ITESM- Campus Puebla

# *American Express - Default Prediction*

**Inteligencia artificial avanzada para la ciencia Datos**


**Integrantes Equipo 1:**

Myroslava Sánchez Andrade A01730712
José Antonio Bobadilla García A01734433
Karen Rugerio Armenta A01733228
Alejandro Castro Reus A01731065


Fecha: 14/09/2022

# Guía para el profesor

| Rubros | Indicadores | Referencia al reporte | Referencia al código |
|---|---|---|---|
| Datos | Limpia los datos con ETLs o una herramienta equivalente de manera correcta | Sección 3.2 - Data Preparation | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/etl |
| | Transforma los datos con ETLs una herramienta equivalente de manera correcta | Sección 3.2 - Transformación y limpieza | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/etl |
| Modelo | Selecciona el modelo adecuado al problema | Sección 4.1 - Selección del modelo | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| | Configura el modelo de manera correcta | Sección 4.2 - Configuración del modelo | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| | Entrena el modelo de manera correcta | Sección 4.3 - Entrenando el modelo | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| Evalúa | Separa los datos en entrenamiento, validación y pruebas, y los compara correctamente | Sección 4.3 Imagen 4.3.1 - Img 4.3.1 - Separación del dataset en test y train | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| | Selecciona métricas adecuadas | Sección 5.1 - Selección de | https://github.com/AntonioBobadilla/ |

| | | | |
|---|---|---|---|
| | | métricas | RetoIA/tree/main/machine_learning/partials/model |
| | Interpreta los resultados del modelo de manera correcta | Sección 5.2 - Interpretación | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| Refinamiento | Utiliza técnicas de regularización | Sección 6.1 - Técnicas de regularización | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| | Ajusta los hiper parámetros o prueba con otras versiones | Sección 6.2 - Ajuste de hiperparámetros y pruebas de versiones | https://github.com/AntonioBobadilla/RetoIA/tree/main/machine_learning/partials/model |
| Solución | Genera un interfaz (web o móvil) para poder utilizar el modelo. | Sección 7 - Deployment<br><br>También se tiene un video del funcionamiento en la sección 7.4 - Video de funcionamiento | https://github.com/AntonioBobadilla/RetoIA/tree/main/web_interface<br><br>----------<br><br>https://youtu.be/Rx2Ertnlvng |

# Índice

# *American express - Default Prediction*

## Introducción

Las empresas bancarias tienen un peso muy grande en la economía global. Tan sólo en México se considera que al menos 41.1 millones de Mexicanos, que representa a un 49.1% del total de adultos de 18 a 70 años, tienen al menos una cuenta bancaria, según el Instituto Nacional de Estadística y Geografía (Inegi, 11 de mayo de 2022).

Los bancos desempeñan un papel importante, ya que se encargan de permitir el flujo de recursos financieros en todo el país, distribuir efectivo, pagar los cheques que se emiten, ofrecer servicio con tarjetas y transferencias bancarias, otorgar préstamos, entre otros.

Para este proyecto, nos enfocamos en los préstamos bancarios que los bancos otorgan mediante tarjetas de crédito. El Portal de Educación Financiera, define el crédito como un préstamo de dinero que una parte otorga a otra, con el compromiso de que, en el futuro, quien lo recibe devolverá dicho préstamo en forma gradual (mediante el pago de cuotas) o en un solo pago y con un interés adicional que compensa a quien presta, por todo el tiempo que no tuvo ese dinero.

Para poder tener acceso a una tarjeta de crédito con American Express, se deben cumplir ciertos requisitos que garanticen que la persona que está aplicando al crédito pagará los montos acordados conforme a las tasas de interés. Es por ello, que este proyecto se enfoca en predecir si un usuario pagará el crédito de la tarjeta a la que aplicó, haciendo uso de un modelo de Machine Learning y entrenando un algoritmo con base en un historial de usuarios.

## 1. Business Understanding

El enfoque de este proyecto es en los préstamos bancarios que los bancos otorgan mediante tarjetas de crédito. El Portal de Educación Financiera, define el crédito como un préstamo de dinero que una parte otorga a otra, con el compromiso de que, en el futuro, quien lo recibe devolverá dicho préstamo en forma gradual (mediante el pago de cuotas) o en un solo pago y con un interés adicional que compensa a quien presta, por todo el tiempo que no tuvo ese dinero.

Para poder tener acceso a una tarjeta de crédito con American Express, se debe tener cierta seguridad de que la persona que está aplicando al crédito tiene los recursos para pagar los montos acordados conforme a las tasas de interés.

## 2. Data Understanding

Los datos del proyecto se encuentran en la página oficial de Kaggle. Lo que se encontró es que hay tres archivos de tipo CSV, dos con la información dividida por clientes y estos a su vez divididos por periodos (uno es el train y otro es el test) y otro con los datos de si el cliente pagó o no.

