Informe: Introducción y Análisis Exploratorio de datos (EDA)

Análisis de cohortes

Contenido

Introducción

Este informe documenta el proceso completo de limpieza, preprocesamiento y análisis estadístico realizado sobre dos datasets: cash y fees, con el objetivo de preparar los datos para un análisis de cohortes y evaluar métricas clave como montos solicitados y pagos realizados.

Preparación Inicial

Se importaron las librerías necesarias como pandas, numpy, matplotlib y seaborn. Se desactivaron advertencias para facilitar la lectura del notebook.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

```

Exploración y visión general: limpieza y preprocesado de datos

Dataset 'cash'

- Se cargó el archivo CSV de solicitudes de dinero. Se revisó la estructura con .info()

```
cash=pd.read_csv(r'C:\Users\NURIA\Desktop\A_PYTHON\Introduccion y Analisi Exploratorio de Datos (EDA) Analisis de cohortes\extract_cash request_
cash.info()
```

y .describe(). Se observa que sólo nos interesa 'amount'

```
1  cash.describe()
```
✓ 0.0s

	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

- Se verificó que no existían duplicados.

```
1  print(f' El número de duplicados es: {cash.duplicated().sum()}') #NO HAY DUPLICADOS
```

```
El número de duplicados es: 0
```

- Se identificaron los valore únicos por columnas

```
1  cash.nunique() #VALORES ÚNICOS POR COLUMNAS
```

```
id                          23970
amount                         41
status                          7
created_at                  23970
updated_at                  23970
user_id                     10798
moderated_at                16035
deleted_account_id           1141
reimbursement_date           4089
cash_request_received_date    312
money_back_date             12221
transfer_type                   2
send_at                     16641
recovery_status                 4
reco_creation                3330
reco_last_update             3330
dtype: int64
```

- Se identificaron múltiples columnas con valores nulos, que **no podían eliminarse completamente sin comprometer el análisis**.

```
1  cash.isnull().sum() #HAY MUCHOS NULOS Y NO SE PUEDESN ELIMINAR
```

```
id                              0
amount                          0
status                          0
created_at                      0
updated_at                      0
user_id                      2103
moderated_at                 7935
deleted_account_id          21866
reimbursement_date              0
cash_request_received_date   7681
money_back_date              7427
transfer_type                   0
send_at                      7329
recovery_status             20640
reco_creation               20640
reco_last_update            20640
dtype: int64
```

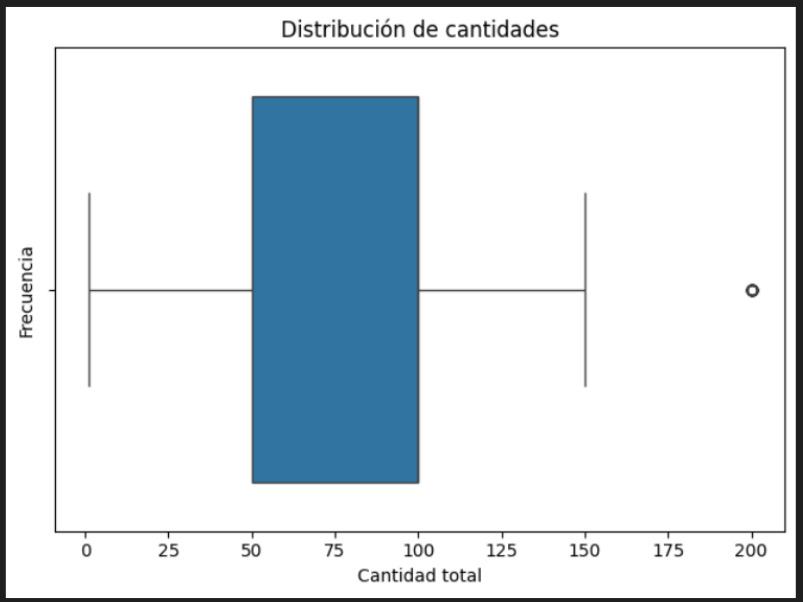
```
1  cash[cash.isna().any(axis=1)].sort_index(ascending=True)
```

Outliers

- Se visualizó la distribución de 'amount' mediante un boxplot.

