Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

Inteligencia Artificial para la ciencia de datos TC3007C

Reto
CRISP-DM

Fernando Jiménez Pereyra	A01734609
Daniel Flores Rodríguez	A01734184
Alejandro López Hernández	A01733984
Daniel Munive Meneses	A01734205

30 de Noviembre de 2022

Índice

1. Business Understanding

1.1. Determine Business Objectives

1.1.1. Background

Naatik AI Solutions, con la colaboración directa de Pablo Ibargüengoytia, quienes son una empresa enfocada en el desarrollo y aplicación de Inteligencia Artificial y Ciencia de Datos. Enfocados a brindar soluciones a diferentes sectores de industria y servicios. Con colaboradores con más de 30 años de experiencia y con presencia en más de 150 empresas. Se enfrenta frente al problema de poder tener un producto tangible que permita demostrar las ideas que la empresa tiene y quiere ofrecer

1.1.2. Business Objectives

El objetivo principal del proyecto desde el punto de vista de Naatik es poder vender sistemas basados en modelos de predicción a empresas de diferente giro pero con un factor en común, que sigan con un modelo de negocio de suscripciones de usuario. Empresas de telefonía, internet, servicios de streaming, entre otras pueden ser potenciales clientes de Naatik.

1.1.3. Business Success Criteria

El principal criterio de éxito es que el sistema sea funcional a lo largo del tiempo y vea un resultado positivo en las potenciales ventas que se puedan generar. Lo importante de este aspecto es saber cómo vender ante las empresas; lo que se traduce a resaltar los aspectos más destacados de los resultados empleando un determinado set de datos.

1.2. Assess Situation

1.2.1. Inventory of Resources

Competencias

- Conocimientos de cursos de BigData y cómputo en la nube.
- Documentación de los lenguajes a usar, conceptos y requerimientos de la necesidad del cliente.

Personas

- 4 desarrolladores:
  - 1 Experto en backend
  - 1 Experto en frontend
  - 2 Fullstack

Instalaciones

- Instalaciones del campus Puebla Tec de Monterrey

Materiales

- Frameworks web: Flask y React
- Frameworks Modelo: Numpy, Pandas, Dask, Scikit-learn

Dinero

- $400 dolares de credito en AWS

1.2.2. Requirements, Assumptions and Constraints

Requerimientos funcionales

Modelos

- El sistema deberá recibir un set de datos determinado y aplicar el procesamiento ETL correspondiente.
- Se debe considerar un modelo de clusterización que segmente la información dada en N cantidad

de clústers o grupos. Dicha cantidad de clústers será dinámica de acuerdo al dataset dado.

- Se deberá contar con un modelo entrenado de clasificación que trabaje con cada uno de los clústers generados por el modelo anterior; dicho modelo debe determinar si un determinado cliente abandonaría la empresa o no. Tentativamente se trataría de la implementación de árboles de decisión.
- La demo debe ser capaz de recibir archivos de tipos csv con diferentes sets de datos que pudieran tener variables diferentes, y procesarlos para generar perfiles de clientes, y determinar su churn. Entre las limitaciones planteadas por el socio formador están:

Interfaz web

- El sistema contará con una interfaz web que despliegue los resultados de la implementación de los modelos entrenados previamente mencionados.
- La interfaz deberá mostrar un resumen de los datos más importantes de los resultados del modelo de clasificación como podrían ser Gender, PaymentMethod, TotalCharges, etc.(aún quedan por definir todas las variables que se mostrarán); haciendo uso de aspectos visuales como gráficas, imágenes, comparaciones estadísticas, etcétera.
- Se debe considerar la opción de poder modificar los parámetros de entrada de los modelos a emplear; es decir, se debe poder modificar aspectos como el rango de clasificación.

Requerimientos no funcionales

- Los colores, uso de imagen y multimedia será proporcionado por Naatik con base al aspecto visual de la empresa
- Los modelos deberán ser implementados en el lenguaje de programación Python.
- La interfaz web será realizada en el framework React en términos de frontend; mientras que para el backend se hará uso de Flask
- El sistema debe funcionar de manera local.
- Funciona completamente de manera local.
- Funciona sobre Windows.
- El sistema debe poder correr en la plataforma de demostración, una computadora con i7 de 11 generación y 16 gigabytes de ram.

Asunciones

- Se asume que los dataset empleados para el sistema podrían variar, por lo que se debe considerar la flexibilidad de la clusterización dada
- Se asume que el acompañamiento por parte de Naatik será constante y cercana con tal de desarrollar un sistema óptimo a base de retroalimentación continua en las diferentes etapas del proyecto

Limitantes

- El desarrollo del proyecto sólo será de 8 semanas aproximadamente.
- Se debe seleccionar sólo la información importante para desplegarse en la plataforma, lo que podría omitir buena parte de la información dada en los resultados

1.2.3. Risks and Contingencies

Dentro del desarrollo del sistema, se pueden presentar una cantidad de situaciones o escenarios que pueden influenciar en el funcionamiento del equipo del desarrollo y afectar la calidad de los entregables del proyecto. Algunos escenarios son los siguientes, mencionando entre paréntesis la probabilidad de que éstos sucedan:

- Uno o más compañeros de equipo presenten un cuadro de COVID-19 y no puedan presentarse presencialmente a trabajar. (Medio)
- Mala comunicación entre miembros del equipo. (Bajo)
- Pobre implementación de los modelos o interfaz debido a las entregas individuales correspondientes a los módulos de clase. (Alto)
- Mala implementación de ETL con el set de datos correspondiente para el desarrollo. (Bajo)
- Mala configuración del modelo de clusterización, generando retrasos significativos en los tiempos de entrega. (Medio)
- Elección inadecuada del modelo de clasificación, generando retrasos significativos en los tiempos de entrega y calidad del modelo (Medio).

