

Presentación: Gestión de Proyectos y Metodologías Ágiles en Data Science

Introducción

¡Bienvenidos a esta sesión dedicada a la gestión de proyectos y metodologías ágiles en el campo de Data Science! En este espacio, exploraremos cómo aplicar principios de gestión de proyectos y metodologías ágiles para llevar a cabo proyectos de análisis de datos de manera efectiva y colaborativa.

¿Qué es la Gestión de Proyectos?

La gestión de proyectos es el proceso de planificación, organización, dirección y control de los recursos para lograr un objetivo específico dentro de un plazo establecido. En el contexto de Data Science, la gestión de proyectos implica coordinar equipos multidisciplinarios para analizar datos y generar insights que impulsen la toma de decisiones informadas.

Componentes de la Gestión de Proyectos:

1. **Planificación:** Comprende la definición de objetivos claros, la identificación de recursos necesarios y la estimación de tiempos y costos. Por ejemplo, al planificar un proyecto de análisis de datos para una empresa de comercio electrónico, se pueden establecer objetivos como mejorar la segmentación de clientes y aumentar las ventas.
2. **Ejecución:** Implica la implementación del plan definido, asignación de tareas y coordinación de actividades del equipo. Durante esta etapa, es importante mantener una comunicación abierta y fluida para garantizar que el trabajo se realice de manera efectiva. Por ejemplo, el equipo de Data Science puede trabajar en estrecha colaboración con el equipo de marketing para recopilar datos y desarrollar modelos predictivos.
3. **Seguimiento y Control:** Consiste en monitorear el progreso del proyecto, identificar desviaciones y tomar medidas correctivas si es necesario. Por ejemplo, si el proyecto de análisis de datos está experimentando retrasos debido a problemas de calidad de datos, es importante identificar y abordar estos problemas de manera oportuna.
4. **Cierre:** Incluye la conclusión del proyecto, evaluación de resultados y lecciones aprendidas para futuros proyectos. Por ejemplo, al finalizar un proyecto de análisis de datos, se pueden realizar sesiones de retroalimentación con el equipo y los stakeholders para identificar áreas de mejora y oportunidades de aprendizaje.

Metodologías Ágiles en Data Science

Las metodologías ágiles son enfoques iterativos y colaborativos para el desarrollo de proyectos que priorizan la entrega de valor al cliente y la adaptabilidad al cambio. En el contexto de Data Science, estas metodologías son especialmente útiles debido a la naturaleza exploratoria y cambiante del análisis de datos.

Principios de las Metodologías Ágiles:

1. **Colaboración con el Cliente:** Trabajar en estrecha colaboración con los stakeholders para comprender y satisfacer sus necesidades. Por ejemplo, en un proyecto de análisis de datos para una empresa de telecomunicaciones, el equipo de Data Science puede colaborar con los gerentes de producto para identificar los requisitos y prioridades del negocio.
2. **Entrega Incremental:** Dividir el proyecto en iteraciones o incrementos de trabajo que proporcionen valor tangible de manera temprana. Por ejemplo, en un proyecto de análisis de datos para una empresa de comercio electrónico, cada iteración puede centrarse en un aspecto específico, como la segmentación de clientes o la recomendación de productos.
3. **Flexibilidad y Adaptabilidad:** Estar abierto a cambios en los requisitos y responder de manera rápida y efectiva a nuevas circunstancias. Por ejemplo, si durante el desarrollo de un modelo predictivo se identifican nuevas fuentes de datos relevantes, el equipo de Data Science puede ajustar el enfoque y la estrategia del proyecto en consecuencia.
4. **Enfoque en Personas:** Valorar la comunicación efectiva, la colaboración entre equipos y el empoderamiento de los miembros del equipo. Por ejemplo, al trabajar en un proyecto de análisis de datos, es importante fomentar un ambiente de trabajo colaborativo donde los miembros del equipo se sientan motivados y comprometidos.
5. **Orientación hacia Resultados:** Centrarse en la entrega de resultados concretos y en la satisfacción del cliente. Por ejemplo, en un proyecto de análisis de datos para una empresa de servicios financieros, el éxito del proyecto puede medirse en función de métricas como la precisión del modelo y el impacto en las decisiones comerciales.

Aplicación de Metodologías Ágiles en Proyectos de Data Science

Aplicar metodologías ágiles en proyectos de Data Science implica adaptar los principios ágiles al ciclo de vida del análisis de datos. Esto implica trabajar de manera iterativa y colaborativa para explorar datos, desarrollar modelos y validar resultados de manera continua.

Ciclo de Vida Ágil en Proyectos de Data Science:

1. **Planificación Iterativa:** Definir objetivos claros para cada iteración o sprint, identificando las tareas necesarias para lograr esos objetivos. Por ejemplo, en un proyecto de análisis de datos para una empresa de seguros, cada iteración puede enfocarse en una parte específica del proceso de modelado, como la selección de variables o la evaluación de modelos.
2. **Desarrollo Incremental:** Construir y validar modelos de manera incremental, integrando retroalimentación del cliente y del equipo en cada etapa del proceso. Por ejemplo, en un proyecto de análisis de datos para una empresa de comercio electrónico, el equipo puede desarrollar un modelo de recomendación de productos en varias iteraciones, refinando y mejorando el modelo en cada paso.
3. **Evaluación Continua:** Evaluar y validar resultados de manera continua, midiendo el desempeño de los modelos y ajustando la estrategia según sea necesario. Por ejemplo, en un proyecto de análisis de datos para una empresa de telecomunicaciones, el equipo

puede monitorear métricas como la precisión y el recall del modelo de detección de fraude y realizar ajustes en función de los resultados.

4. **Adaptación y Mejora:** Estar abierto a cambios en los requisitos y responder de manera rápida y efectiva a nuevas circunstancias. Por ejemplo, si durante el desarrollo de un modelo de clasificación de clientes se identifican nuevas variables predictoras relevantes, el equipo puede ajustar el enfoque del proyecto y volver a entrenar el modelo en consecuencia.

Herramientas y Tecnologías para Proyectos de Data Science

En proyectos de Data Science, el uso de herramientas y tecnologías adecuadas puede facilitar el desarrollo, la colaboración y la implementación de soluciones analíticas. Algunas herramientas y tecnologías comunes incluyen:

1. **Python:** Un lenguaje de programación popular en el análisis de datos debido a su amplia variedad de bibliotecas y su sintaxis intuitiva. Por ejemplo, pandas es una biblioteca de Python ampliamente utilizada para la manipulación y análisis de datos tabulares.
2. **Jupyter Notebooks:** Un entorno interactivo para escribir y ejecutar código, visualizar datos y compartir resultados. Por ejemplo, en un proyecto de análisis de datos, los Jupyter Notebooks pueden utilizarse para explorar datos, desarrollar modelos y presentar resultados de manera interactiva.
3. **Git y GitHub:** Herramientas de control de versiones que facilitan la colaboración y el seguimiento de cambios en el código y los archivos de un proyecto. Por ejemplo, en un proyecto de análisis de datos, Git y GitHub pueden utilizarse para gestionar el código fuente y los conjuntos de datos, permitiendo a los miembros del equipo colaborar de manera eficiente.
4. **Bibliotecas de Machine Learning:** Conjuntos de herramientas y algoritmos para desarrollar modelos predictivos y analíticos. Scikit-Learn y TensorFlow son bibliotecas populares en el análisis de datos debido a su facilidad de uso y rendimiento.
5. **Docker:** Para la creación de entornos de desarrollo reproducibles y escalables, garantizando la consistencia y la portabilidad de los proyectos. Docker simplifica la gestión de entornos de desarrollo y facilita la colaboración entre equipos.

La gestión de proyectos y las metodologías ágiles son fundamentales para el éxito de proyectos de Data Science. Al aplicar estos principios y herramientas, los equipos pueden mejorar la eficiencia, la calidad y la satisfacción del cliente en sus proyectos de análisis de datos.

