()
plt.title('SVC Performance Comparison')
plt.show()
```



Este código puede verse en detalle en este hipervículo que lleva a Colab, o en el siguiente enlace: https://colab.research.google.com/drive/1ze4NZkQvdJhXqqTQdT6KU0rNnW-2izQl

Retos y consideraciones de despliegue

El despliegue de un proyecto de inteligencia artificial que utiliza la regresión clasificatoria y máquinas de vectores de soporte (SVM) para evaluar la elegibilidad al crédito presenta varios retos y consideraciones. Uno de los principales desafíos es garantizar la precisión y la generalización del modelo, especialmente cuando se analizan conjuntos de datos complejos como los 5000 casos estudiados en este proyecto. La calidad y la relevancia de los datos son críticas, y los parámetros como la edad, los ingresos, el número de hijos y la posesión de vehículo deben ser seleccionados y ponderados cuidadosamente para evitar sesgos en la clasificación de los solicitantes como aptos o no aptos.

Además, la implementación de estos modelos en entornos de producción requiere un marco sólido de validación y pruebas para asegurar su robustez y fiabilidad. Las consideraciones éticas también son fundamentales, ya que los modelos no deben discriminar a los individuos basándose en características no relacionadas con su capacidad de pago. Esto implica una revisión constante y una posible recalibración del modelo a medida que se recopila más información y se identifican nuevas tendencias.

Conclusiones

Tras lo observado, se puede concluir que este ejercicio logró un acercamiento a lo que es un modelamiento real de la industria, aunque de manera más básica, logrando los siguientes resultados:

- Aproximadamente 1300 peticiones o individuos serían aprobados en caso de solicitar un crédito, de cada 5000, representado cerca del 26% del total de casos.
- Esta tendencia puede extenderse a un rango más grande de la población, por lo que los créditos seguirían siendo un producto no accesible para la mayoría.
- Aunque la cifra de créditos sea baja para los candidatos, esto garantiza que quienes son aprobados cuenten con el suficiente respaldo como par a hacerle frente a la deuda.

En cuanto al ejercicio académico como tal, es de concluir que lo ejecutado puso a prueba los conocimientos de Python, y los temas vistos en los videos del curso, sobre la generación de modelo y ajuste de los mismos, especialmente lo que tiene que ver con limpieza y depuración de datos, ya que de ello consiste gran parte del éxito de las predicciones.