- Poco o nulo acompañamiento por parte de la organización socio formadora, en este caso Naatik. (Bajo)

1.2.4. Terminology

- Churn rate: la tasa de abandono por parte de los clientes.
- Cluster: un conjunto de cosas que poseen cualidades muy similares entre sí.
- Amazon Web Service: es una nube que ofrece servicios infraestructura como servicio, en donde podremos realizar un entorno de desarrollo y simulación de despliegue para el producto final.
- Infrastructure as a service: en español infraestructura como servicio, consiste en un modelo de negocio en el cual se ofrece una infraestructura de hardware administrada por el proveedor.

1.2.5. Costs and Benefits

Actualmente no se plantea realizar nada que genere un gasto que no pueda ser cubierto más allá de los $400 dólares de crédito que tenemos disponible en Amazon Web Service.

Se plantea que el beneficio final sea el obtener una demo funcional de los sistemas que puede ofrecer Naatik con el fin de conseguir clientes interesados en desarrollar modelos similares.

1.3. Determine Data Mining Goals

El objetivo de este proyecto es crear un sistema computacional con el fin de clasificar, como ejemplo para entrenar el sistema, a los clientes de una empresa de telefonía para así predecir la probabilidad de abandono de la empresa y poder invertir para poder tomar acciones que reduzcan la cantidad o porcentaje de este tipo de usuario.

1.4. Data Mining Goals

- Implementar un modelo de clusterización con una cantidad de clústers flexible según el dataset a utilizar.
- Implementar un modelo de predicción de abandono de clientes para cada clúster generado. Considerando que se trata de un problema de clasificación, una opción viable de modelo podría ser los árboles de decisión.
- Detectar los perfiles de los clientes a partir de un porcentaje determinado de abandono o churn a partir de un set de datos dado.
- Obtener las matrices de confusión para poder comprender la tendencia de falsos negativos o falsos negativos del modelo de predicción para poder así diseñar un uso interesante de dicha información desde el punto de vista del cliente.

1.5. Produce Project Plan

El plan del proyecto se encuentra dentro de un archivo de Project Libre y se puede encontrar en el repositorio de Github

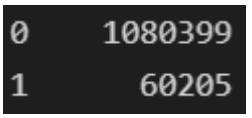
dentro de la carpeta administración bajo el nombre “Reto Bloque 2.pod”. En este archivo se tiene un diagrama con todos los aspectos importantes a considerar cronológicamente para el desarrollo del proyecto.

2. Data Understanding

2.1. Explore Data

El set de datos principal empleado durante el reto refiere a una lista de clientes de una compañía de telecomunicaciones; en la cual se desea saber si un cliente cancelaría su servicio con la empresa (churn) o no, de acuerdo a su perfil.

El set de datos cuenta con 1140604 filas que contienen información de 28 columnas; 5 variables categóricas, y 23 variables numéricas. Respecto a la proporción de clases de predicción, siendo no churn un valor a 0 y churn valor a 1, los clientes se distribuyen de la siguiente manera:



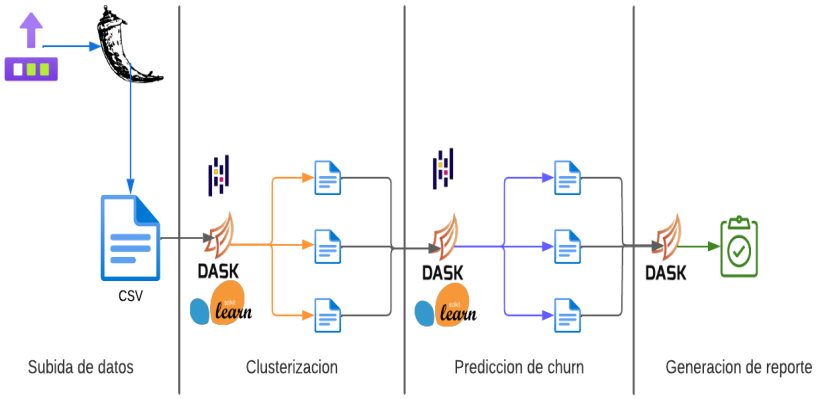
2.2. Verify data

Los datos presentes en el dataset principal no poseen datos anómalos, estando todos los datos presentes, y sin la presencia de outliers.

2.3. Herramientas y tecnologías

Debido a las limitaciones planteadas por el socio formador, se usará el framework de numpy, pandas, y dask para hacer el procesamiento de los datos. Ya que las limitaciones de hardware y software impiden la implementación de algo más avanzado como un cluster de spark.

- Se utiliza principalmente el framework de dask para hacer la mayoría de las tareas del manejo del dataset, ya que nos permite de forma nativa hacer un mejor uso de la memoria, permitiéndonos manejar volúmenes de datos significativos para los recursos de hardware sin que el sistema colapse.
- El uso de pandas tiene como fin realizar las tareas que son más sencillas de implementar en pandas, y que no presentan un problema de manejo de memoria.
- Numpy tiene el fin de ser una herramienta auxiliar para realizar tareas de dask y pandas.



Manejo de los datos

2.4. Big Data en el reto

Consideramos que si se podría dar un enfoque a Big Data, ya que como es necesario que el sistema pueda trabajar con diferentes sets de datos con diversidad de variables, será necesario que se implemente un proceso de ETL que se pueda ajustar a los diferentes sets de datos de forma automática, por lo que se estaría cumpliendo el criterio de veracidad.

