PARTE 1

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from scipy import stats
# Cargar el dataset
titanic_MD = pd.read_csv('titanic_MD.csv')

# Ver los valores únicos en la columna 'Sex'
unique_sex_values = titanic_MD['Sex'].unique()
print("Valores únicos en la columna 'Sex':", unique_sex_values)

# Reemplazar los valores no estándar (como '?') por NaN en la columna 'Sex'
titanic_MD['Sex'] = titanic_MD['Sex'].replace('?', np.nan)

# Generar el reporte de missing data después de reemplazar
missing_data = titanic_MD.isnull().sum()  # Número de valores faltantes por columna
missing_percentage = (missing_data / len(titanic_MD)) * 100  # Porcentaje de valores faltantes

# Crear el reporte
missing_report = pd.DataFrame({
    'Missing Values': missing_data,
    'Percentage': missing_percentage
})

# Ordenar el reporte
missing_report = missing_report.sort_values(by='Missing Values', ascending=False)

# Mostrar el reporte
print(missing_report)
```

```
Valores únicos en la columna 'Sex': ['?' 'female' 'male']
             Missing Values  Percentage
Sex                      51   27.868852
Age                      25   13.661202
Parch                    12    6.557377
Embarked                 12    6.557377
Fare                      8    4.371585
SibSp                     3    1.639344
PassengerId               0    0.000000
Survived                  0    0.000000
Pclass                    0    0.000000
Name                      0    0.000000
Ticket                    0    0.000000
Cabin                     0    0.000000
```

```
# 2. Imputaciones:
# 1. 'Sex' - Los valores faltantes (representados por '?') se imputarán con la moda, es decir, con el valor más frecuente: 'female' o 'male'
# 2. 'Age' - Dado que es una variable numérica, se imputará con la mediana, ya que es más robusta frente a valores atípicos.
# 3. 'Parch' - Los valores faltantes se imputarán con la mediana, ya que es una variable numérica.
# 4. 'Embarked' - Los valores faltantes se imputarán con la moda, ya que es una variable categórica y queremos usar el valor más frecuente.
# 5. 'Fare' - Los valores faltantes se imputarán con la mediana, ya que es una variable numérica y la mediana es más robusta a valores atípi
# 6. 'SibSp' - Los valores faltantes se imputarán con la mediana, ya que es una variable numérica.
```

```
# 3. Filas Completas
import pandas as pd

titanic_MD = pd.read_csv('titanic_MD.csv')

complete_rows = titanic_MD.dropna()

total_rows = len(titanic_MD)

num_complete_rows = len(complete_rows)

percentage_complete = (num_complete_rows / total_rows) * 100

print(f"Número de filas completas: {num_complete_rows}")
print(f"Porcentaje de filas completas: {percentage_complete:.2f}%")
```

```
Número de filas completas: 136
Porcentaje de filas completas: 74.32%
```

```
# 4. Imputaciones
```

```
# Cargar el dataset
titanic_MD = pd.read_csv('titanic_MD.csv')

# Paso 1: Reemplazar los valores "?" por NaN en la columna 'Sex'
titanic_MD['Sex'].replace('?', np.nan, inplace=True)

# Paso 2: Imputación para columnas numéricas (mediana)
# Columnas numéricas: 'Age', 'Parch', 'Fare', 'SibSp'
titanic_MD['Age'].fillna(titanic_MD['Age'].median(), inplace=True)
titanic_MD['Parch'].fillna(titanic_MD['Parch'].mode(), inplace=True)
titanic_MD['Fare'].fillna(titanic_MD['Fare'].mean(), inplace=True)
titanic_MD['SibSp'].fillna(titanic_MD['SibSp'].mode(), inplace=True)

# Paso 3: Imputación para columnas categóricas (moda)
# Columnas categóricas: 'Sex', 'Embarked'
sex_mode = titanic_MD['Sex'].mode()[0]  # Moda de 'Sex'
titanic_MD['Sex'].fillna(sex_mode, inplace=True)

embarked_mode = titanic_MD['Embarked'].mode()[0]  # Moda de 'Embarked'
titanic_MD['Embarked'].fillna(embarked_mode, inplace=True)

# Mostrar el dataset completo con las imputaciones aplicadas
print("Dataset después de imputación general (mediana para numéricas, moda para categóricas):")
print(titanic_MD)  # Imprimir todo el dataset
```

```
4      0.0    113783  26.5500         C103        S
..     ...       ...      ...          ...      ...
178    1.0     11751  56.9292          D35        S
179    0.0       695   5.0000  B51 B53 B55        S
180    0.0     11767  83.1583          C50        S
181    0.0    112053  30.0000          B42        S
182    0.0    111369  30.0000         C148        C