Análisis de Calidad de Datos

El análisis de la calidad de datos es una parte fundamental en el proceso de ciencias de datos. Su objetivo es evaluar la integridad, precisión, consistencia y confiabilidad de los datos utilizados en un proyecto o análisis

1. **Identificación de valores faltantes:** Este paso implica detectar si hay campos o variables en el conjunto de datos que carecen de valores. Los valores faltantes

pueden deberse a diversos motivos, como errores en la entrada de datos, fallas en los sistemas de recolección o simplemente la ausencia de información.

2. **Detección de inconsistencias en los datos:** Consiste en buscar discrepancias o contradicciones entre los datos. Por ejemplo, si en un conjunto de datos sobre ventas se encuentra que el número de productos vendidos en un día es mayor que el inventario inicial, esto podría ser una inconsistencia.
3. **Identificación de posibles errores:** Esto implica buscar valores atípicos, anomalías o datos que no tienen sentido dentro del contexto del conjunto de datos. Los errores pueden manifestarse de diversas formas, como valores extremadamente altos o bajos que no son representativos del resto de los datos, o datos que violan las restricciones lógicas de un dominio específico.
4. **Evaluación de la consistencia y precisión:** Se refiere a verificar si los datos están en concordancia con las expectativas y reglas del dominio del problema. Esto implica comprobar la coherencia de los datos con respecto a las reglas de negocio y los estándares establecidos.
5. **Limpieza de datos:** Una vez identificados los problemas de calidad de los datos, se procede a corregirlos o eliminarlos. Esto puede implicar imputar valores faltantes, corregir errores evidentes o eliminar registros que no cumplen con ciertos criterios de calidad.

llenar valores faltantes

Para llenar valores faltantes en conjuntos de datos tanto cuantitativos como cualitativos, existen varias estrategias comunes que los científicos de datos suelen emplear:

- **Valores cuantitativos:**
 - a. **Imputación por la media o mediana:** Los valores faltantes se reemplazan por la media o mediana de la variable en cuestión. Esta estrategia es simple y efectiva cuando los datos siguen una distribución normal y los valores faltantes son aleatorios.
 - b. **Imputación por regresión:** Se utiliza un modelo de regresión para predecir los valores faltantes en función de otras variables del conjunto de datos. Esta técnica es útil cuando existe una correlación significativa entre la variable con valores faltantes y otras variables del conjunto de datos.
 - c. **Métodos basados en vecinos más cercanos (KNN):** Los valores faltantes se imputan utilizando los valores de las observaciones más similares en función de una métrica de distancia. Esta estrategia es útil cuando las observaciones se pueden agrupar en un espacio métrico.
 - d. **Imputación múltiple:** Se generan múltiples imputaciones para los valores faltantes, teniendo en cuenta la incertidumbre asociada con la imputación.

Esta técnica es útil cuando se desea tener en cuenta la variabilidad en los datos imputados.

- **Valores cualitativos:**

- a. **Moda o valor más frecuente:** Los valores faltantes se reemplazan por el valor más común o moda de la variable categórica. Esta estrategia es simple y efectiva cuando hay una categoría dominante en la variable.
- b. **Imputación por regresión logística:** Se utiliza un modelo de regresión logística para predecir la categoría de los valores faltantes en función de otras variables del conjunto de datos.
- c. **Muestreo de valores:** Los valores faltantes se imputan seleccionando aleatoriamente valores observados de la misma variable categórica.
- d. **Imputación basada en modelos de clasificación:** Se utilizan modelos de clasificación (como árboles de decisión o métodos de clasificación más avanzados) para predecir la categoría de los valores faltantes en función de otras variables predictoras.

Consolidando Insights Accionables

¡Bienvenidos a la sesión final dedicada a consolidar los hallazgos del análisis en insights accionables! En esta etapa crucial, nos sumergiremos en el vasto mar de datos para extraer sabiduría que guiará nuestras decisiones en Ironhack. Durante esta presentación detallada, exploraremos los aspectos fundamentales del análisis de datos, desde su exploración inicial hasta la presentación final de insights. Nuestro objetivo es proporcionarles las herramientas y el conocimiento necesarios para convertir datos crudos en información valiosa y accionable.

¿Qué es un Insight Accionable?

Un insight accionable va más allá de la simple observación; es una revelación que no solo ilumina, sino que también impulsa a la acción. Estos insights son puntos de inflexión que desencadenan decisiones informadas y estratégicas. Por ejemplo, descubrir que los estudiantes que participan en actividades extracurriculares tienen un rendimiento académico más alto puede conducir a la implementación de más actividades fuera del aula.

Ejemplo:

Imaginemos que, después de analizar los datos de rendimiento académico de una escuela, descubrimos que los estudiantes que participan en actividades extracurriculares tienen un promedio de calificaciones significativamente más alto que aquellos que no participan. Este insight sugiere una estrategia clara para mejorar el rendimiento académico implementando más actividades extracurriculares.

Componentes de un Insight Accionable:

1. **Relevancia:** Debe estar directamente relacionado con los objetivos y necesidades de la organización. Identificar insights relevantes garantiza que las acciones derivadas tengan un impacto significativo en los resultados deseados.
2. **Claridad:** Debe ser fácilmente comprensible y comunicable. La claridad en la presentación de insights facilita su comprensión por parte de todas las partes interesadas, lo que aumenta la probabilidad de adopción de medidas.
3. **Acción:** Debe sugerir una acción específica o una decisión a tomar. Los insights deben ser lo suficientemente claros como para inspirar acciones concretas que impulsen el cambio positivo.

Resumen del Análisis Exploratorio de Datos (AED)

El análisis exploratorio de datos (AED) es la brújula que nos guía a través del vasto territorio de nuestros conjuntos de datos. Nos sumerge en la riqueza de la información, revelando patrones, tendencias y anomalías que pueden pasar desapercibidas a simple vista. Desde la distribución de calificaciones hasta la correlación entre variables, el AED nos proporciona una comprensión profunda de nuestros datos y nos ayuda a formular hipótesis valiosas para análisis posteriores.

Ejemplo de AED:

Al realizar un AED en los datos de los estudiantes de Ironhack, podemos utilizar técnicas como histogramas, diagramas de dispersión y matrices de correlación para visualizar la distribución de calificaciones, explorar relaciones entre variables como horas de estudio y rendimiento académico, y detectar posibles valores atípicos que requieran una mayor investigación.

Componentes del AED:

1. **Exploración de Datos Univariable:** Análisis de una variable a la vez para comprender su distribución y estadísticas descriptivas. Este enfoque nos permite entender la naturaleza de cada variable y cómo se distribuyen en el conjunto de datos.
2. **Exploración de Datos Bivariable:** Análisis de la relación entre dos variables para identificar posibles asociaciones o correlaciones. Este análisis nos ayuda a comprender las interacciones entre diferentes variables y su impacto en los resultados.
3. **Identificación de Valores Atípicos:** Detección de puntos que se desvían significativamente de la tendencia general de los datos. La identificación de valores atípicos nos permite comprender mejor la distribución de los datos y evaluar su influencia en nuestros análisis.

Insights de Calidad de Datos

La calidad de los datos es la piedra angular de cualquier análisis significativo. Los datos incompletos, incorrectos o inconsistentes pueden socavar la validez de nuestros resultados y

conducir a conclusiones erróneas. Por lo tanto, es crucial realizar una evaluación rigurosa de la calidad de los datos y tomar medidas para abordar cualquier problema identificado.

Ejemplo de Problemas de Calidad de Datos:

Supongamos que al analizar los datos de asistencia de los estudiantes de Ironhack, descubrimos que hay entradas duplicadas debido a errores en el proceso de registro. Esto podría distorsionar nuestros análisis y llevar a conclusiones incorrectas sobre la asistencia real de los estudiantes. Para abordar este problema, necesitaríamos implementar un proceso de limpieza de datos para identificar y eliminar duplicados de manera sistemática.

Estrategias de Mejora de Calidad de Datos:

1. **Limpieza de Datos:** Identificación y corrección de errores, valores atípicos y datos faltantes. La limpieza de datos es un proceso esencial para garantizar la integridad y la precisión de nuestros análisis.
2. **Validación de Datos:** Verificación de la precisión y consistencia de los datos mediante técnicas como la validación cruzada. La validación de datos nos ayuda a confirmar la calidad de nuestros datos y a identificar posibles problemas.
3. **Actualización Regular:** Mantenimiento constante de la integridad de los datos mediante actualizaciones periódicas y revisión continua. La actualización regular de los datos garantiza que estén actualizados y reflejen con precisión la realidad en curso.