Además como el tamaño de los sets de datos que el sistema va a manejar dependerá directamente de las acciones del usuario final, cabiendo la posibilidad de que lleve el sistema al límite para los recursos planteados, se podría decir que cumple el criterio de volumen de forma parcial, ya que no se tiene ninguna certeza de que se cumpla este escenario.

El sistema no requiere complir con el procesamiento de los datos en un plazo de tiempo fijo, sin embargo sería ideal que pudiera procesar un volumen de datos que una persona de marketing puede analizar de forma manual durante un dia trabajo en un plazo de 15 minutos.

3. Data Preparation

3.1. Selecting Data

Con el fin de optimizar el proceso de la siguiente etapa de modelado se realizó un reducción de dimensionalidad, mediante un análisis automático de PCA(Principal Component Analysis). El proceso implementado tiene como fin el conservar las principales variables, que en su conjunto son capaces de explicar 80% o más de la información del dataset.

3.1.1. Implementación de reducción por PCA

A partir del dataset recibido el sistema calcula la proporción de información explicada por cada variable, posteriormente lo asocia con la variable, y la ordena de mayor a menor proporción. Una vez ordenadas las variables el sistema va seleccionando las variables hasta que la suma de la proporciones de la información explicadas por las variables es igual o superior al 80%.

```
def reduce_csv(csv_file):
  df = dd.read_csv(csv_file)
  result = df[df.columns.intersection(target)]
```

```
  df = df.drop(columns= target + ["Unnamed: 0"],
errors='ignore')

  x = getNumericDataset(df)
  columns = list(x.columns)
  x = scaler(x)

  pca = PCA()
  pca.fit(x)

  to_reduce = {}
  for i in range(len(columns)):
     to_reduce[pca.explained_variance_ratio_[i]] =
columns[i]
  to_reduce = OrderedDict(sorted(to_reduce.items(),
reverse=True))

  reduced = []
  explain_ratio = 0
  for i in to_reduce:
     if explain_ratio < .8:
        reduced.append(to_reduce[i])
        explain_ratio += i
     else:
        break

  result[reduced] = df[reduced]

  result.to_csv(csv_file, index=False, single_file=True)
  return str(list(result.columns))
```

3.2. Cleaning Data

Para mantener la mayor integridad posible del dataset, a la vez que generamos un dataset adecuado para procesar, aquellas columnas con una proporción de valores faltantes igual o superior al 20% fueron eliminadas, y aquellas con una

proporción inferior al 20% se utilizó el promedio para reemplazar los valores faltantes.

3.3. Formatting Data

Debido a que los datasets que pudieran ser introducidos al sistema no tienen porque seguir un formato único, al momento de introducir un dataset se comprueba la presencia de variables categóricas, y en caso de existir se transforman mediante un proceso de label encoding.

3.3.1. Implementación de conversión de variables categóricas

El sistema obtiene las variables categóricas presente en el dataset, y posteriormente divide el dataset en categóricas y no categóricas, a las categóricas las transforman en variables numéricas mediante label encoding, y finalmente vuelve a juntar el dataset.

```
# Obtiene una lista con las columnas categoricas a
partir de una diferencia de conjuntos
def getCategoricalColumns(df):
  cols = set(df.columns)
  numColumns = set(df._get_numeric_data().columns)

  return list(cols - numColumns)

# Convierte las variables categoricas en numericas en
base a label encoding
def getNumericDataset(df):
  categoricalColumns = getCategoricalColumns(df)
  numDataset = df.drop(columns= categoricalColumns,
errors='ignore')
  categoricalDataset = df[categoricalColumns]

  le = preprocessing.LabelEncoder()
```

```
    for i in categoricalColumns:
        categoricalDataset[i] =
 le.fit_transform(categoricalDataset[i])

    for i in categoricalColumns:
        numDataset[i] = categoricalDataset[i]

    return numDataset
```