**Variables independientes**

Las variables independientes se dividen en las siguientes 5 categorías:

- D_* = Deliquency variables
- S_* = Spend variables
- P_* = Payment variables
- B_* = Balance variables
- R_* = Risk variables

Siendo categóricas las siguientes: ['B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68']

**Variables dependientes**

Probabilidad de un pago futuro para un cliente

`['target']` == 0 (el cliente sí pagó)

`['target']` == 1 (el cliente no pagó)

Como variable dependiente, es decir, la variable que se desea predecir se tiene ‘Target’, un valor binario que ayudará a determinar si el cliente pagó o no pagó el crédito.

Las variables se encuentran protegidas por privacidad, es por esta razón que no se puede identificar qué representan de manera literal. Sin embargo debido al contexto del reto no es necesario entender el significado de las variables que conforman el sistema, para ello se realizarán técnicas de obtención de variables explicativas de la variable independiente, las cuales veremos a continuación.

# 3. Data Preparation

## *3.1. Extracción*

Dado el tamaño de los archivos, fue difícil trabajar en el análisis y la manipulación de los datos en nuestras computadoras portátiles. Aún cuando logramos poder importar el dataset con la herramienta dask, fue irrealizable un análisis. Debido a esta situación y después de comentarle nuestra problemática al Dr. Ahuactzin, se concluyó que se trabajaría con sólo un porcentaje designado por el equipo del archivo "train_data.csv".

Como equipo se decidió que se trabajaría sólo con el 20% de los datos del archivo, correspondiente a un tamaño de aproximadamente 3 GB, debido a que esto nos permitiría a todos trabajar desde nuestra computadora personal.

El archivo pudo ser leído y procesado del dataset en la computadora de un integrante del equipo, que cuenta con un procesador Arm M1 de Apple, Intel Core i5-10300H, 4 procesadores principales y 8 procesadores lógicos. Esto fue un contratiempo ya que se tuvo que organizar de una mejor manera el equipo para procesar la limpieza de los datos como primer aproach y posteriormente investigar alternativas que ayudarán a analizar el dataset completo en cada una de las computadoras de los miembros del equipo, teniendo en cuenta el hardware de cada integrante.

Es importante resaltar que para mantener la integridad de los datos, la extracción de este porcentaje fue realizada conservando los mismos porcentajes de los clientes que pagaron y los que no del archivo original. También se hizo un shuffle de los datos por si el archivo tenía un orden específico que pudiera afectar al modelo de predicción. El archivo final con el que se trabajó para el entrenamiento y el testeo del mismo, cuenta con un tamaño de 3.03 GB; lo que representan: 91783 clientes, 1107261 filas y 191 columnas.

Posteriormente, se procedió a realizar el método de Extracción, Transformación y Load del set, con el fin de integrar sólamente los datos necesarios y las variables que mejor explican el modelo.

## *3.2. Transformación y limpieza*

Sabemos que para realizar un modelo de predicción, es indispensable que los datos con los que se trabaje sean los correctos. Es por eso que se realizaron los siguientes procesos para asegurarnos de que los datos que serán alimentados para nuestro modelo sean de gran utilidad y eviten la creación de sesgos. Al final de cada proceso se exportó el DF para así poder trabajar de una manera más sencilla.

**- Análisis de columnas**

Tomando en cuenta que una mala elección de las columnas podrían llegar a incluso afectar nuestro modelo, lo primero que se hizo fue eliminar las columnas que tuvieran 70% o más valores nulos. Esto resultó en la eliminación de 32 columnas, dejando un total de 159 columnas

**- Análisis de filas**

Ya que cada fila no representaba una instancia por sí misma, se hizo el análisis por clientes, pues un cliente (input para el modelo) puede tener más de una entrada. Después de calcular el total de celdas que tiene cada cliente, se eliminaron los clientes (y todas sus respectivas filas) que tuvieran más del 75% de valores nulos. Esto resultó en la eliminación de ningún cliente, en realidad todos los clientes contaban con por lo menos 83% de valores no nulos.

**- One Hot Encoding**

En la descripción de los datos en Kaggle, se nos indicaba que se contaban con un total de 10 columnas categóricas: ['B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68'] .

Se hizo uso de la función preprocessing.LabelEncoder de sklearn para la categorización de los valores de las columnas por números, de esta manera remplazamos todas las columnas categóricas por nuevas columnas con valores numéricos. Esto resultó en la creación de 52 columnas, resultando en un total de 201 columnas.

**- Imputación**

Se hizo uso de la función impute.IterativeImputer de sklearn para la imputación de valores mediante la estimación de las características de las variables a partir de todas las otras.

**- Principal Component Analysis (PCA)**

Dado que se tiene un gran número de columnas, consideramos que sería mejor hacer una reducción de estas y así facilitar su manipulación.

Implementamos la función decomposition.PCA de sklearn para hacer una reducción de la dimensión lineal del DataFrame usando la Decomposición de Valor Singular de los datos. Esta función se aplicó al DataFrame sin las columnas ['customer_ID', 'target'], pues estas columnas deben ser mantenidas con sus valores originales.