[183 rows x 12 columns]
<ipython-input-28-8392be571374>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assi
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting v

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]


  titanic_MD['Sex'].fillna(sex_mode, inplace=True)
<ipython-input-28-8392be571374>:23: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting v

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]
```

```
# Modelo de regresión Lineal
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer

# Cargar el dataset
titanic_MD = pd.read_csv('titanic_MD.csv')

# Paso 1: Reemplazar los valores "?" por NaN en la columna 'Sex'
titanic_MD['Sex'].replace('?', np.nan, inplace=True)

# Paso 2: Seleccionar las columnas sin valores faltantes para entrenar el modelo
# Filtrar las filas donde 'Age' no es nulo (datos para entrenar el modelo)
train_data = titanic_MD.dropna(subset=['Age'])  # Solo utilizamos filas donde 'Age' no tiene missing values

# Filtrar las filas donde 'Age' es nulo (datos para predecir)
test_data = titanic_MD[titanic_MD['Age'].isna()]

# Definir las columnas que no tienen missing values
# No utilizamos 'Sex', 'Age', 'Parch', 'Embarked', 'Fare', 'SibSp' ya que tienen missing values
columns_to_use = ['PassengerId', 'Pclass']  # Columnas sin missing values

# Seleccionar solo las columnas necesarias para el modelo
X_train = train_data[columns_to_use]
X_test = test_data[columns_to_use]

# Variable a predecir 'Age'
y_train = train_data['Age']

# Imputación de los valores faltantes en las columnas numéricas de X_train y X_test (si los hay)
imputer = SimpleImputer(strategy='mean')  # Usamos la media para las variables numéricas
X_train_imputed = imputer.fit_transform(X_train)  # Imputar en X_train
X_test_imputed = imputer.transform(X_test)  # Imputar en X_test

# Crear el modelo de regresión lineal
model = LinearRegression()

# Entrenar el modelo con las características imputadas
model.fit(X_train_imputed, y_train)

# Ahora predecimos los valores faltantes de 'Age' en las filas donde 'Age' es NaN
predicted_age = model.predict(X_test_imputed)

# Asignar las predicciones al dataset original
titanic_MD.loc[titanic_MD['Age'].isna(), 'Age'] = predicted_age

# Mostrar el dataset con las imputaciones aplicadas por regresión lineal
print("Dataset después de imputación de 'Age' con regresión lineal:")
print(titanic_MD[['Age', 'PassengerId', 'Pclass']].head())  # Mostrar algunas filas para verificar
```

```
Dataset después de imputación de 'Age' con regresión lineal:
         Age  PassengerId  Pclass
0  38.000000            2       1
1  35.000000            4       1
2  54.000000            7       1
3  18.036912           11       3
4  58.000000           12       1
<ipython-input-39-c3a7ea283554>:10: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assign
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  titanic_MD['Sex'].replace('?', np.nan, inplace=True)
```

```
import pandas as pd
import numpy as np

# Cargar el dataset
titanic_MD = pd.read_csv('titanic_MD.csv')

# Reemplazar los valores "?" por NaN en la columna 'Sex'
titanic_MD['Sex'].replace('?', np.nan, inplace=True)

# Imputación de valores faltantes usando el Percentile Approach
def imputar_outliers_percentile(df, column_name, percentile=50):
    # Calcular el percentil de la columna
    percentile_value = np.percentile(df[column_name].dropna(), percentile)

    # Imputar los valores faltantes con el percentil calculado
    df[column_name].fillna(percentile_value, inplace=True)

# Aplicar imputación para las columnas 'Age', 'Parch' y 'Fare' usando el Percentile Approach (percentil 50)
columns_to_impute_percentile = ['Age', 'Parch', 'Fare']

for column in columns_to_impute_percentile:
    imputar_outliers_percentile(titanic_MD, column, percentile=50)

# Mostrar el dataframe completo con las columnas imputadas
print("Dataset después de imputación de missing values con el método de Percentiles (Percentil 50):")
print(titanic_MD[['Age', 'Parch', 'Fare']])  # Mostrar toda la tabla con las columnas imputadas
```

```
Dataset después de imputación de missing values con el método de Percentiles (Percentil 50):
      Age  Parch     Fare
0    38.0    0.0  71.2833
1    35.0    0.0  53.1000
2    54.0    0.0  51.8625
3    35.5    0.0  16.7000
4    58.0    0.0  26.5500
..    ...    ...      ...
178  47.0    1.0  56.9292
179  35.5    0.0   5.0000
180  56.0    0.0  83.1583
181  19.0    0.0  30.0000
182  35.5    0.0  30.0000