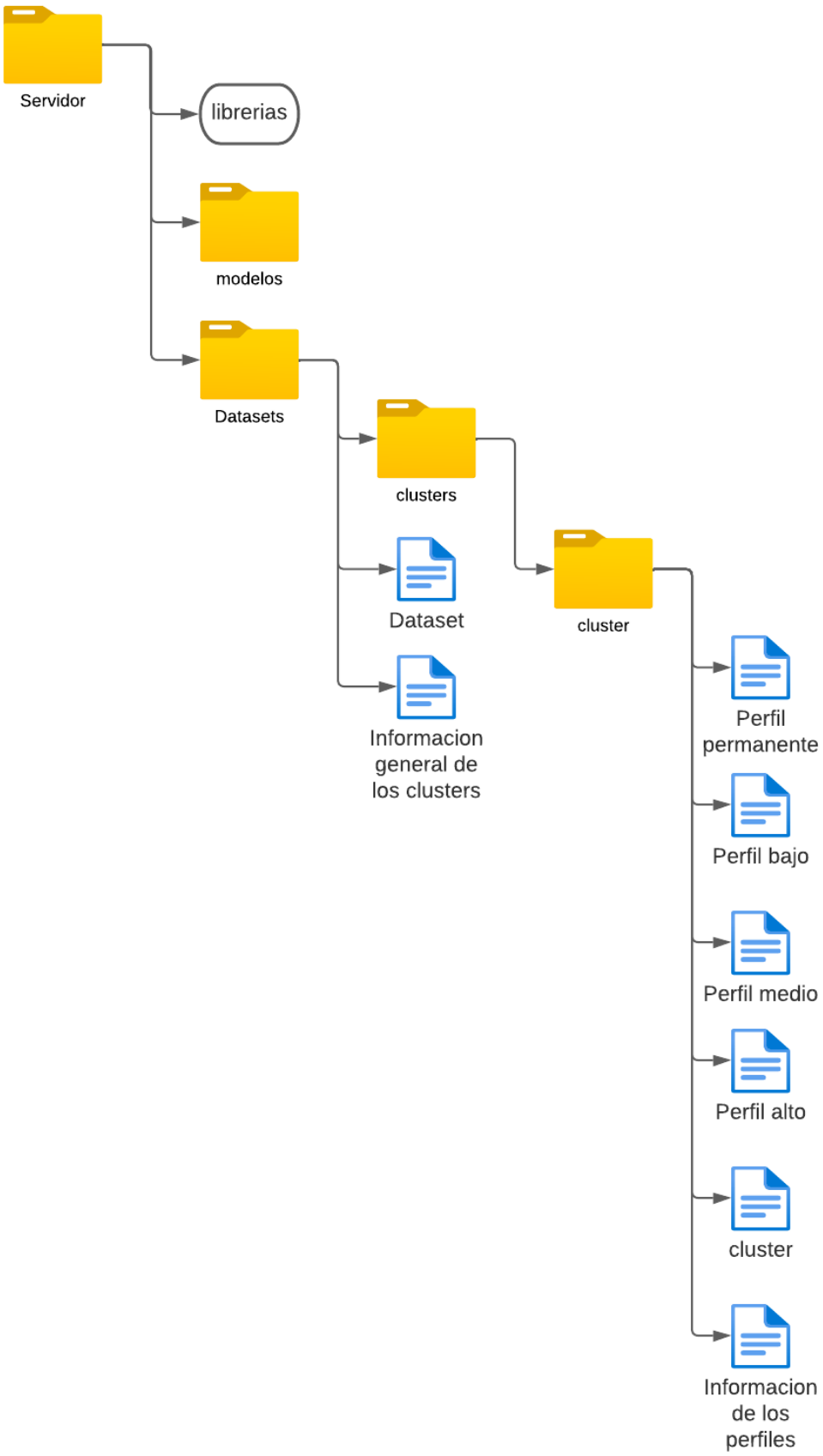
Con el fin de generar los resultados más óptimos, las variables del dataset se estandarizan anterior al proceso de reducción de dimensionalidad, sin embargo la estandarización no se guarda para la siguiente etapa. Antes de seguir con la etapa de modelado se vuelve a estandarizar las variables. El estandarizar las variables tiene el fin de evitar que las dimensiones de las variables afecte de manera negativa a los procesos de reducción de dimensionalidad y de modelado, y el motivo por el cual se vuelve a estandarizar las variables originales después de la reducción de dimensionalidad es evitar que las variables descartadas afecten a la estandarización y por consiguiente influyan al modelado.

3.4. Almacenamiento de los datos

El alcance del reto no plantea que el sistema sea capaz de acceder a bases de datos o sistemas de archivos externos, limitándose a acceder únicamente a archivos locales, sin ser capaz de proporcionar los datos originales o resultantes de

forma externa, por lo que la seguridad de acceso a los datos dependerá de la seguridad que impongan los stakeholders a los usuarios finales. A su vez la limitación sobre el sistema operativo sobre el cual debe trabajar hace que el sistema deba ser capaz de trabajar correctamente sobre el sistema de archivos NTFS.

Con el fin de que el sistema sea lo más eficiente posible en cuanto a tiempo de operación, a medida que el sistema vaya procesando un set de datos irá guardando archivos intermedios. Una vez que el sistema haya procesado los datos, y generado los grupos junto con sus análisis de churn, el sistema generará un reporte sobre el set de datos, el cual se guardará de forma local.



Almacenamiento de los datos

La información general de los clusters, y la información de los perfiles se guarda en formato json, y una posible mejora al manejo de estos datos y de los modelos generados sería el poder almacenarlos en la nube con un servicio como S3 de AWS, sin embargo debido al enfoque local del sistema es una opción que no se implementó.

4. Modeling

4.1. Selecting Modeling Techniques

4.1.1. Clustering

Para la parte de clustering del proyecto, se optó por implementar un K-Means que tiene como objetivo la partición de un conjunto de observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano. Si bien una inconveniencia del modelo es que forzosamente se debe indicar una cantidad k de grupos, se tiene el planteamiento de implementar una manera de poder determinar k de una manera relativamente automática a través del algoritmo ELBOW.

El algoritmo ELBOW determina, entre otras cosas, la distorsión entre clusters; es decir, la cantidad promedio que se tiene entre un cluster con su centroide. De este modo, corre el K Means iterativamente sumando 1

cluster a K por cada iteración. Considerando esto, una manera de implementar K Means de manera semiautomática es aplicar el algoritmo ELBOW , guardar las distorsiones y aplicar una tasa de cambio respecto a las 2 pendientes presentes en 3 distorsiones. Esta tasa de cambio es la razón entre la resta de 2 pendientes respecto a la suma de las mismas; cuando esta tasa sea menor o igual a un 25% arbitrario, significa que el K Means presentará la situación ELBOW; es decir, la distorsión entre K Means con más cantidad de K se hará cada vez más pequeño, por lo que la cantidad óptima de clusters será cuando se llegue al ELBOW.