Resultados del Análisis de Cohortes

El análisis de cohortes es una herramienta poderosa para comprender cómo se comportan grupos de individuos a lo largo del tiempo. Al dividir a los usuarios en cohortes basadas en ciertos criterios (por ejemplo, fecha de inscripción), podemos rastrear su comportamiento y evaluar el impacto de diferentes factores en su experiencia.

Ejemplo de Análisis de Cohortes:

Supongamos que al analizar las tasas de retención de los estudiantes de Ironhack en función de la cohorte de ingreso, descubrimos que las cohortes que se inscribieron durante ciertos períodos tienen una tasa de retención más alta que otras. Este insight nos permite identificar tendencias y patrones significativos en la experiencia del estudiante y nos brinda información valiosa para mejorar nuestras estrategias de retención.

Beneficios del Análisis de Cohortes:

1. **Identificación de Tendencias a Largo Plazo:** Permite observar cómo cambian los comportamientos a lo largo del tiempo. El análisis de cohortes nos proporciona una visión a largo plazo de la evolución del comportamiento de los usuarios.
2. **Comparación entre Grupos Homogéneos:** Facilita la comparación de diferentes cohortes con características similares. Esto nos ayuda a comprender mejor las diferencias en el comportamiento de los usuarios y a identificar áreas de mejora.

3. **Evaluación del Impacto de Intervenciones:** Ayuda a evaluar el efecto de cambios o intervenciones en la experiencia del usuario. El análisis de cohortes nos permite medir el impacto de nuestras acciones y ajustar nuestras estrategias en consecuencia.

Creando una Presentación Efectiva

Una presentación efectiva es el vehículo que lleva nuestros insights desde el laboratorio de datos hasta la sala de juntas, donde pueden inspirar acción y cambio. Para lograr esto, es crucial diseñar una narrativa coherente y visualmente atractiva que resalte los puntos clave y motive a la audiencia a tomar medidas.

Componentes de una Presentación Efectiva:

1. **Introducción Impactante:** Captura la atención de la audiencia y establece el contexto para el resto de la presentación. Una introducción impactante establece el tono y despierta el interés de la audiencia desde el principio.
2. **Visualizaciones Claras y Concisas:** Utiliza gráficos y tablas para ilustrar tus puntos de manera efectiva y comprensible. Las visualizaciones claras y concisas ayudan a transmitir información de manera efectiva y atractiva.
3. **Narrativa Coherente:** Estructura tu presentación de manera lógica y secuencial para mantener el interés y la claridad. Una narrativa coherente guía a la audiencia a través de la presentación de manera clara y comprensible.
4. **Llamado a la Acción:** Concluye tu presentación con recomendaciones claras y acciones específicas que se deriven de tus insights. Un llamado a la acción motivador inspira a la audiencia a tomar medidas basadas en los insights presentados.

El Arte de Contar Historias con Datos

Contar historias con datos es el arte de transformar números en narrativas convincentes que resuenen con la audiencia. Al agregar un toque humano a nuestros análisis, podemos capturar la atención y el interés de la audiencia, inspirando empatía y acción.

Estrategias para Contar Historias con Datos:

1. **Identifica tu Audiencia:** Comprende quiénes son tus espectadores y adapta tu narrativa para resonar con ellos. Al conocer a tu audiencia, puedes personalizar tu mensaje para que sea relevante y persuasivo.
2. **Personaliza tus Ejemplos:** Utiliza ejemplos y casos de estudio relevantes y emocionalmente impactantes. Los ejemplos personalizados ayudan a contextualizar los datos y a hacer que la información sea más accesible y significativa.
3. **Construye una Narrativa Emocional:** Combina datos con anécdotas humanas para crear una conexión emocional con tu audiencia. Una narrativa emocional involucra a la audiencia a un nivel personal, lo que aumenta la resonancia y el impacto del mensaje.

4. **Utiliza Visualizaciones Persuasivas:** Incorpora gráficos y visualizaciones que refuercen tu mensaje y lo hagan memorable. Las visualizaciones persuasivas son herramientas poderosas para transmitir información de manera impactante y memorable.

Conclusión

En esta sesión, hemos explorado los fundamentos de consolidar insights accionables, desde la exploración inicial de datos hasta la presentación final de hallazgos. Al integrar estos conceptos en nuestro enfoque de análisis de datos en Ironhack, podemos tomar decisiones más informadas y estratégicas que impulsen el éxito de nuestra organización.

```
# Importando la biblioteca pandas para manipulación y análisis de
datos
import pandas as pd

# Importando Plotly Express para visualizaciones interactivas de alto
nivel y fáciles de usar
import plotly.express as px

# Importando Plotly Graph Objects para un control más detallado sobre
las visualizaciones
import plotly.graph_objects as go

# Importando NumPy para operaciones numéricas y manipulación de
arreglos
import numpy as np

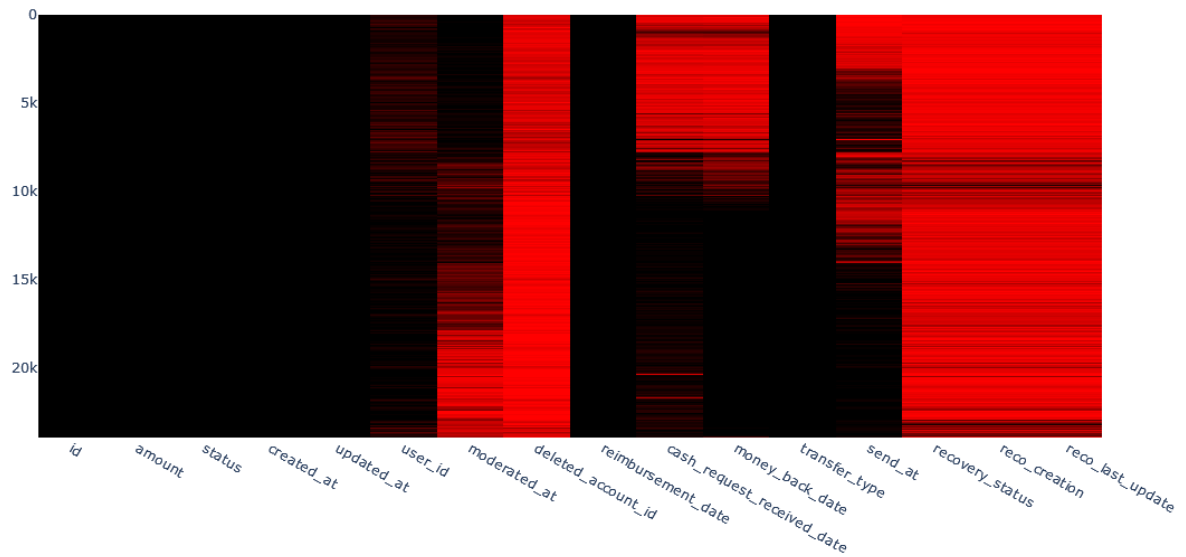
# Importando Seaborn para visualización de datos estadísticos
(opcional, pero puede complementar a Plotly)
import seaborn as sns

# Importando itertools para generar combinaciones de columnas
import itertools

cash_request=pd.read_csv('./project_dataset/extract - cash request -
data analyst.csv')
fees=pd.read_csv('./project_dataset/extract - fees - data analyst
- .csv')

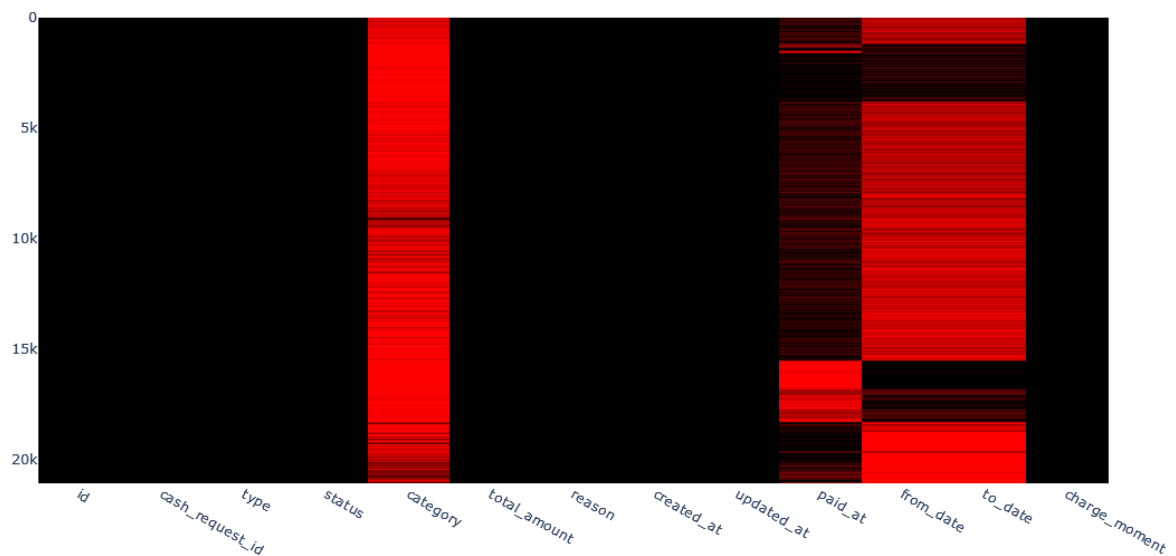
fig1 = px.imshow(cash_request.isnull(),
                  color_continuous_scale=['black', 'red'],
                  labels=dict(color="NaN"),
                  title="Mapa de calor para cash_request")
fig1.update_layout(width=500, height=600)
fig1.update_coloraxes(showscale=False)
fig1.show()
```