[183 rows x 3 columns]
<ipython-input-42-977a124047fc>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  titanic_MD['Sex'].replace('?', np.nan, inplace=True)
<ipython-input-42-977a124047fc>:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assign
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df[column_name].fillna(percentile_value, inplace=True)
```

```
## 5.
## Después de usar los metodos de imputaciones para las distintas columnas segun se fue posible debido a la los tipos de datos, se observo q
## como la media, la mediana o la moda. Esto se debe a que los datos que tienen valores faltantes como la edad, el sexo, el fare, el Parched
## son muy utiles para predecir, mientras que otras columnas tenian valores más caategoricos con los cuales la moda es extremadamente útil
## Sin embargo se puede ver que el metodo de percentiles también fue muy útil para estos casos
## La regresión lineal fue algo complicada y no muy acertada ya que se utilizaron las columnas que no tuvieran valores faltantes para predic
## seguramente el bajo rendimiento de la regresion lineal se debio de dar por la poca relación que tenian las variables



## 6. Conclusiones

## Para poder realizar una imputación de datos se tiene que observar con mucho detenimiento los datos primero, es muy importante entenderlos
## Saber si es un dato numerico, texto, un dato categorico es la base con la cual decidimos que imputación se debe de seguir
## también sirve ver el comportamiento general de los datos para saber que metodo nos dara los resultados más precisos
## además hay que tener cuidado con los datos faltantes ya que por ejemplo con la variable sexo no es que hubieran casillas vacias sino que
```

```
## PARTE 2


# Solo volvi a preprocesar los datos para normalizarlos
titanic_MD = pd.read_csv('titanic_MD.csv')
titanic_MD['Sex'].replace('?', np.nan, inplace=True)
titanic_MD['Age'].fillna(titanic_MD['Age'].median(), inplace=True)
titanic_MD['Parch'].fillna(titanic_MD['Parch'].mode(), inplace=True)
titanic_MD['Fare'].fillna(titanic_MD['Fare'].median(), inplace=True)
titanic_MD['SibSp'].fillna(titanic_MD['SibSp'].mode(), inplace=True)


sex_mode = titanic_MD['Sex'].mode()[0]
titanic_MD['Sex'].fillna(sex_mode, inplace=True)

embarked_mode = titanic_MD['Embarked'].mode()[0]
titanic_MD['Embarked'].fillna(embarked_mode, inplace=True)

print("Dataset después de imputación general (mediana para numéricas, moda para categóricas):")
print(titanic_MD)

## NORMALIZACION TITANIC MD
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

# Cargar el dataset
titanic_MD = pd.read_csv('titanic_MD.csv')

# Paso 1: Reemplazar los valores "?" por NaN en la columna 'Sex'
titanic_MD['Sex'].replace('?', np.nan, inplace=True)

# Paso 2: Imputación para columnas numéricas (ya realizada previamente)
titanic_MD['Age'].fillna(titanic_MD['Age'].median(), inplace=True)
titanic_MD['Parch'].fillna(titanic_MD['Parch'].mode()[0], inplace=True)
titanic_MD['Fare'].fillna(titanic_MD['Fare'].median(), inplace=True)
titanic_MD['SibSp'].fillna(titanic_MD['SibSp'].mode()[0], inplace=True)

# Paso 3: Imputación para columnas categóricas (ya realizada previamente)
sex_mode = titanic_MD['Sex'].mode()[0]
titanic_MD['Sex'].fillna(sex_mode, inplace=True)
embarked_mode = titanic_MD['Embarked'].mode()[0]
titanic_MD['Embarked'].fillna(embarked_mode, inplace=True)

# Seleccionar las columnas numéricas que vamos a normalizar
numerical_columns = ['Age', 'Parch', 'Fare', 'SibSp']

# a. Standardization (Z-score normalization)
scaler_standard = StandardScaler()
titanic_MD[numerical_columns] = scaler_standard.fit_transform(titanic_MD[numerical_columns])

# b. MinMax Scaling (Transformación a un rango [0, 1])
scaler_minmax = MinMaxScaler()
titanic_MD[numerical_columns] = scaler_minmax.fit_transform(titanic_MD[numerical_columns])

# c. MaxAbsScaler (Escala en el rango [-1, 1], manteniendo la signatura)
scaler_maxabs = MaxAbsScaler()
titanic_MD[numerical_columns] = scaler_maxabs.fit_transform(titanic_MD[numerical_columns])

# Mostrar el dataset después de la normalización
print("Dataset después de la normalización:")
print(titanic_MD[numerical_columns])  # Mostrar las columnas numéricas normalizadas
```