Se decidió que la reducción de las columnas sería a 20 columnas finales, dejando un total de 22 columnas (añadimos las columnas que no fueron aplicadas al PCA).

**- Agregación de los clientes**

Ya que se tiene más de una entrada por cliente, es óptimo la reudcción de estas a una sola, para que sólo se tenga una entrada por cliente. Agrupamos por cliente y aplicamos el promedio de cada columna para obtener una sola entrada. Así el DataFrame se redujo a 91783 filas (1 por cliente).

### *3.3. Load*

Una vez terminada la transformación del datset, se exportó el dataset resultante a un archivo csv. Este tiene un tamaño de 41.6 MB, 91783 filas y 22 columnas.

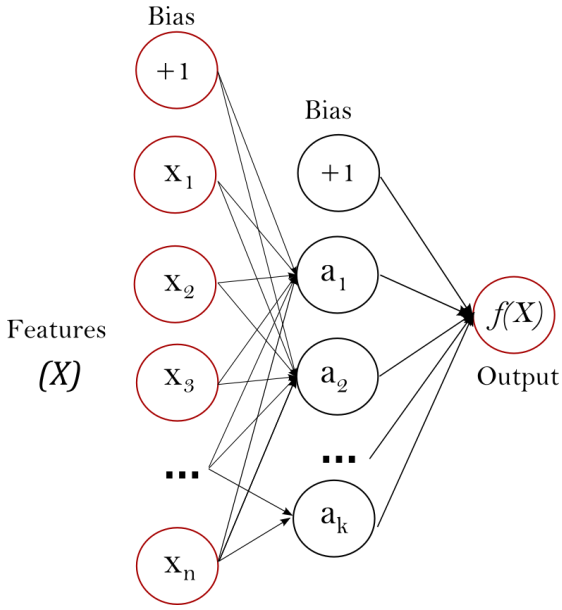
```
# Exporting the final version of the DF (transformations applied)
collapsed_dataframes_customers.to_csv("customers_train_collapsed.csv")
```

Img 3.3.1 - Obtención del dataset final

## 4. Modeling

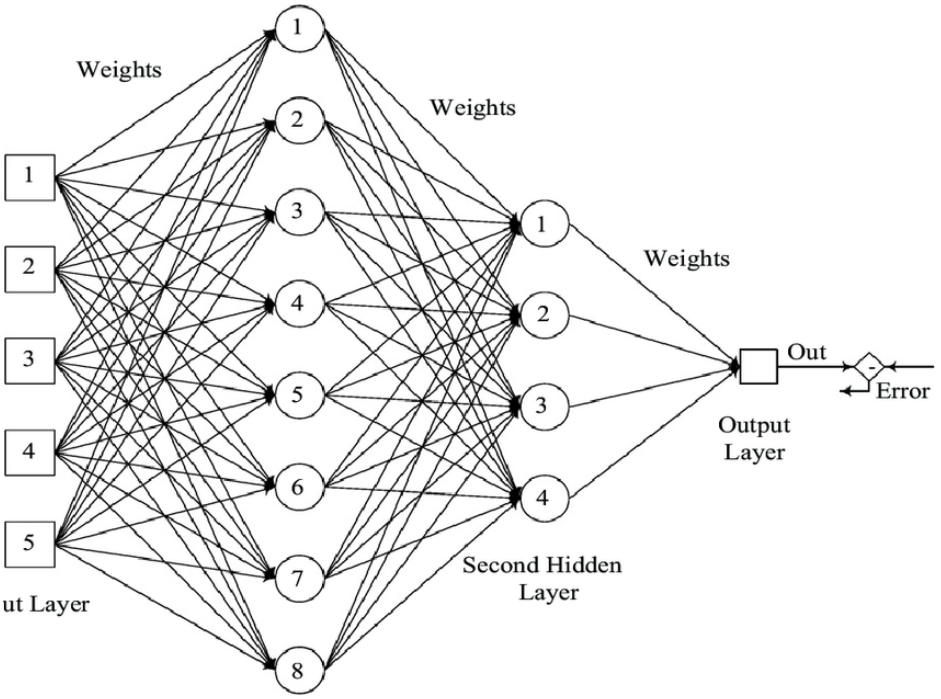
### *4.1. Selección del modelo*

Una vez finalizada la transformación y limpieza de los datos, se hizo uso del dataset para entrenar el modelo. Como primer acercamiento se realizó una investigación sobre qué modelos de Machine Learning son apropiados para darle una solución al reto. Finalmente se tomó la decisión de entrenar al modelo mediante el algoritmo de Multi Layer Perceptron Classifier (MLP-classifier) debido a su alta efectividad al tratar problemas de clasificación, justo como el problema que se tiene.




Img 4.1.1 - Muestra de funcionamiento del modelo MLP-classifier

Así mismo, se decidió realizar un segundo modelo. Con base a la investigación, se llegó a la conclusión de usar el modelo Multi-layer Perceptron regressor (MLP-regressor), el cual se encarga de realizar un algoritmo de backpropagation como parte del aprendizaje.