```
3  plt.title('Distribución de cantidades')
4  plt.xlabel('Cantidad total')      # <- Añade etiqueta al eje X
5  plt.ylabel('Frecuencia')
6  plt.tight_layout()
7  plt.show()
```

✓ 0.2s



- Se aplicó el método IQR (Interquartile Range) para detectar outliers.

```
#OUTLIERS
q1, q3 = np.percentile(cash['amount'], [25, 75])
iqr=q3-q1
lim_inferior = q1 - 1.5 * iqr
lim_superior = q3 + 1.5 * iqr
print(f"\nEn la variable cantidad de dinero pedido")
print(f"Q1:{q1}, Q3:{q3}, IQR: {iqr}")
print(f"Limite inferior: {lim_inferior}")
print(f"Limite superior: {lim_superior}")

outliers_iqr= cash['amount'][(cash['amount'] < lim_inferior) | (cash['amount'] > lim_superior)]
print(f'\nEl número de outliers es:{len(outliers_iqr)}')
print(f"\nOutliers using IQR method: \n{outliers_iqr}")
```

- Se encontraron y eliminaron 25 registros con valores extremos (mayores a 200).

```
En la variable cantidad de dinero pedido
Q1:50.0, Q3:100.0, IQR: 50.0
Limite inferior: -25.0
Limite superior: 175.0


El número de outliers es:25
```

- El dataframe resultante fue almacenado en data para su análisis posterior.

```
#ELIMINAMOS LOS OUTLIERS (25 DE >20000)
cash_so = cash[(cash['amount'] >= lim_inferior) & (cash['amount'] <= lim_superior)]
data=cash_so
data.info()
```

Dataset 'fees'

- Se cargó el archivo CSV de pagos realizados y de forma análoga se analizó con info() y describe() para examinar las estadísticas.

```
#CARGAR FEES
fees = pd.read_csv(r'C:\Users\NURIA\Desktop\A_PYTHON\Introduccion y Analisi Exploratorio de Datos (EDA) Analisis de cohortes\extract_fees_data
fees.info()
```

- También se detectaron múltiples columnas con valores nulos y ningún duplicado.

```
1  fees[fees.isna().any(axis=1)].sort_index(ascending=True) #demasiadas columnas no podemos eliminar
✓ 0.0s
```