```
Dataset después de imputación general (mediana para numéricas, moda para categóricas):
     PassengerId  Survived  Pclass  \
0              2         1       1
1              4         1       1
2              7         0       1
3             11         1       3
4             12         1       1
..           ...       ...     ...
178          872         1       1
179          873         0       1
180          880         1       1
```

```
181          888         1       1
182          890         1       1

                                                  Name     Sex   Age  SibSp  \
0    Cumings, Mrs. John Bradley (Florence Briggs Th...    male  38.0    1.0
1         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0    1.0
2                              McCarthy, Mr. Timothy J    male  54.0    0.0
3                    Sandstrom, Miss. Marguerite Rut  female  35.5    1.0
4                           Bonnell, Miss. Elizabeth  female  58.0    NaN
..                                               ...     ...   ...    ...
178   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0    1.0
179                          Carlsson, Mr. Frans Olof    male  35.5    0.0
180      Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0    0.0
181                      Graham, Miss. Margaret Edith    male  19.0    0.0
182                            Behr, Mr. Karl Howell    male  35.5    0.0

     Parch    Ticket     Fare        Cabin Embarked
0      0.0  PC 17599  71.2833          C85        C
1      0.0    113803  53.1000         C123        S
2      0.0     17463  51.8625          E46        S
3      NaN   PP 9549  16.7000           G6        S
4      0.0    113783  26.5500         C103        S
..     ...       ...      ...          ...      ...
178    1.0     11751  56.9292          D35        S
179    0.0       695   5.0000  B51 B53 B55        S
180    NaN     11767  83.1583          C50        S
181    0.0    112053  30.0000          B42        S
182    0.0    111369  30.0000         C148        C

[183 rows x 12 columns]
Dataset después de la normalización:
          Age  Parch      Fare     SibSp
0    0.468892   0.00  0.139136  0.333333
1    0.430956   0.00  0.103644  0.333333
2    0.671219   0.00  0.101229  0.000000
3    0.437279   0.00  0.032596  0.333333
4    0.721801   0.00  0.051822  0.000000
..        ...    ...       ...       ...
178  0.582701   0.25  0.111118  0.333333
179  0.437279   0.00  0.009759  0.000000
180  0.696510   0.00  0.162314  0.000000
181  0.228629   0.00  0.058556  0.000000
182  0.437279   0.00  0.058556  0.000000

[183 rows x 4 columns]
<ipython-input-46-a1226827785d>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assi
```

```
## NORMALIZACION TITANIC
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

# Cargar el dataset "titanic.csv"
titanic = pd.read_csv('titanic.csv')

# Seleccionar las columnas numéricas que vamos a normalizar
numerical_columns = ['Age', 'Parch', 'Fare', 'SibSp']

# a. Standardization (Z-score normalization)
scaler_standard = StandardScaler()
titanic[numerical_columns] = scaler_standard.fit_transform(titanic[numerical_columns])

# b. MinMax Scaling (Transformación a un rango [0, 1])
scaler_minmax = MinMaxScaler()
titanic[numerical_columns] = scaler_minmax.fit_transform(titanic[numerical_columns])

# c. MaxAbsScaler (Escala en el rango [-1, 1], manteniendo la signatura)
scaler_maxabs = MaxAbsScaler()
titanic[numerical_columns] = scaler_maxabs.fit_transform(titanic[numerical_columns])

# Mostrar el dataset después de la normalización
print("Dataset después de la normalización:")
print(titanic[numerical_columns])  # Mostrar las columnas numéricas normalizadas
```

```
Dataset después de la normalización:
          Age  Parch      Fare     SibSp
0    0.468892   0.00  0.139136  0.333333
1    0.430956   0.00  0.103644  0.333333
2    0.671219   0.00  0.101229  0.000000
3    0.038948   0.25  0.032596  0.333333
```

```
4    0.721801   0.00  0.051822  0.000000
..        ...    ...       ...       ...
178  0.582701   0.25  0.102579  0.333333
179  0.405665   0.00  0.009759  0.000000
180  0.696510   0.25  0.162314  0.000000
181  0.228629   0.00  0.058556  0.000000
182  0.317147   0.00  0.058556  0.000000

[183 rows x 4 columns]
```

TITANIC MD y TITANIC muestran resultados muy similares en términos de las métricas clave después de la normalización de las columnas numérica
Age, Parch, Fare, y SibSp tienen medias, desviaciones estándar, máximos y mínimos muy cercanos entre ambos datasets.
La normalización ha tenido un efecto mínimo en la distribución de las columnas, lo cual es esperado, ya que las transformaciones de normaliza
Los pequeños cambios en la media o el mínimo en algunas columnas pueden ser el resultado de diferencias en los valores originales antes de la