4.1.2. Churn

Se implementarán 2 modelos Random Forest y una Red Convolutiva de 1 dimensión; además de una regresión logística a manera de modelo Benchmark o base. A continuación se mostrará una tabla comparativa entre dichos modelos

Modelo a emplear	**Características**	**Requerimientos**	**Hiperparámetros**
Random Forest	Consta de un conjunto grande de árboles de decisión ensamblados. Cada árbol es	Las predicciones y errores de cada árbol individual debe tener poca correlación entre	Profundidad máxima de los árboles, cantidad máxima de hojas por árbol, distribución de

	considerado como una clase con diferentes características que arrojan una predicción, y aquella clase con una predicción más asertiva será la que el modelo tome como la predicción final.	sí y respecto a otros árboles	pesos de cada hoja de árboles
Regresión Logística	Muestra la relación entre una característica, para posteriormente calcular la probabilidad de un resultado determinado.	Únicamente es válido para modelo en los que la variable dependiente es dicotómica (binaria).	El intercepto y los pesos que multiplican cada una de las variables independientes.
Red Neuronal Convolucional (CNN)	La CNN es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y “ver”. Se puede implementar en 1 dimensión para el análisis de clasificación poco demandante como texto, o en 2 dimensiones para análisis de	CNN entrena iterativamente ya que en cada paso de tiempo se calculan las derivadas parciales de la función de pérdida con respecto a los parámetros del modelo para actualizar los parámetros. También se puede agregar un término de regularización a la función de pérdida que reduce los parámetros del modelo para evitar el sobreajuste. Esta	Definición de la red en capa de input, capas convolutivas, MaxPooling, Flatten y capas de activación Tamaño de batch Épocas Verbosidad Datos de validación Algoritmo de distribución de pesos Algoritmo de pérdida Métricas a considerar para cada

	patrones o imágenes Cuenta con gran robustez para el análisis de los datos La cantidad de capas, la jerarquía y su estructura queda totalmente a consideración del usuario	implementación funciona con datos representados como matrices numpy densas o matrices scipy dispersas de valores de punto flotante.	época

4.2. Generating a Test Design

4.2.1. Métodos de validación de datos

4.2.1.1. K-Folds Cross Validation

La Validación Cruzada o k-fold Cross Validation consiste en tomar los datos originales y crear a partir de ellos dos conjuntos separados: un primer conjunto de entrenamiento (y prueba), y un segundo conjunto de validación. Luego, el conjunto de entrenamiento se va a dividir en k subconjuntos y, al momento de realizar el entrenamiento, se va a tomar cada k subconjunto como conjunto de prueba del modelo, mientras que el resto de los datos se tomará como conjunto de entrenamiento.

Este proceso se repetirá k veces, y en cada iteración se seleccionará un conjunto de prueba diferente, mientras los datos restantes se

emplearán, como se mencionó, como conjunto de entrenamiento. Una vez finalizadas las iteraciones, se calcula la precisión y el error para cada uno de los modelos producidos, y para obtener la precisión y el error final se calcula el promedio de los k modelos entrenados.

4.2.1.2. Data Balancing

Un conjunto de datos desbalanceado es uno donde el número de observaciones pertenecientes a un grupo o clase es significativamente mayor que las pertenecientes a las otras clases. En este caso, la gran mayoría de muestras cuentan con un target de 0; es decir, de clientes que potencialmente no harían churn. Empleando la función StandardScaler de la librería scikit learn estandariza los datos eliminando la media y escalando los datos de forma que su varianza sea igual a 1, lo cual sería una técnica para poder evitar un set de datos tan desbalanceados y modelos sesgados por el mismo desbalance.

4.2.2. Métricas de comparación entre modelos

4.2.2.1. *Accuracy en test (Validation score)*

La fracción de predicciones que el modelo realizó correctamente. Se representa como un porcentaje

o un valor entre 0 y 1. Es una buena métrica cuando tenemos un conjunto de datos balanceado, esto es, cuando el número de etiquetas de cada clase es similar.

4.2.2.2. Mean Squared Error (MSE)

MSE básicamente mide el error cuadrado promedio de nuestras predicciones. Para cada punto, calcula la diferencia cuadrada entre las predicciones y el objetivo y luego promedia esos valores. Cuanto mayor sea este valor, peor es el modelo.

4.2.2.3. *Mean Absolute Error (MAE)*

El error absoluto medio o MAE es un puntaje lineal, lo que significa que todas las diferencias individuales se ponderan por igual en el promedio. Por ejemplo, la diferencia entre 10 y 0 será el doble de la diferencia entre 5 y 0.

4.3. Building the Models

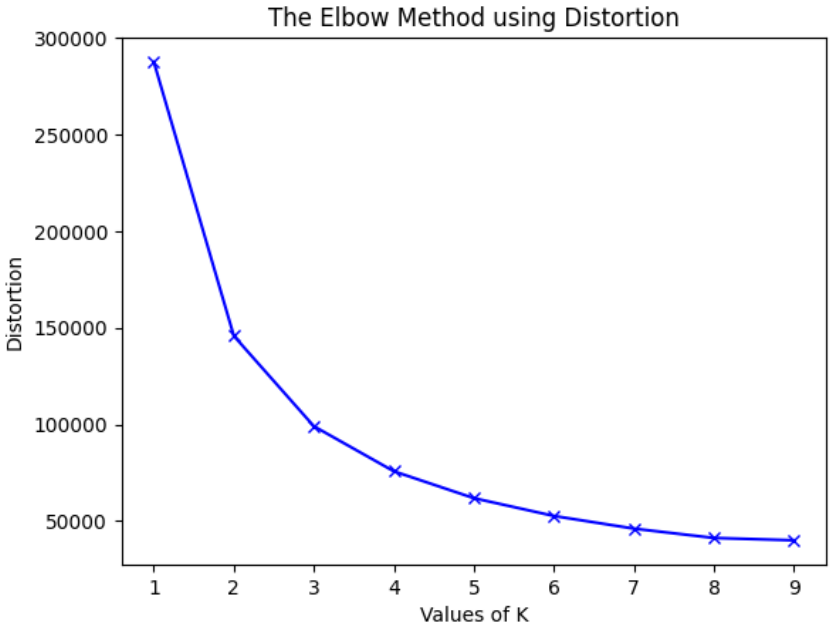
4.3.1. Implementación K Means Clustering con Elbow