```
1  fees.isnull().sum()
✓ 0.0s
```

```
id                     0
cash_request_id        4
type                   0
status                 0
category           18865
total_amount           0
reason                 0
created_at             0
updated_at             0
paid_at             5530
from_date          13295
to_date            13295
charge_moment          0
dtype: int64
```

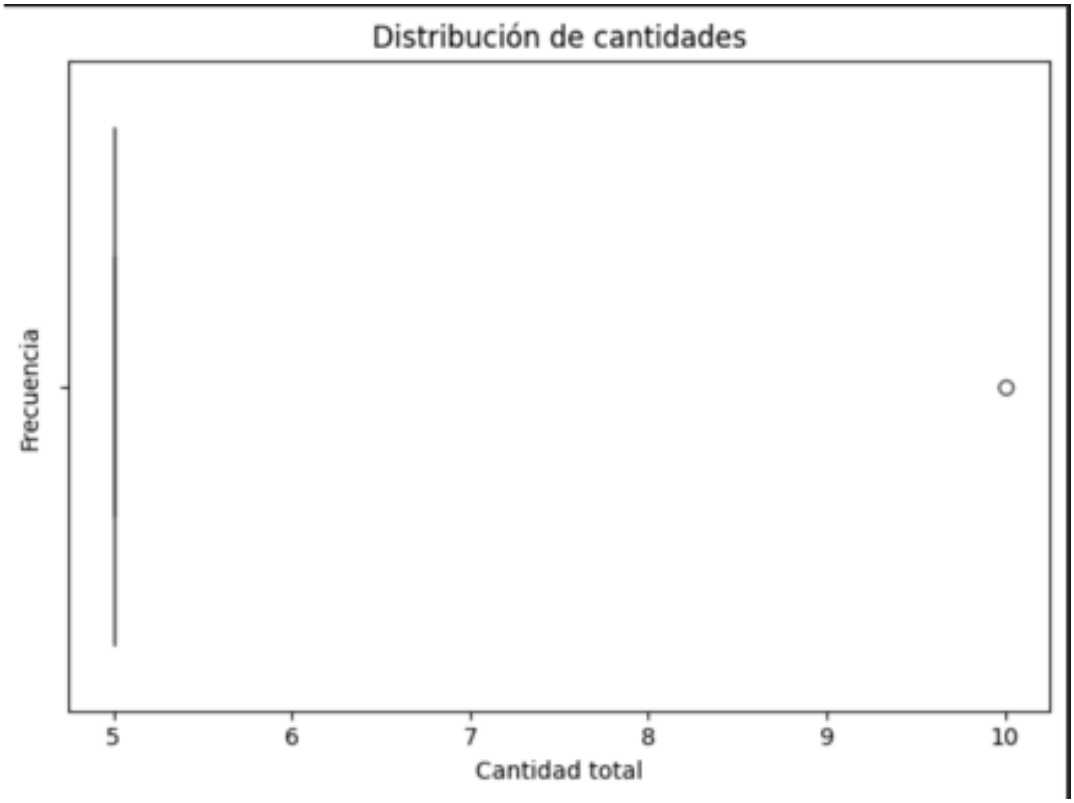
```
1  fees.duplicated().sum() #no hay duplicados
✓ 0.0s
```

```
np.int64(0)
```

Outliers

- Se generó un boxplot de la columna 'total_amount'.

```
1  #VISUALIZACIÓN DE LAS VARIABLES CUANTITATIVAS PARA VER OUTLIERS
2  sns.boxplot(x=fees['total_amount'])
3  plt.title('Distribución de cantidades')
4  plt.xlabel('Cantidad total')      # <- Añade etiqueta al eje X
5  plt.ylabel('Frecuencia')
6  plt.tight_layout()
7  plt.show()
✓ 0.2s
```



- Se aplicó el método IQR, encontrándose 1 outlier extremo (valor igual a 10), que fue eliminado.

```python
#OUTLIERS
q1, q3 = np.percentile(fees['total_amount'], [25, 75])
iqr=q3-q1
lim_inferior = q1 - 1.5 * iqr
lim_superior = q3 + 1.5 * iqr
print(f"\nEn la variable cantidad de dinero (incluyendo IVA)")
print(f"Q1:{q1}, Q3:{q3}, IQR: {iqr}")
print(f"Limite inferior: {lim_inferior}")
print(f"Limite superior: {lim_superior}")

outliers_iqr= fees['total_amount'][(fees['total_amount'] < lim_inferior) | (fees['total_amount'] > lim_superior)]
print(f'\nEl número de outliers es:{len(outliers_iqr)}')
print(f"\nOutliers using IQR method: \n{outliers_iqr}")
```

✓ 0.0s

```
En la variable cantidad de dinero (incluyendo IVA)
Q1:5.0, Q3:5.0, IQR: 0.0
Limite inferior: 5.0
Limite superior: 5.0

El número de outliers es:1

Outliers using IQR method:
20604    10.0
Name: total_amount, dtype: float64
```

- El dataset limpio fue guardado como fees_so.