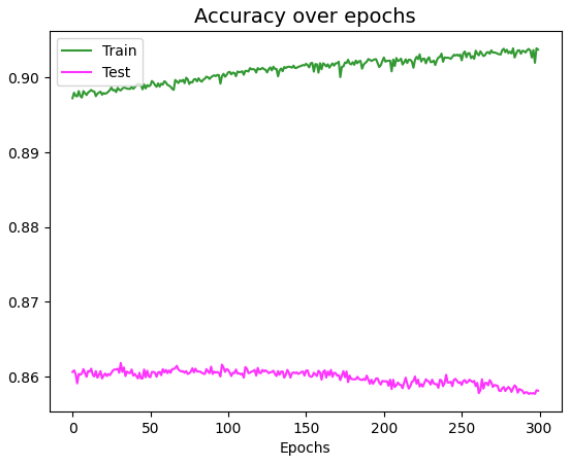

Img 4.1.2 - Muestra de funcionamiento del modelo MLP-regressor

## *4.2. Configuración del modelo*

Al integrar el modelo MLP-classifier se utilizó el train test con el 80% de los datos totales del dataset y se probó con las siguiente configuración inicial de la red neuronal:

- 2 capas profundas de 60 nodos cada una.
- Epochs: 300
- Función de activación: ReLu.
- Algoritmo de optimización de pesos: Adam.

Los resultados se muestran en la siguiente imagen:

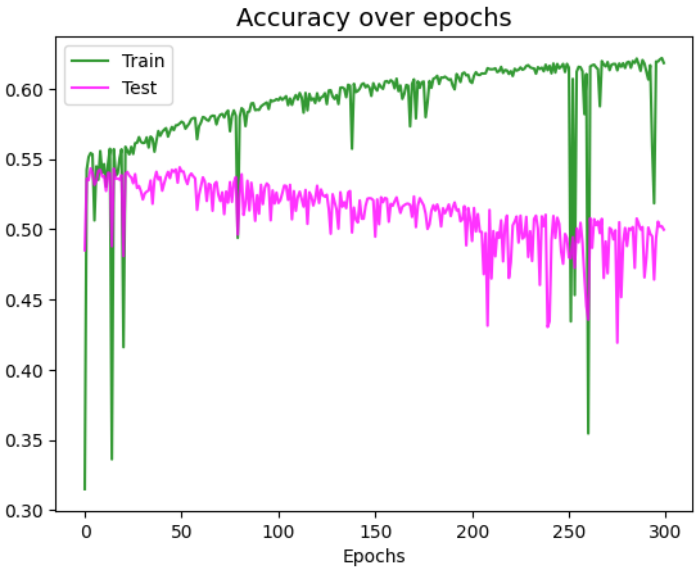



Img 4.2.1 - Gráfica del comportamiento del modelo MLP-classifier con configuración (300,60,60)

Como se puede observar un overfit en este algoritmo, ya que el train aumenta su precisión con base a las épocas y el test disminuye. Como resultado se tiene un train con un accuracy de aproximadamente el 90% pero un test con accuracy del 86%. Esto no significa que es un mal modelo, simplemente que se está overfitteando, se cree que su accuracy es muy alta pero en realidad no llega nunca al 90% con el set de test.

El modelo de MLP-regressor fue realizado haciendo uso de la librería de scikit-learn y se realizó con las mismas configuraciones, para poder realizar una comparación justa de modelos:

- 2 capas profundas de 60 nodos cada una.
- Epochs: 300
- Función de activación: ReLu.
- Algoritmo de optimización de pesos: Adam.




Img 4.2.1 - Gráfica del comportamiento del modelo MLP-regressor con configuración (300,60,60)

Como se puede observar, ambos modelos tienen el mismo punto de inicio, sin embargo, conforme pasan las épocas, el test disminuye y el train aumenta, es por ello que los resultados que se obtienen no son precisos. De igual manera se observan resultados muy exagerados cuya precisión baja al 35%.

Gracias a los resultados obtenidos y al análisis de las gráficas, se tomó la decisión de utilizar el modelo de MLP-classifier para el entrenamiento del modelo, el cual veremos a continuación.

## *4.3. Entrenando el modelo*

Se realizó un split de los datos utilizando el 80% del dataset para utilizarlo posteriormente en el entrenamiento del modelo con las configuraciones mencionadas anteriormente. En cada uno de ellos el tiempo promedio fue de 3 min por entrenamiento. También se utilizó un random state de 21. Lo que realiza este parámetro es controlar el barajeo de los datos aplicado a los datos originales antes de realizar el split.

```
#Splitting the dataset into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state = 21)
```

Img 4.3.1 - Separación del dataset en test y train

Se inicializa el modelo. Los parámetros de epochs, activation, solver y learning rate en esta imagen no son representativos porque fueron cambiando para obtener distintos modelos:

```
#Initializing the MLPClassifier hyperparameters
N_EPOCHS = 200
classifier = MLPClassifier(hidden_layer_sizes=(10,10,10,5), max_iter=N_EPOCHS,activation = 'relu',solver='adam',random_state=1,  learning
```