Mapa de calor para cash_request



```
fig2 = px.imshow(fees.isnull(),
                  color_continuous_scale=['black', 'red'],
                  labels=dict(color="NaN"),
                  title="Mapa de calor para fees")
fig2.update_layout(width=500, height=600)
fig2.update_coloraxes(showscale=False)
fig2.show()
```

Mapa de calor para fees



```
cash_request.describe()
```

	id	amount	user_id	deleted_account_id
count	23970.000000	23970.000000	21867.000000	2104.000000
mean	13910.966124	82.720818	32581.250789	9658.755228
std	7788.117214	26.528065	27618.565773	7972.743249
min	3.000000	1.000000	34.000000	91.000000
25%	7427.250000	50.000000	10804.000000	3767.000000
50%	14270.500000	100.000000	23773.000000	6121.500000
75%	20607.750000	100.000000	46965.000000	16345.000000
max	27010.000000	200.000000	103719.000000	30445.000000

```
fees.describe()
```

	id	cash_request_id	total_amount
count	21061.000000	21057.000000	21061.000000
mean	10645.355111	16318.449162	5.000237
std	6099.315256	6656.149949	0.034453
min	1.000000	1456.000000	5.000000
25%	5385.000000	11745.000000	5.000000
50%	10652.000000	17160.000000	5.000000
75%	15925.000000	21796.000000	5.000000
max	21193.000000	27010.000000	10.000000

```
cash_request.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 23970 entries, 0 to 23969
```

```
Data columns (total 16 columns):
```

#	Column	Non-Null Count	Dtype
0	id	23970 non-null	int64
1	amount	23970 non-null	float64
2	status	23970 non-null	object
3	created_at	23970 non-null	object
4	updated_at	23970 non-null	object
5	user_id	21867 non-null	float64
6	moderated_at	16035 non-null	object
7	deleted_account_id	2104 non-null	float64
8	reimbursement_date	23970 non-null	object
9	cash_request_received_date	16289 non-null	object
10	money_back_date	16543 non-null	object
11	transfer_type	23970 non-null	object
12	send_at	16641 non-null	object
13	recovery_status	3330 non-null	object
14	reco_creation	3330 non-null	object
15	reco_last_update	3330 non-null	object

```
dtypes: float64(3), int64(1), object(12)
```

```
memory usage: 2.9+ MB
```

```
fees.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21061 entries, 0 to 21060
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    21061 non-null  int64
1   cash_request_id      21057 non-null  float64
2   type                 21061 non-null  object
3   status               21061 non-null  object
4   category             2196 non-null   object
5   total_amount         21061 non-null  float64
6   reason               21061 non-null  object
7   created_at          21061 non-null  object
8   updated_at          21061 non-null  object
9   paid_at             15531 non-null  object
10  from_date            7766 non-null   object
11  to_date              7766 non-null   object
12  charge_moment        21061 non-null  object
dtypes: float64(2), int64(1), object(10)
memory usage: 2.1+ MB

```

Revisión de columnas con valores clave

```

columnas_cash_request=cash_request.columns.tolist()
for i in columnas_cash_request:
    print("=====")
    print(f"{cash_request[i].value_counts().head(10)}")

=====
id
5          1
15322      1
19569      1
15338      1
15329      1
15341      1
15336      1
15334      1
15165      1
15346      1
Name: count, dtype: int64
=====
amount
100.0      16094
50.0       5304
25.0       1276
80.0        267
60.0        190

```

```

70.0      151
20.0      132
30.0      114
40.0      100
90.0       91
Name: count, dtype: int64
=====
status
money_back      16397
rejected        6568
direct_debit_rejected    831
active          59
transaction_declined     48
direct_debit_sent       34
canceled        33
Name: count, dtype: int64
=====
created_at
2019-12-10 19:05:21.596873+00    1
2020-09-09 11:06:31.382449+00    1
2020-10-07 05:56:54.442506+00    1
2020-09-09 12:32:29.661278+00    1
2020-09-09 11:42:31.956338+00    1
2020-09-09 12:43:05.909299+00    1
2020-09-09 12:23:09.95394+00     1
2020-09-09 12:16:17.993242+00    1
2020-09-08 16:01:25.146577+00    1
2020-09-09 12:57:42.599222+00    1
Name: count, dtype: int64
=====
updated_at
2019-12-11 16:47:42.40783+00     1
2020-12-18 13:10:44.348964+00    1
2020-12-18 13:10:44.448561+00    1
2020-12-18 13:10:44.439598+00    1
2020-12-18 13:10:44.429451+00    1
2020-12-18 13:10:44.420825+00    1
2020-12-18 13:10:44.408793+00    1
2020-12-18 13:10:44.396931+00    1
2020-12-18 13:10:44.380111+00    1
2020-12-18 13:10:44.368343+00    1
Name: count, dtype: int64
=====
user_id
3377.0      19
2142.0      18
13851.0     18
2530.0      17
1159.0      17

```

```

10086.0    17
15219.0    17
6634.0     16
9901.0     16
20583.0    16
Name: count, dtype: int64
=====
moderated_at
2019-12-11 16:47:42.405646+00    1
2020-07-10 12:43:27.336286+00    1
2020-07-08 17:55:11.030979+00    1
2020-07-02 12:53:57.219204+00    1
2020-07-17 09:13:58.632576+00    1
2020-07-22 08:58:35.732462+00    1
2020-07-11 12:21:17.278948+00    1
2020-07-20 13:16:58.240135+00    1
2020-08-06 08:51:36.743714+00    1
2020-07-24 08:37:06.064422+00    1
Name: count, dtype: int64
=====
deleted_account_id
6334.0     19
3767.0     16
11444.0    16
4047.0     15
24168.0    13
23326.0    10
4203.0      9
7180.0      9
4267.0      9
12175.0     9
Name: count, dtype: int64
=====
reimbursement_date
2020-11-05 22:00:00+00    1106
2020-10-05 22:00:00+00     761
2020-08-05 22:00:00+00     732
2020-09-07 22:00:00+00     554
2020-11-02 22:00:00+00     432
2020-11-04 22:00:00+00     359
2020-11-03 22:00:00+00     350
2020-08-04 22:00:00+00     350
2020-11-09 22:00:00+00     333
2020-09-04 22:00:00+00     324
Name: count, dtype: int64
=====
cash_request_received_date
2020-10-27     716
2020-10-28     482