```
1  fees_so = fees[(fees['total_amount'] >= lim_inferior) & (fees['total_amount'] <= lim_superior)]
2  fees_so.info() #21060 filas
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 21060 entries, 0 to 21060
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               21060 non-null  int64
 1   cash_request_id  21056 non-null  float64
 2   type             21060 non-null  object
 3   status           21060 non-null  object
 4   category         2196 non-null   object
 5   total_amount     21060 non-null  float64
 6   reason           21060 non-null  object
 7   created_at       21060 non-null  object
 8   updated_at       21060 non-null  object
 9   paid_at          15530 non-null  object
 10  from_date        7766 non-null   object
 11  to_date          7766 non-null   object
 12  charge_moment    21060 non-null  object
dtypes: float64(2), int64(1), object(10)
memory usage: 2.2+ MB
```

Análisis de la calidad de Datos

- Ambos datasets fueron procesados eliminando outliers extremos sin afectar la mayoría de los registros.

- Se mantuvieron los valores nulos donde era necesario para no perder información clave.Se recomienda realizar imputación de valores nulos si se requiere un modelo predictivo.

- Se aseguraron datasets sin duplicados y sin ruido numérico en los valores monetarios.

- Se estandarizarán las fechas para análisis de cohortes. Las métricas se realizaran en base en los datasets limpios (data para cash y fees_so para fees).

Métricas a analizar

Frecuencia de Uso del Servicio

Comprender con qué frecuencia los usuarios de cada cohorte utilizan los servicios de adelanto de efectivo de IronHack Payments a lo largo del tiempo.

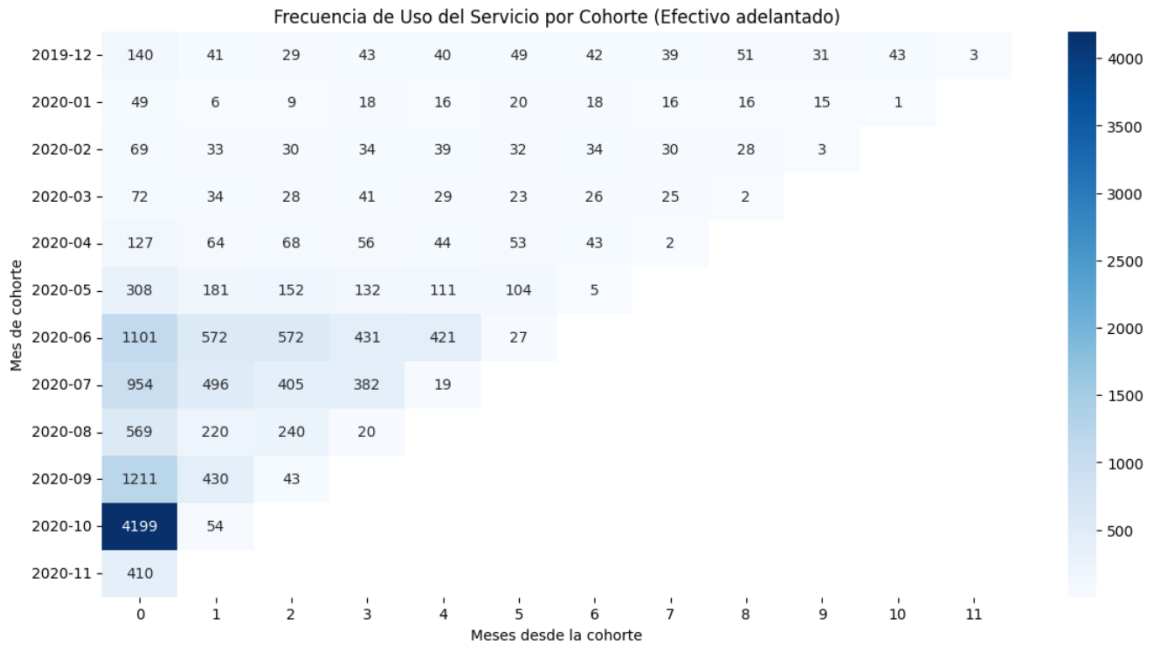
```
data['cash_request_received_date'] = pd.to_datetime(data['cash_request_received_date'])

data['cohort'] = data.groupby('user_id')['cash_request_received_date'].transform('min').dt.to_period('M') # Asignar cohorte: mes del primer adelanto de cada usuario
data['period'] = data['cash_request_received_date'].dt.to_period('M') # Periodo del adelanto actual (mes y año)