Img 4.3.2 - Hiper Parámetros Iniciales

Se entrena el modelo con el set de datos train:

```
#Fitting the training data to the network
classifier.fit(X_train, y_train)
```

Img 4.3.3 - Entrenamiento del modelo

# 5. Evaluation

## *5.1. Selección de métricas*

Para evaluar la efectividad del modelo MLP se utilizaron dos métricas de evaluación del modelo, la primer métrica utilizada, es 'score' incluida en la API de scikit learn, la cual compara los resultados de las predicciones, con los valores actuales esperados, a partir de esta comparación se puede identificar qué porcentaje de los datos que se predijeron son correctos y qué porcentaje son incorrectos.

```
#Getting the accuracy of the model
classifier.score(X_test, y_test)
```

```
0.8745982459007463
```

Img 5.1.1 - Precisión del modelo con base en el score

En este caso, el porcentaje de accuracy obtenido haciendo uso del score es de 87%

Como segunda métrica, se hizo uso de una función de 'confusion_matrix' misma que se encuentra de igual manera en scikit learn, en la cual se comparan los valores del target que se predijeron con los valores reales. Como resultado, la función realiza un conteo de negativos verdaderos, falsos negativos, verdaderos positivos y falsos positivos.

```
#Evaluataion of the predictions against the actual observations in y_val
cm = confusion_matrix(y_pred, y_test)
```

Img 5.1.2 - Creación de matriz de confusión

Al finalizar el uso de esta métrica se obtuvo la siguiente matriz de confusión:

```
array([[12467,  1111],
       [ 1191,  3588]], dtype=int64)
```

Img 5.1.3 - Resultado de matriz de confusión

| | | |
|---|---|---|
| 12467 | 1111 | 0.91% |
| 1191 | 3588 | 0.75% |
| 0.91% | 0.76% | |

La matriz dice que nuestro modelo dice que es muy bueno prediciendo positivos y no tan bueno prediciendo negativos.

Una vez obtenida la matriz de confusión, se procede a evaluar la accuracy del modelo con una función que suma la diagonal, es decir, los verdaderos negativos y los verdaderos positivos y posteriormente la divide entre el total de datos que conforman la matriz, obteniendo como resultado la precisión del modelo.

```
#Creating a confusion matrix to help determinate accuracy wtih classification model
def accuracy(confusion_matrix):
   diagonal_sum = confusion_matrix.trace()
   sum_of_all_elements = confusion_matrix.sum()
   return diagonal_sum / sum_of_all_elements
```

Img 5.1.4 - Obtención de porcentaje con base a la matriz de confusión

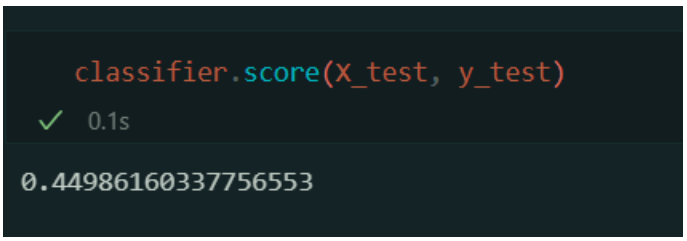
```
#Printing the accuracy
print(f"Accuracy of Model: {accuracy(cm)}")
```

```
Accuracy of Model: 0.8745982459007463
```

Img 5.1.5 - Precisión del modelo con base a la matriz de confusión

En este caso, se obtuvo una precisión del 87% la cual es sumamente buena.

Como evaluación del modelo MLP-classifier se tomó la decisión de realizar de igual manera el accuracy test de score , sin embargo con este modelo, el porcentaje de precisión del modelo fue de tan solo 44% de efectividad.




Img 5.1.6 - Resultado de precisión de MLP-regressor

## *5.2. Interpretación*

Tras realizar ambos modelos y analizar su efectividad, se llegó al acuerdo de utilizar el modelo MLP-classifier para la implementación de la solución del reto, debido a que mostraba mejores resultados en el entrenamiento y testeo del modelo.

# 6. Refinamiento

## *6.1. Técnicas de regularización*

La implementación de Multilayer Perceptron Classifier de SciKit Learn ya incluye por defecto la técnica de regularización L2 con un valor base de 0.001

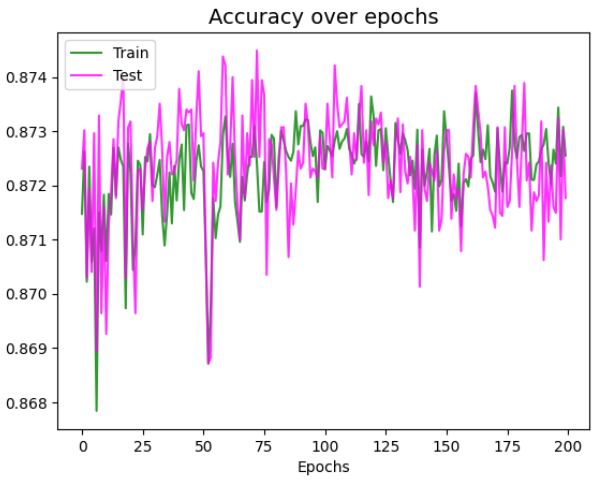
## *6.2. Ajuste de hiper parámetros y pruebas de versiones*

Se realizaron pruebas con otros hiper parámetros para mejorar la precisión del modelo, los resultados se encuentran a continuación:

A. Modelo de 300 épocas y capas 30-30, 0.1 de learning rate, función de activación ReLu y Adam como algoritmo de optimización de pesos.