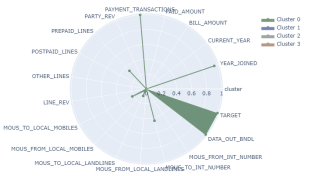
```
#get K value for K Means
def getK(x):
    scaled_x = x
    distortions = {}
```

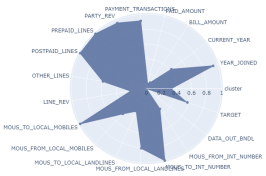
```
    i = 1
    while True:
        #fit k means clustering according to i
        km = KMeans(
            n_clusters= i, init='random',
            n_init=10, max_iter=300,
            tol=1e-04, random_state=0
        ).fit(scaled_x)
        #get distortion of actual k, which is the sum
distance between clusters and their centroid
        current_distortion = sum(np.min(cdist(scaled_x,
km.cluster_centers_,'euclidean'), axis=1)) /
scaled_x.shape[0]
        distortions[i] = current_distortion
        #getting 3 iterations
        if i >= 3:
            #get slope between i -1 and i - 2, i and i - 1
            m1 = distortions[i - 2] - distortions[i - 1]
            m2 = distortions[i - 1] - distortions[i]
            #get the differential between slopes and
addition
            m_dif = m1 - m2
            m_sum = m1 + m2
            #get the percentage representation of
differential, since 100% equals to the sum of slope
values
            dif_percentage = (m_dif * 100) / m_sum
            print(dif_percentage)
            #if this percentage is less than 25%, it means
that  distortion will have a linear behaviour as more k
iterations
            #so we can say that a correct k value for
optimal clustering is i - 2.
            if dif_percentage < 25.0:
                break
        i += 1
    return i - 2
```

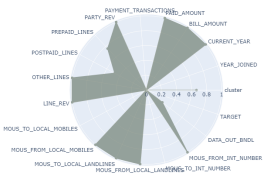
Con el set de datos empleado, se determina que la cantidad adecuada de clusters serían 4. Una vez determinados, se hace el manejo de datos y se

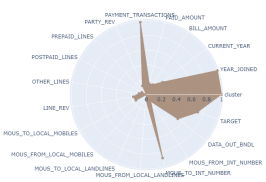
segmenta cada cluster en archivos csv diferentes, además de graficarse sus valores promedio de cada una de las columnas para demostrar gráficamente el comportamiento de cada cluster.











4.3.2. Implementación Random Forest

La implementación del modelo Random Forest quedó de la siguiente manera y con los siguientes hiper parámetros: profundidad máxima por árbol 3, random state de 1, sin definición de hojas máximas por árbol ni distribución de pesos definida El resto de hiper parámetros necesarios para el modelo quedaron en su configuración por default

```
random_forest = RandomForestClassifier(max_depth = 3, random_state = 1, max_leaf_nodes = None, class_weight = None)
```

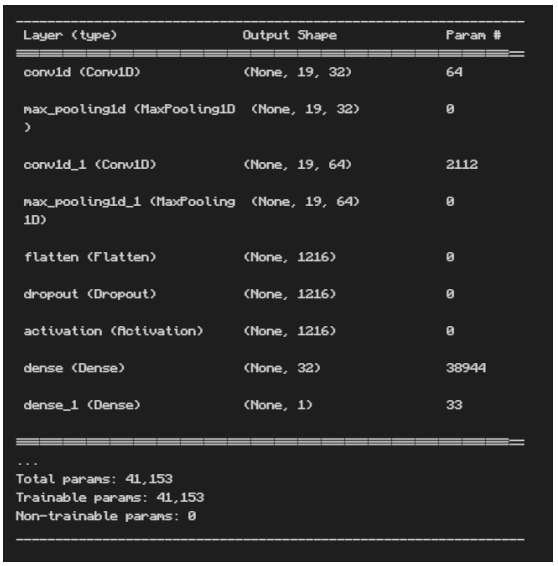
4.3.3. Implementación Regresión Logística

La implementación de la regresión logística se implementó de la siguiente manera, optando por dejar todas las configuraciones predeterminadas que la librería scikit learn otorga con tal de poder tener un modelo más neutro y ver su comportamiento con el set de datos; el único parámetro modificado es la declaración de iteraciones máximas para la conversión del algoritmo de solución empleado por la regresión logística, en este caso se trata del algoritmo lbfgs.

4.3.4. Implementación CNN

La red neuronal empleada como tercer modelo consta con una entrada de tamaño (19,1) correspondiendo a la

cantidad de columnas con el set de datos Se tienen dos capas convolutivas de 1 dimensión, seguidas de MaxPooling con tamaño 1. En adición de ello, se aplica Flatten y se determina un Dropout de 0.5 para evitar tanto overfitting. Asimismo, la función de activación corresponde a ReLU y la última casa emplea sigmoide, debido a las características de clasificación binaria (0 o 1 respecto al churn). La implementación y el resumen del modelo se muestra a continuación.