```

```

2020-10-30      435
2020-10-20      355
2020-11-03      345
2020-10-24      309
2020-10-13      271
2020-10-21      265
2020-09-08      258
2020-10-22      255
Name: count, dtype: int64
=====
money_back_date
2020-08-04 22:00:00+00      364
2020-08-05 22:00:00+00      295
2020-07-29 22:00:00+00      244
2020-07-07 22:00:00+00      180
2020-09-01 22:00:00+00      134
2020-11-03 23:00:00+00      130
2020-10-06 22:00:00+00      125
2020-09-02 22:00:00+00      118
2020-08-30 22:00:00+00      110
2020-11-11 23:00:00+00      102
Name: count, dtype: int64
=====
transfer_type
instant      13882
regular      10088
Name: count, dtype: int64
=====
send_at
2020-10-23 15:21:26.878525+00      1
2020-10-15 09:12:29.887167+00      1
2020-10-15 09:26:23.065397+00      1
2020-10-08 08:51:51.386371+00      1
2020-10-08 07:52:04.638453+00      1
2020-11-01 10:40:30.955915+00      1
2020-10-08 08:53:46.065496+00      1
2020-10-08 09:21:43.067327+00      1
2020-10-08 07:14:09.553728+00      1
2020-10-08 09:42:20.888932+00      1
Name: count, dtype: int64
=====
recovery_status
completed      2468
pending        845
pending_direct_debit      16
cancelled      1
Name: count, dtype: int64
=====
reco_creation

```

```

2020-06-12 22:27:04.837525+00    1
2020-11-11 23:01:30.183037+00    1
2020-09-22 22:34:09.112102+00    1
2020-09-22 22:23:31.824006+00    1
2020-09-22 22:41:35.956212+00    1
2020-10-02 22:18:26.544227+00    1
2020-09-11 22:43:31.480966+00    1
2020-09-22 22:26:39.034945+00    1
2020-10-04 22:31:44.850342+00    1
2020-10-03 22:28:33.169866+00    1
Name: count, dtype: int64
=====
reco_last_update
2020-07-06 03:36:03.030904+00    1
2020-11-11 23:01:33.548034+00    1
2020-10-03 04:45:39.909619+00    1
2020-09-22 22:23:35.480525+00    1
2020-10-03 07:48:36.457294+00    1
2020-11-06 01:01:41.491634+00    1
2020-10-05 15:03:26.030634+00    1
2020-09-24 15:40:09.086746+00    1
2020-10-15 02:56:51.541306+00    1
2020-10-05 05:46:13.88533+00     1
Name: count, dtype: int64

columnas_fees=fees.columns.tolist()
for i in columnas_fees:
    print("=====")
    print(f"{fees[i].value_counts().head(10)}")

=====
id
6537    1
7852    1
7846    1
7844    1
6968    1
7819    1
7816    1
7815    1
9183    1
7814    1
Name: count, dtype: int64
=====
cash_request_id
12225.0    35
5006.0     28
4410.0     24
12452.0    23
11376.0    21

```



```

2358.0      19
11746.0     17
15319.0     16
4956.0      15
8713.0      15
Name: count, dtype: int64
=====
type
instant_payment      11099
postpone              7766
incident              2196
Name: count, dtype: int64
=====
status
accepted      14841
cancelled      4938
rejected       1194
confirmed        88
Name: count, dtype: int64
=====
category
rejected_direct_debit      1599
month_delay_on_payment      597
Name: count, dtype: int64
=====
total_amount
5.0      21060
10.0       1
Name: count, dtype: int64
=====
reason
rejected direct debit      1599
month delay on payment - 9/2020      283
month delay on payment - 8/2020      203
month delay on payment - 10/2020     102
Postpone Cash Request 12225          34
Postpone Cash Request 5006           25
Postpone Cash Request 4410           23
Postpone Cash Request 12452           20
Postpone Cash Request 11376           20
Postpone Cash Request 2358            18
Name: count, dtype: int64
=====
created_at
2020-06-20 22:12:27.824285+00      13
2020-06-10 07:49:37.83168+00        4
2020-06-19 16:57:53.686817+00        3
2020-06-23 07:46:15.133376+00        3
2020-06-20 08:20:50.052537+00        2

```

```
2020-06-16 20:49:46.374314+00    2
2020-09-22 10:14:11.331781+00    2
2020-06-23 04:07:13.711326+00    2
2020-06-20 04:50:05.304195+00    2
2020-06-23 00:21:05.11354+00     2
```

Name: count, dtype: int64

=====

updated_at

```
2020-10-13 14:25:09.396112+00    1
2020-10-13 14:25:11.713775+00    1
2020-10-13 14:25:11.710272+00    1
2020-10-13 14:25:11.706004+00    1
2020-10-13 14:25:11.703705+00    1
2020-10-13 14:25:11.698162+00    1
2020-10-13 14:25:11.694476+00    1
2020-10-13 14:25:11.692227+00    1
2020-10-13 14:25:11.689347+00    1
2020-10-13 14:25:11.687989+00    1
```

Name: count, dtype: int64

=====

paid_at

```
2020-12-07 16:53:12+00           2
2020-11-05 13:31:25+00           2
2020-12-17 14:50:07.47011+00     1
2020-11-16 19:53:57.573859+00    1
2020-11-06 04:27:32.171688+00    1
2020-11-03 07:04:29.547314+00    1
2020-10-08 07:04:02.101196+00    1
2020-11-07 19:54:47.641021+00    1
2020-10-08 07:04:32.541581+00    1
2020-10-06 19:32:44.115792+00    1
```

Name: count, dtype: int64

=====

from_date

```
2020-10-05 22:00:00+00    297
2020-11-05 22:00:00+00    281
2020-09-04 22:00:00+00    241
2020-08-05 22:00:00+00    241
2020-09-07 22:00:00+00    207
2020-11-02 22:00:00+00    192
2020-11-04 22:00:00+00    137
2020-10-06 22:00:00+00    127
2020-08-04 22:00:00+00    120
2020-10-26 22:00:00+00    104
```

Name: count, dtype: int64

=====

to_date

```
2020-11-06 11:00:00+00    192
2020-11-05 11:00:00+00    187
```

```

2020-10-30 23:00:00+00    150
2020-10-04 22:00:00+00    148
2020-09-04 22:00:00+00    124
2020-10-31 11:00:00+00    111
2020-11-04 23:00:00+00    103
2020-11-04 22:00:00+00     93
2020-12-05 11:00:00+00     89
2020-10-07 22:00:00+00     83

```

```
Name: count, dtype: int64
```

```
=====
```

```
charge_moment
```

```
after      16724
```

```
before      4337
```

```
Name: count, dtype: int64
```

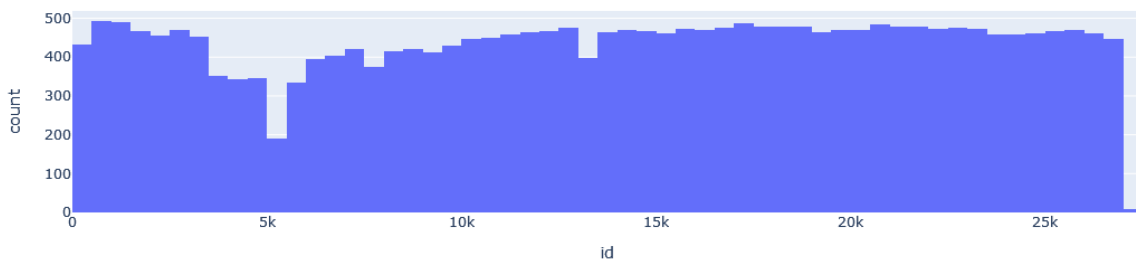
Visualización de distribuciones de datos

```

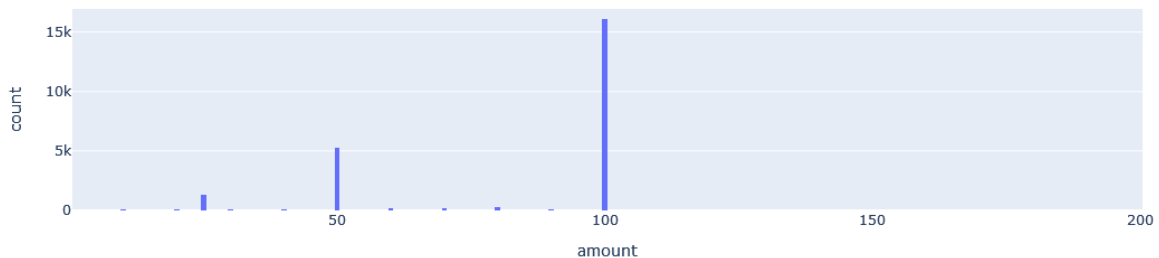
columnas_numericas_cash_request=cash_request.describe().columns.tolist()
for c in columnas_numericas_cash_request:
    fig = px.histogram(cash_request, x=c, title=f'Histograma de {c}')
    fig.show()