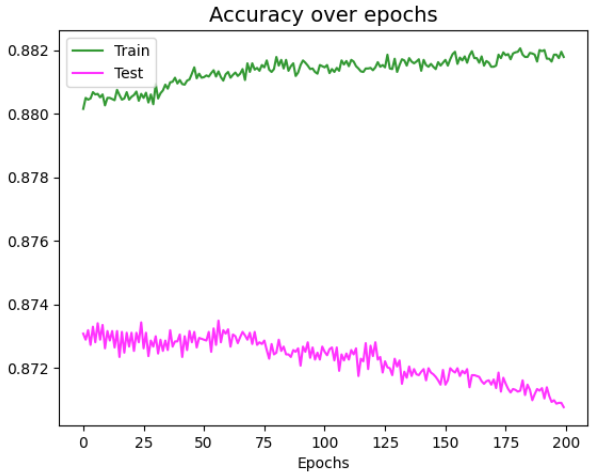

Img 6.2.1- Grafica con configuraciones (300,30,30,0.1)

B. Modelo 200 épocas, capas 33-33, 0.6 de learning rate , función de activación ReLu y Adam como algoritmo de optimización de pesos.



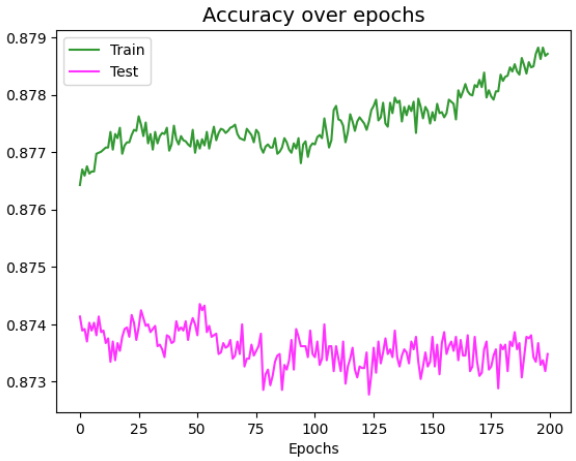

Img 6.2.2- Grafica con configuraciones (200,20,20,0.6)

C. Modelo de 200 épocas, capas 25-25, 0.1 de learning rate, función de activación ReLu y Adam como algoritmo de optimización de pesos.



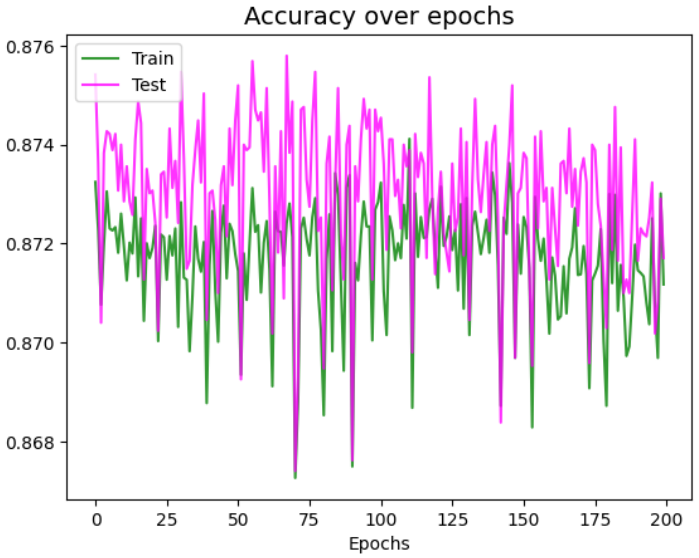

Img 6.2.1- Grafica con configuraciones (200,25,25,0.1)

D. Modelo de 200 épocas, capas 20-20, 0.1 de learning rate, función de activación ReLu y Adam como algoritmo de optimización de pesos.




Img 6.2.1- Grafica con configuraciones (200,20,20,0.1)

E. Modelo de 200 épocas, capas 10-10-10-5, 0.6 de learning rate, función de activación ReLu y Adam como algoritmo de optimización de pesos.




Img 6.2.1- Grafica con configuraciones (200,10,10,10,5,0.6)

Como se puede ver, en su mayoría, los modelos presentan el comportamiento típico de aprendizaje, en donde el test comienza a descender y el train comienza a aumentar, es decir, a realizar un overfit. Así mismo, dos de estas configuraciones (B y E) empiezan a presentar overfitting, ya que se mueven a la par la línea de train y la de predicción.