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 19, 32)	64
max_pooling1d (MaxPooling1D)	(None, 19, 32)	0
conv1d_1 (Conv1D)	(None, 19, 64)	2112
max_pooling1d_1 (MaxPooling1D)	(None, 19, 64)	0
flatten (Flatten)	(None, 1216)	0
dropout (Dropout)	(None, 1216)	0
activation (Activation)	(None, 1216)	0
dense (Dense)	(None, 32)	38944
dense_1 (Dense)	(None, 1)	33

...
Total params: 41,153
Trainable params: 41,153
Non-trainable params: 0

4.4. Assessing the model

Para poder tener una comparativa en el rendimiento de los modelos, se mostrará una tabla comparativa con las métricas seleccionadas. En adición a ello, se aplica esta comparación antes y después de aplicar técnicas de balanceo de datos sobre los datos a emplear.

4.4.1. Comparativa entre modelos antes de Data Balancing

Modelo	Score (%)	MSE train	MSE test	MAE train	MAE test
Random Forest	99.52	0.0048	0.0047	0.0048	0.0047
Regresión Logística	94.72	0.04924	0.04922	0.1003	0.1004
CNN 1D	94.75	0.04915	0.04916	0.10047	0.10045

4.4.2. Comparativa entre modelos tras Data Balancing

Modelo	Score (%)	MSE train	MSE test	MAE train	MAE test
Random Forest	99.52	1.5487x10^-29	1.5455x10^-29	1.4389x10^-15	1.5445x10^-15
Regresión Logística	100	1.7132x10^-30	1.6947x10^-30	3.2427x10^-15	3.2384x10^-16
CNN 1D	100	1.5487x10^-29	1.5455x10^-29	1.4389x10^-15	1.4245x10^-15

5. Evaluation

Ya con el rendimiento de los modelos respecto a un modelo de benchmark establecido, se evaluarán tanto el Random Forest como la red convolutiva con 5 métricas de a

evaluación que engloben en rasgos generales las características y rendimiento de los modelos.

5.1. Métricas a considerar

5.1.1. Accuracy

La fracción de predicciones que el modelo realizó correctamente. Se representa como un porcentaje o un valor entre 0 y 1. Es una buena métrica cuando tenemos un conjunto de datos balanceado, esto es, cuando el número de etiquetas de cada clase es similar.

5.1.2. Loss

Una función de pérdida, o Loss function, es una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones realizadas durante el aprendizaje.

5.1.3. ROC-AUC Curve

Una curva ROC es una representación gráfica de la sensibilidad frente a la especificidad para un modelo de clasificación. Es la representación de la razón o proporción de verdaderos positivos respecto a la razón o proporción de falsos positivos.

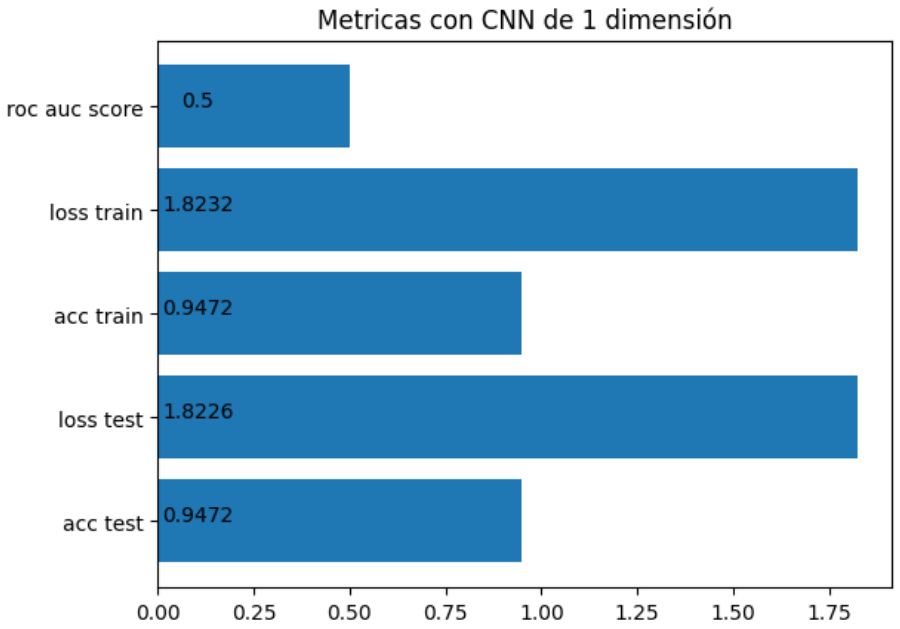
5.2. Evaluating the results

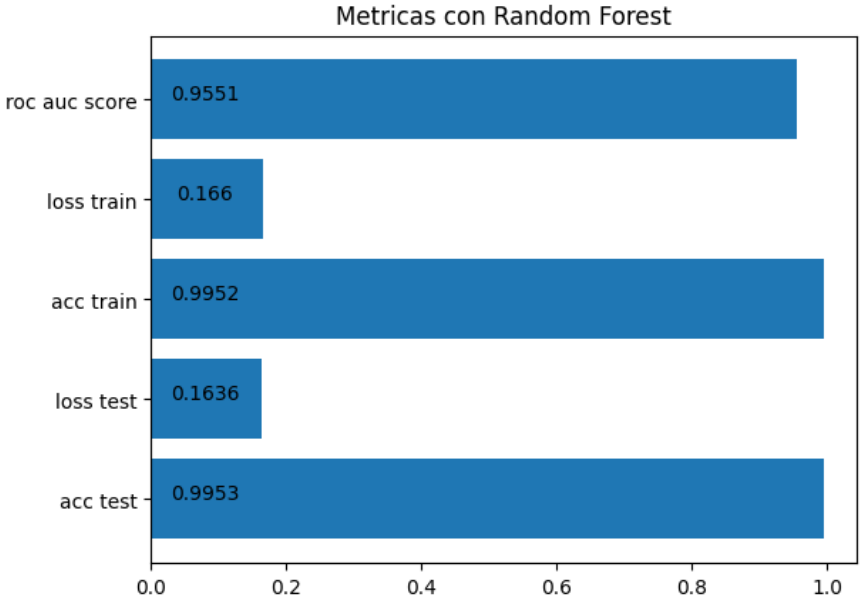
5.2.1. Tabla de comparación de modelos respecto a las métricas seleccionadas