```

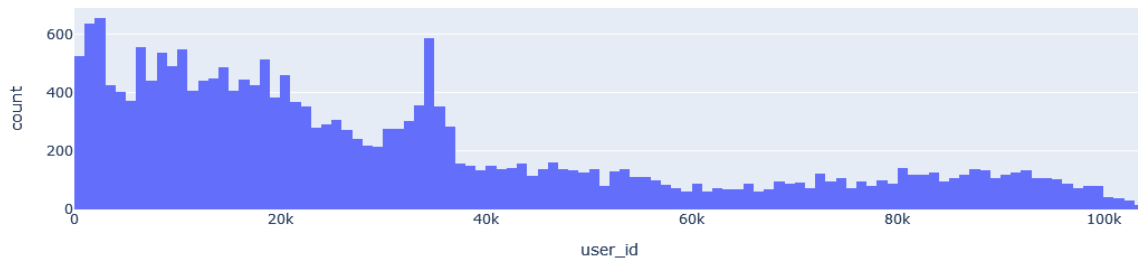
Histograma de id



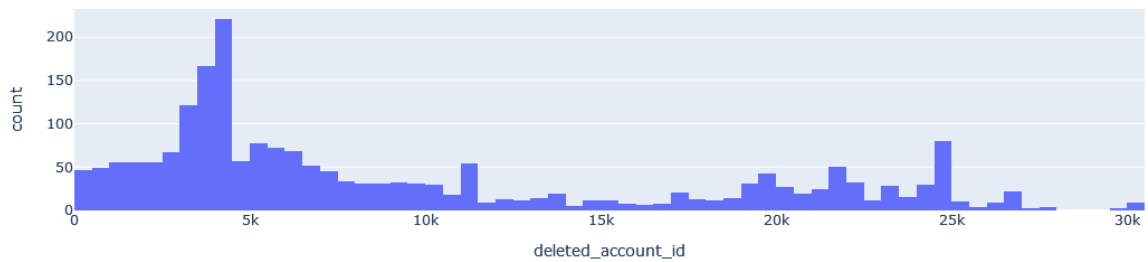
Histograma de amount



Histograma de user_id

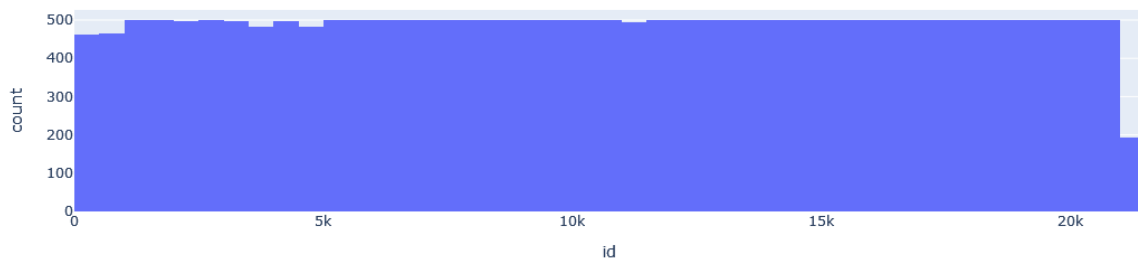


Histograma de deleted_account_id

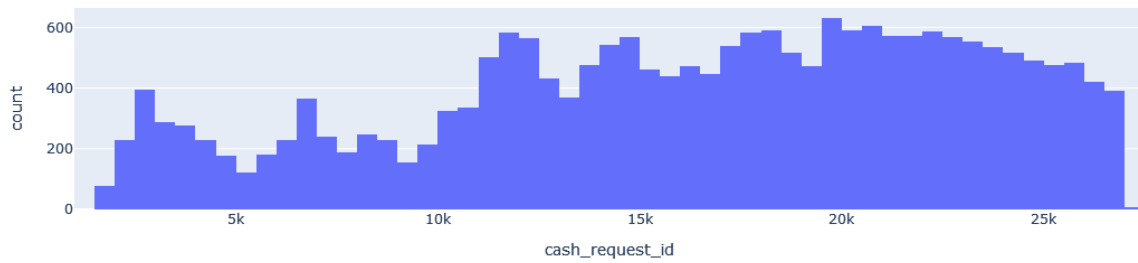


```
columnas_numericas_fees=fees.describe().columns.tolist()
for c in columnas_numericas_fees:
    fig = px.histogram(fees, x=c, title=f'Histograma de {c}')
    fig.show()
```

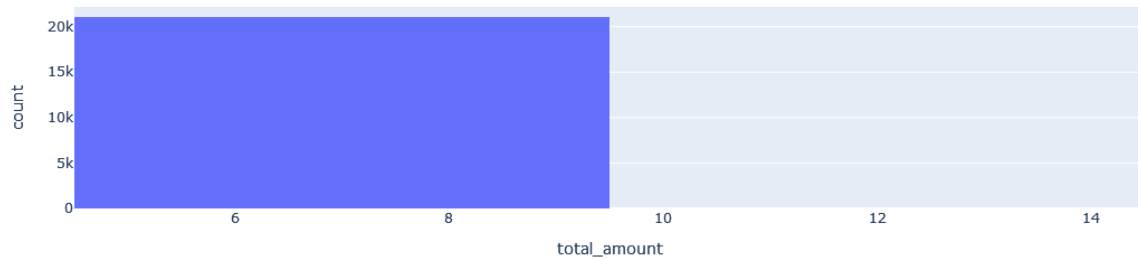
Histograma de id



Histograma de cash_request_id



Histograma de total_amount



```
for c in columnas_cash_request:
    if c not in columnas_numericas_cash_request:
        counts = cash_request[c].value_counts()
        if len(counts)<=10:
            fig = px.pie(names=counts.index, values=counts.values,
title=f'Distribución de {c}')
            fig.show()
```

Distribución de status



Distribución de transfer_type



Distribución de recovery_status

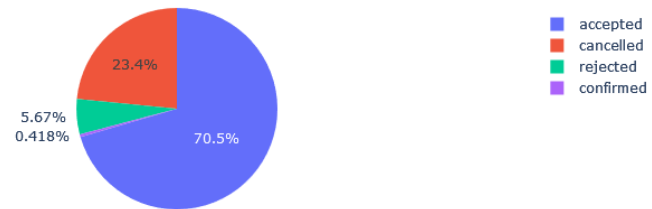


```
for c in columnas_fees:
    if c not in columnas_numericas_fees:
        counts = fees[c].value_counts()
        if len(counts)<=10:
            fig = px.pie(names=counts.index, values=counts.values,
title=f'Distribución de {c}')
            fig.show()
```