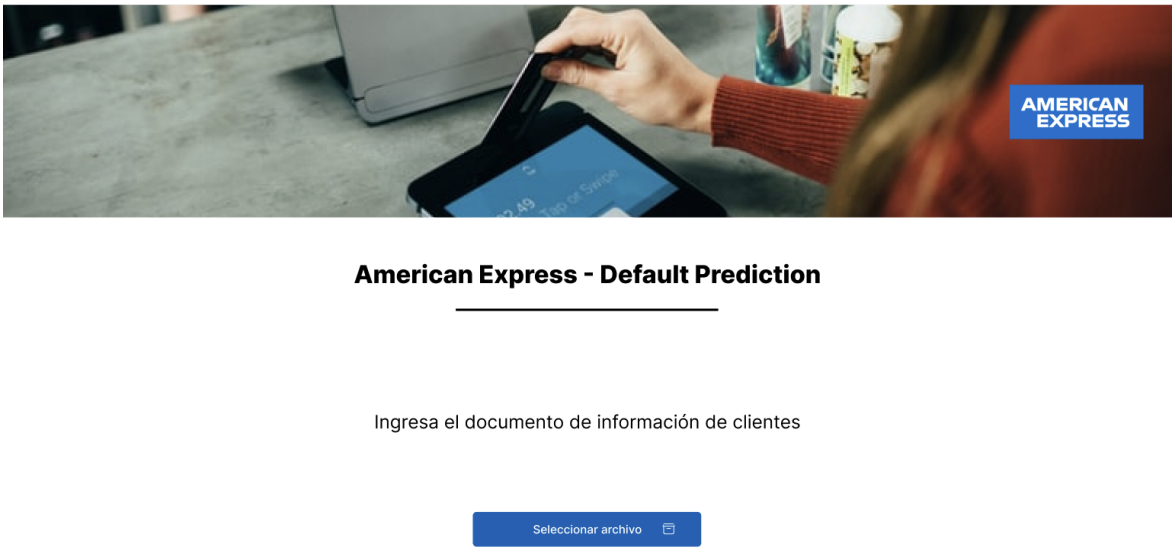
Se tomó la decisión de elegir el modelo “D” debido al comportamiento que tienen las predicciones y los valores reales, se puede observar en la imágen anterior que en la época 25 ambas gráficas alcanzan un punto más alto. Así mismo, tras realizar las pruebas de evaluación con este modelo, se obtuvo un 87.4% de precisión.

# 7. Deployment

Para el despliegue y uso del modelo, se acordó realizar una plataforma web como interfaz gráfica. En ella, las y los miembros encargados de proporcionar los créditos solamente necesitarán adjuntar un archivo .csv con la información requerida de cada uno de los clientes y el sistema se encargará de procesar estos datos en Python haciendo uso del modelo MLP, el cual realizará las predicciones y desplegará los resultados que les ayudarán a determinar si el cliente pagará el crédito.

## *7.1. Mockups de la plataforma*

El diseño del modelo fue realizado en la plataforma web Figma. A continuación se presenta el diseño del producto:




Img 7.1.1- Primera pantalla de la plataforma web mockup


Img 7.1.2- Segunda pantalla de la plataforma web mockup


Img 7.1.3 - Tercera pantalla de la plataforma web mockup

## *7.2. Selección de tecnologías*

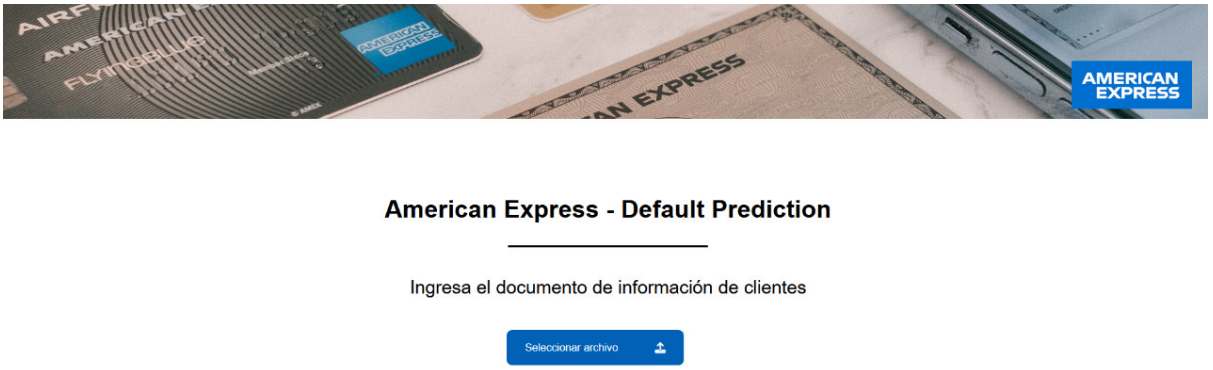
Las tecnologías con las que se decidió trabajar para implementar la plataforma web son HTML, CSS y JS para el frontend, logrando desarrollar una SPA - Single Page Application, la cual se dividió en 3 secciones:

- Página para subir el archivo CSV.
- Página en donde se procesa dicho archivo.
- Página en donde muestra los resultados de las predicciones.