Modelo	Accuracy test	Accuracy train	Loss test	Loss train	ROC-AUC Curve
Random	0.9952	0.9951	0.1636	0.1636	0.9551

Forest					
CNN 1D	0.9472	0.9472	1.8226	1.8232	1

5.2.2. Graficación





5.3. Review process

Si bien el CNN presenta resultados con unos coeficientes casi perfectos, el Random Forest parece tener un comportamiento más realista y eficiente sin la necesidad de aplicar balanceo de datos. En adición a ello, se solicita que el sistema planteado para el reto contemple reajustar el modelo sobre la plataforma en caso de recibir set de datos diferentes; por lo que el rendimiento para el ajuste de los modelos en términos de tiempo resulta fundamental. El CNN, con el estado en el que está, tarda poco más de media hora en realizar el ajuste; mientras que el RandomForest tarda apenas unos minutos. En resumen, por términos de practicidad se opta por emplear el Random Forest sobre una arquitectura de deep learning como lo puede ser una red convolutiva de 1 dimensión.

5.4. Determining the next steps

Evidentemente se trata de un set de datos limitado en cuanto a la cantidad de filas que maneja, sin embargo resulta interesante trabajar con él debido a las características que presenta al tener muchas variables numéricas y pocas categóricas. A raíz de ello, es probable que muchos de los modelos vistos puedan presentar overfitting al momento de ajustarse a este dataset, o presentar poco funcionamiento debido al desbalance de los datos.

Tanto la regresión logística como la red neuronal convolutiva presentaron problemas al trabajar con las características del set de datos en cuestión. Al haber un desbalance importante entre las clases disponibles, ambos modelos prácticamente estaban omitiendo todos los casos en los cuales la clase a predecir resultaba churn. Si bien mostraba una efectividad bastante alta e igual para ambos modelos, no representaba nada si se indaga un poco más en la matriz de confusión y en las probabilidades de cada clase. Ya con el balance de datos, ambos modelos presentaron unos mejores coeficientes y rendimiento; específicamente la CNN.

Por otro lado, el Random Forest resultó una mejor opción que los modelos previamente mencionados ya que, al basarse en árboles de decisión nulamente correlacionados entre sí, forzosamente debe considerar todas las clases presentes en la predicción. En adición a ello, el tiempo de ajuste que se tiene es corto en comparación al CNN, el cual puede terminar de ajustarse en 30 minutos; mientras que el random forest sólo en 4 minutos. Considerando que parte del reto es elaborar un sistema el cual permite entrenar un modelo en una plataforma web cuando se le introduzcan nuevos sets de datos, el tiempo de ajuste es fundamental: por lo que el Random Forest resulta una mejor elección para implementarse en este proyecto.

6. Deployment

6.1. Planning for deployment

Para el despliegue del sistema se optó por la realización de una aplicación web implementada en Javascript bajo el framework web React.js. Dicho sistema considera la implementación de los procesos de ETL, así como de los modelos previamente mostrados para clustering y clasificación dados un set de datos.

En términos de backend, se emplea el framework Flask para el manejo de los datos de la página y del modelado en cuestión. La elección de Flask reside totalmente en que, al ser Python lo que emplea como lenguaje de programación, la adecuación del ETL, la clusterización o el modelo de predicción a endpoints de backend sea más orgánica.

6.2. Planning monitoring and maintenance

El monitoreo y el mantenimiento del sistema recae totalmente en la empresa Naatik, la cual tiene la libertad de hacer las modificaciones que se crean convenientes con tal de preservar o mejorar la plataforma.

6.3. Producing a final report

Como parte del sistema, se genera un reporte pdf con gráficas y aspectos destacados del uso de los modelos con un determinado set de datos. Algunas cosas que se muestran son la matriz de confusión del Random Forest, el comportamiento

de los clusters generados, datos monetarios de cada perfil de probabilidad de churn, etcétera.

En términos de administración de proyecto y documentación, se considera el presente documento como el reporte final del desarrollo del proyecto; en adición a la entrega de un conciso manual de instalación y de usuario para Naatik.

6.4. Conducting a final project review

6.4.1. Que funcionó

- La implementación de React como parte de la plataforma e hizo que el diseño fuera más estético y práctico de implementar, considerando una estructura de componentes web
- Implementar Random Forest como modelo de clasificación considerando su eficiencia y su manejo de set de datos a pesar del desbalance que se pueda presentar
- La consideración de Flask en la teoría para el backend en Python

6.4.2. Qué no funcionó

- La implementación de una plataforma potencialmente cuadrada y poco flexible para la posible llegada de datasets más complejos

- Pésima organización de tiempo por parte de algunos integrantes del equipo considerando las fechas de entrega
- Redefinición de pantallas y funcionalidades aún con tiempo encima
- Mala comunicación por parte de algunos integrantes del equipo