Distribución de type



Distribución de status



Distribución de category



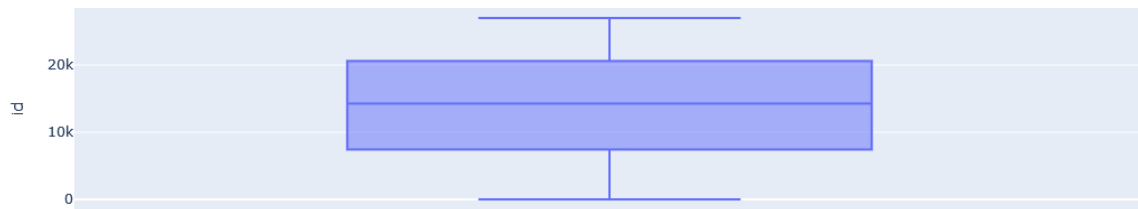
Distribución de charge_moment



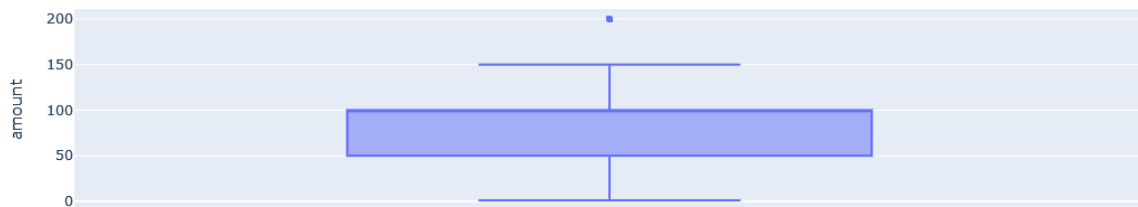
Identificación de valores atípicos

```
for c in columnas_numericas_cash_request:
    fig = px.box(cash_request, y=c, title=f'Boxplot de {c}')
    fig.show()
```

Boxplot de id



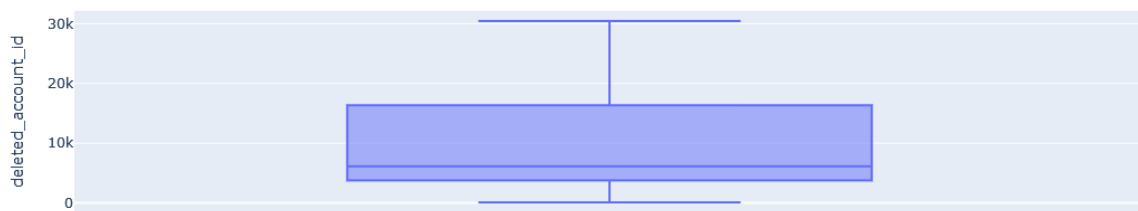
Boxplot de amount



Boxplot de user_id

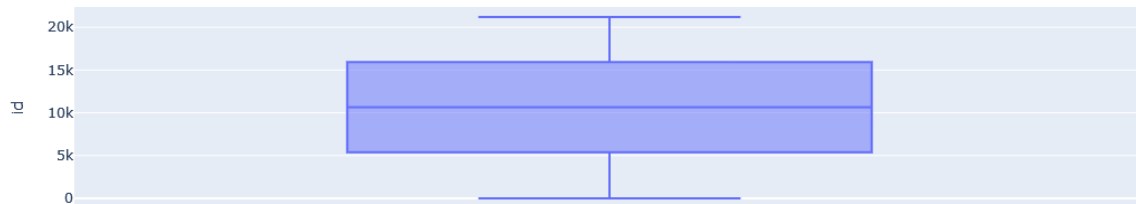


Boxplot de deleted_account_id

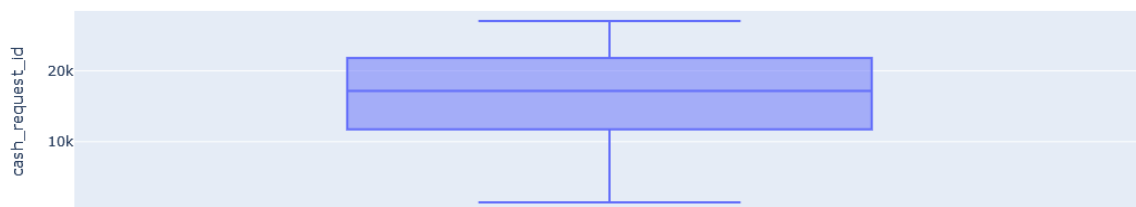



```
for c in columnas_numericas_fees:
    fig = px.box(fees, y=c, title=f'Boxplot de {c}')
    fig.show()
```

Boxplot de id



Boxplot de cash_request_id

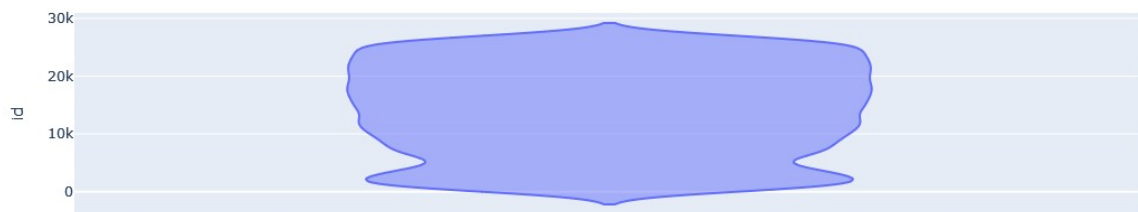


Boxplot de total_amount

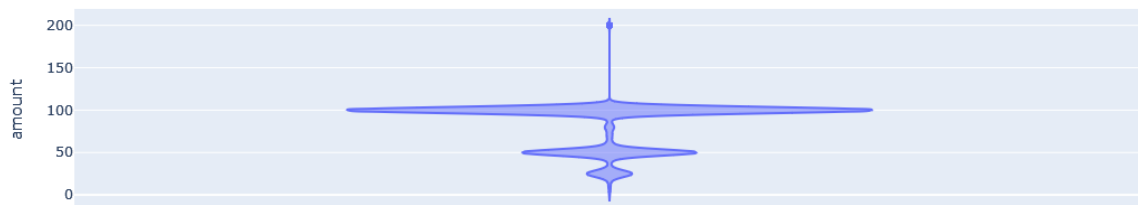


```
for c in columnas_numericas_cash_request:
    fig = px.violin(cash_request, y=c, title=f'Violinplot de {c}')
    fig.show()
```

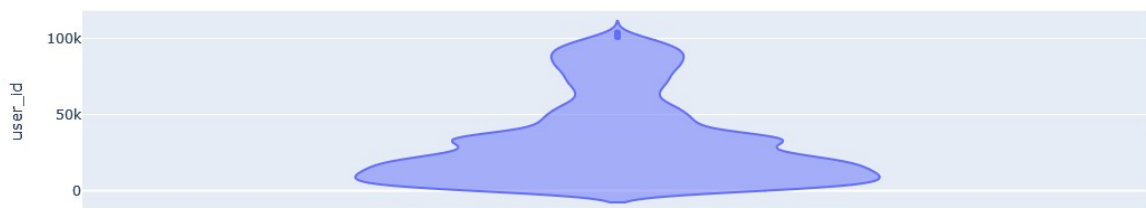
Violinplot de id



Violinplot de amount



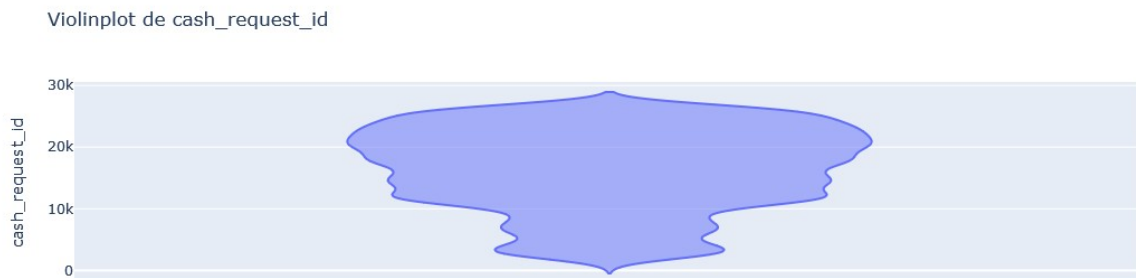
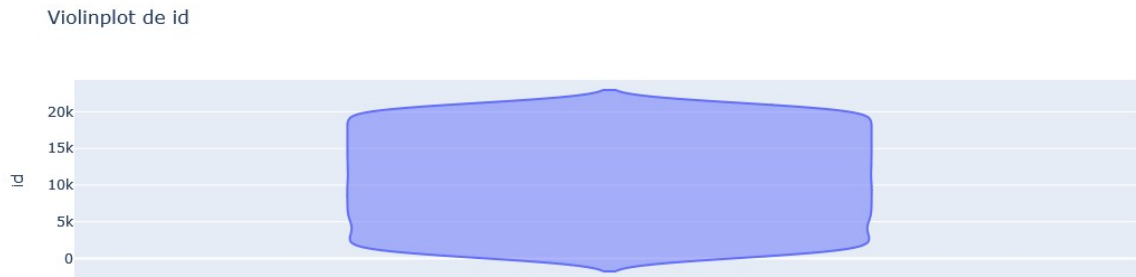
Violinplot de user_id