Para el Backend se hizo uso de Express.js el cual realiza una conexión con el script de python. Como primer paso, procesa el archivo .csv y lo convierte en un dataframe, posteriormente se carga el dataframe al modelo MLP ya entrenado. Finalmente, realiza las predicciones y regresa los resultados al frontend, de una manera fácil de interpretar.

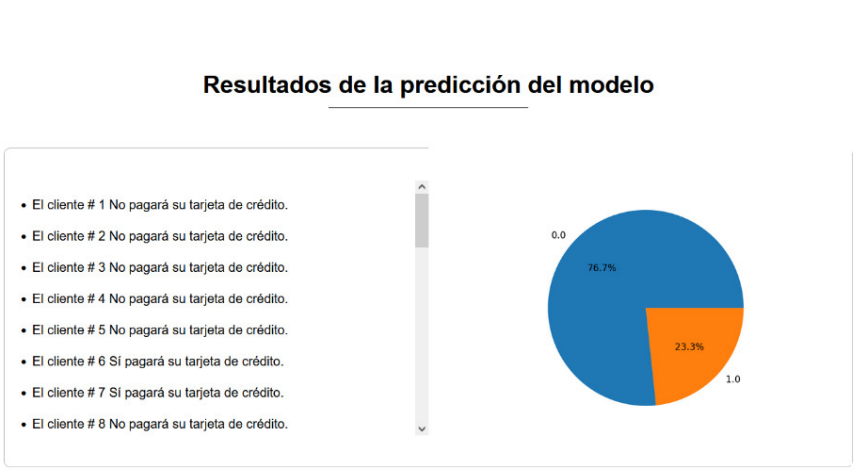
## *7.3. Funcionamiento y despliegue del modelo*

Como resultado final se tiene la siguiente página web:




Img 7.3.1- Primera pantalla de la plataforma web real


Img 7.3.2- Segunda pantalla de la plataforma web real




Img 7.3.3- Tercera pantalla de la plataforma web real

### *7.4. Video de funcionamiento*

A continuación se puede observar un video del funcionamiento de la página web: https://youtu.be/Rx2Ertnlvng

## 8. Resultados

El modelo entrenado cuenta con un alto nivel de efectividad, resultado de un correcto proceso de extracción, limpieza y transformación de datos. Así mismo, es importante mencionar que las variables del modelo tienen alto valor explicativo.

De igual manera sería importante poder asignar diferentes parámetros al modelo de MPL-classifier, ya que al ser un algoritmo diferente, podría mejorar haciendo un cambio de hiper parámetros.

## 9. Normas y principios éticos

El proyecto planteado es una competencia de Kaggle. Para poder participar en esta y obtener toda la información de los datos para su análisis, se necesitó de un registro formal a dicha competencia. Para que el registro fuera exitoso, fue necesario aceptar las reglas de la competencia; en esta se especificaba:

- Elegibilidad: para poder participar el concursante debe ser mayor de 18 años, en caso de proporcionar información falsa se descalificará al concursante.

- Acceso y uso de los datos: el acceso y uso de los datos proporcionados podrán ser sólo utilizados para la participación de la competencia en Kaggle.

- Seguridad de los datos: al aceptar las reglas, los competidores se asegurarán de evitar que personas ajenas a la competencia tengan acceso a los datos proporcionados.

- Condiciones generales: se aplican todas las leyes y reglamentos federales, estatales, provinciales y locales.

Tomando en cuenta que esta competencia es publicada por la empresa American Express a través de Kaggle, es importante resaltar la protección de datos y los principios de la privacidad que tiene esta empresa.

- Colección de los datos: sólo se realizará la colección de datos personales necesaria y por medios legales y justos.

- Avisos y procesamiento: siempre y cuando no se desprenda de los servicios, los clientes serán informados de cómo se procesarán sus datos personales.

- Seguridad y confidencialidad: los datos personales de los usuarios se mantendrá confidencial y se limitará el acceso a su uso a aquellos que los necesiten para llevar a cabo las actividades comerciales del cliente.

- Intercambio de datos: sólo se compartirán los datos personales de los clientes con terceros cuando esto sea necesario para proporcionar un mejor servicio.

**Referencias:**


- Kaggle. (2022). “American Express - Dafault Prediction”. Kaggle. Recuperado el 14 de septiembre del sitio web: https://www.kaggle.com/competitions/amex-default-prediction/overview
- American Express. (s/f). “American Express Data Protection and Privacy Principles”. Recuperado el 14 de septiembre del sitio web: https://www.americanexpress.com/icc/data-protection-and-privacy-principles.html