Import pandas as pd

Import numpy as np

url = ‘ any link’

df = pd.read\_csv(url, engine=’pyarrow’, dtype\_backend=’pyarrow’)

df.shape

df.head()

df.describe() : Gives statistical summary of each column

in this the count means non zero values, not the entire count

df.dtypes : Gives datatypes

df.select\_dtypes(‘string’): Gives the columns with datatype string

df.select\_dtypes(‘string').describe() : Gives the description of string type columns

df.select\_dtypes(‘string').describe().T : Transposes the above data

If the unique entries in a column or categories in a column are not too much, we call it low cardinality in statistics. In these cases it’s better to convert these values to column. Allows to store memory.

(df

.select\_dtypes(‘string’)

.memory\_usage(deep=True)

.sum()

)

HERE it is selecting strings next checking byte memory usage of each column & then summing the memory usage

(df

.select\_dtypes(‘string’)

.memory\_usage(deep=True)

.sum()

)

Here we did the same, but we used category which changes the categorical values to numeric, which saves lot of memory. So use categorical columns where cardinality is less.

(df

.select\_dtypes(‘string’)

.eq(‘ ‘)

.any(axis=’colums’)

)

This returns, true or false to each row, with row true as missing value in that row, else FALSE

(df

[~df

.select\_dtypes(‘string’)

.eq(‘ ‘)

.any(axis=’colums’)]

)

This negates the above code, It actually gives the rows with the missing values

(df

.select\_dtypes(‘string’)

.eq(‘NA’)

.mean( )

.mul(100)

.pipe(lamda ser: ser[ser > 0])

)

This checjs missing string values encoded as NA

It gives out how many percente of values are missing of each column.

(df

.query(‘`Pool QC`.isna()’)

)

Here we are checking which column the Pool QC is missing

(df

.query(‘`Pool QC`== “NA”’)

)

Here same but checking where Pool QC has NA, because data can have any NA or missing values.

(df

.assign(\*\*df

.select\_dtypes(‘string’).replace(‘ ‘, ‘Not Applicable’))

)

Here we replaced ‘ ‘ with Not applicable & assignedthat to the dataframe

(df

.Elecrical

.value\_counts( )

)

This checks column Elecectrical and returns its unique entries with their count, helps in identifying the missing value counts.

(df

.query(Electrical == “”’)

)

This returns the row where electrical is zero. This is using kind of querying language to retirew.

So what we can do is, take the whole dataframe and replace the “ “ with Not applicable.

(df

.assign\*\*(df

.select\_dtypes(‘string’)

.replace(‘ ‘, ‘Not Applicable’)

.astype(‘category’)

))

Here I also used astype category so that the valuss are converted to category so that the memory isused efficiently.