Violinplot de deleted_account_id



```
for c in columnas_numericas_fees:
    fig = px.violin(fees, y=c, title=f'Violinplot de {c}')
    fig.show()
```



Obtener gráficos de dispersión

```
# Genera gráficos de dispersión para todas las combinaciones de
# columnas numéricas
for c1, c2 in itertools.combinations(columnas_numericas_cash_request,
2):
```

```
fig = px.scatter(cash_request, x=c1, y=c2, title=f'Gráfico de  
dispersión de {c1} vs {c2}')  
fig.show()
```

Gráfico de dispersión de id vs amount

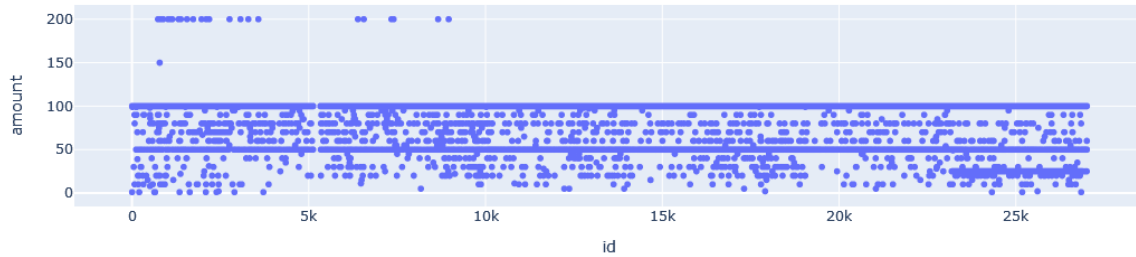


Gráfico de dispersión de id vs user_id

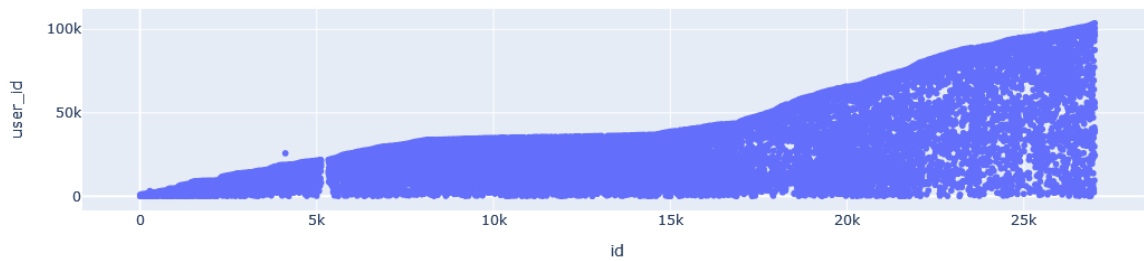


Gráfico de dispersión de id vs deleted_account_id

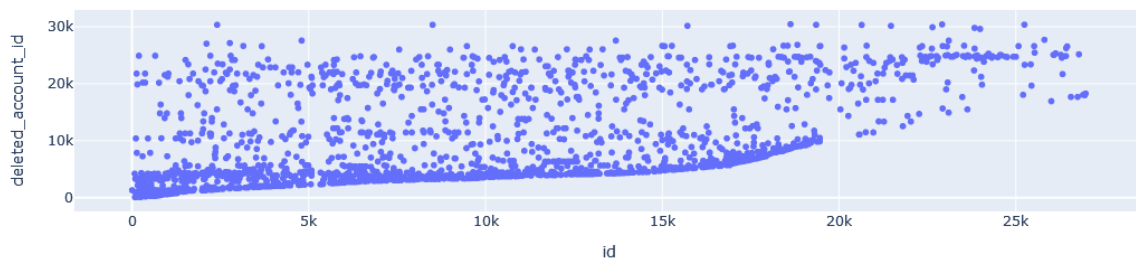


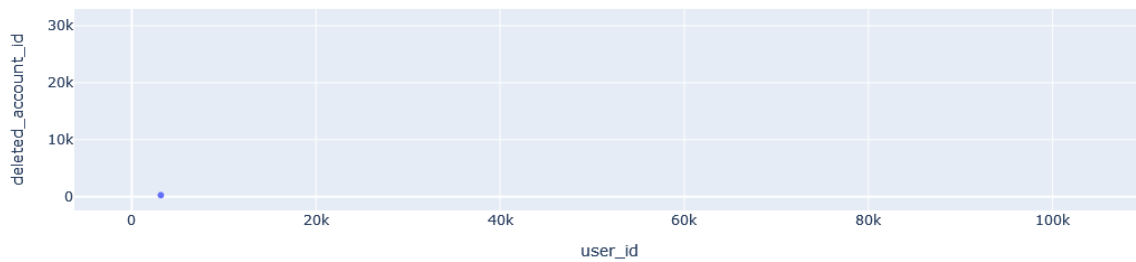
Gráfico de dispersión de amount vs user_id



Gráfico de dispersión de amount vs deleted_account_id



Gráfico de dispersión de user_id vs deleted_account_id



```
# Genera gráficos de dispersión para todas las combinaciones de
columnas numéricas
for c1, c2 in itertools.combinations(columnas_numericas_fees, 2):
    fig = px.scatter(fees, x=c1, y=c2, title=f'Gráfico de dispersión
de {c1} vs {c2}')
    fig.show()
```

Gráfico de dispersión de id vs cash_request_id

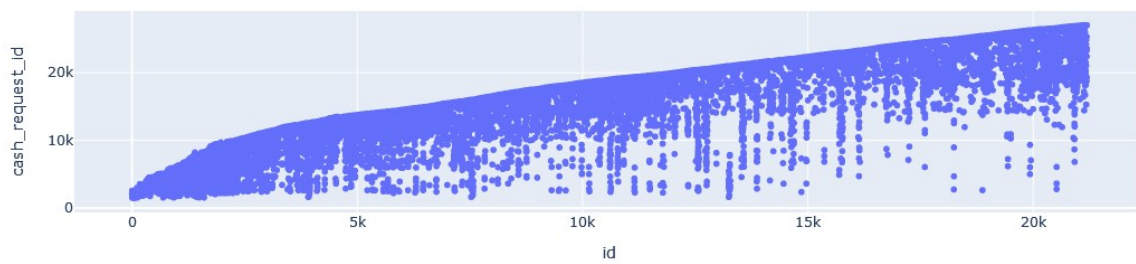


Gráfico de dispersión de id vs total_amount

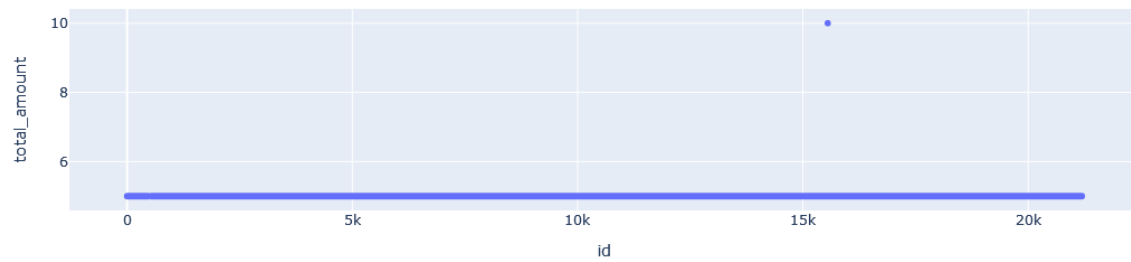


Gráfico de dispersión de cash_request_id vs total_amount