# Contar adelantos por cohorte y mes
cohort_counts = data.groupby(['cohort', 'period']).agg({'user_id': 'count'}).rename(columns={'user_id': 'usage_count'}).reset_index()
# Convertir períodos a timestamps para calcular diferencias
cohort_counts['cohort_start'] = cohort_counts['cohort'].dt.to_timestamp()
cohort_counts['period_start'] = cohort_counts['period'].dt.to_timestamp()

# Calcular meses desde la cohorte
cohort_counts['months_since_cohort'] = ((cohort_counts['period_start'] - cohort_counts['cohort_start']) / pd.Timedelta(days=30)).round().astype(int)
cohort_pivot = cohort_counts.pivot(index='cohort', columns='months_since_cohort', values='usage_count')

plt.figure(figsize=(14, 7)) # Visualización con heatmap
sns.heatmap(cohort_pivot, annot=True, fmt='.0f', cmap='Blues')
plt.title('Frecuencia de Uso del Servicio por Cohorte (Efectivo adelantado)')
plt.xlabel('Meses desde la cohorte')
plt.ylabel('Mes de cohorte')
plt.show()
```



Tasa de Incidentes

Determinar la tasa de incidentes, enfocándose específicamente en los incidentes de pago, para cada cohorte. Identificar si hay variaciones en las tasas de incidentes entre diferentes cohortes.

```python
# Convertir created_at a datetime
fees_so['created_at'] = pd.to_datetime(fees_so['created_at'])

# Definir cohorte por mes de creación
fees_so['cohort'] = fees_so['created_at'].dt.to_period('M')

# Filtrar incidentes de pago
incident_types = ['rejected_direct_debit', 'month_delay_on_payment']

# Filtrar solo fees de tipo incident y categoría incidentes de pago
incidents = fees_so[(fees_so['type'] == 'incident') & (fees_so['category'].isin(incident_types))]

# Contar incidentes por cohorte
incident_counts = incidents.groupby('cohort').size().rename('incident_count')

# Contar total fees por cohorte (o total cash requests únicos si prefieres)
total_counts = fees_so.groupby('cohort')['cash_request_id'].nunique().rename('total_cash_requests')

# Combinar en un dataframe
cohort_stats = pd.concat([incident_counts, total_counts], axis=1).fillna(0)

# Calcular tasa de incidentes
cohort_stats['incident_rate'] = cohort_stats['incident_count'] / cohort_stats['total_cash_requests']

print(cohort_stats)
```

✓ 0.0s

cohort	incident_count	total_cash_requests	incident_rate
2020-06	7.0	452	0.015487
2020-07	241.0	1054	0.228653
2020-08	476.0	2315	0.205616
2020-09	537.0	3540	0.151695
2020-10	925.0	8422	0.109831
2020-11	10.0	240	0.041667
2020-05	0.0	14	0.000000

Ingresos Generados por la Cohorte

Calcular el total de ingresos generados por cada cohorte a lo largo de los meses para evaluar el impacto financiero del comportamiento de los usuarios.

```python
data['created_at'] = pd.to_datetime(data['created_at'])
data['cohort'] = data['created_at'].dt.to_period('M')

valid_status = ['approved', 'money_sent', 'active', 'money_back']
data_valid = data[data['status'].isin(valid_status)]

income_by_cohort = data_valid.groupby('cohort')['amount'].sum().reset_index()
income_by_cohort.rename(columns={'amount': 'total_income'}, inplace=True)

print(income_by_cohort)
```

✓ 0.0s

	cohort	total_income
0	2019-12	14490.0
1	2020-01	9729.0
2	2020-02	9525.0
3	2020-03	15450.0
4	2020-04	29133.0
5	2020-05	54903.0
6	2020-06	148533.0
7	2020-07	175461.0
8	2020-08	171436.0
9	2020-09	222073.0
10	2020-10	497209.0
11	2020-11	9710.0