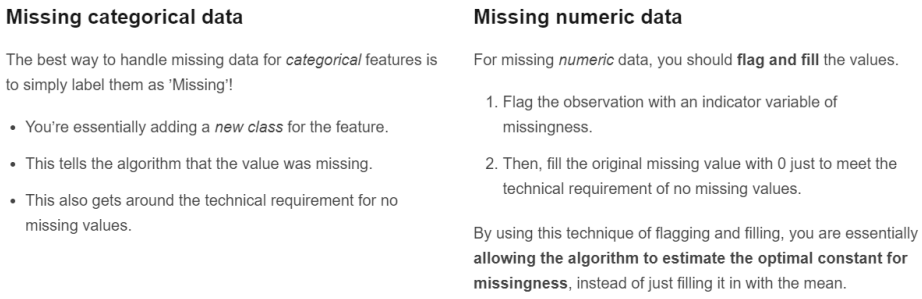
CLEANING DATA

Inconsistent Data — Capitalization, Addresses and more

Missing Data

Missingness is informative in itself



Missing categorical data

The best way to handle missing data for *categorical* features is to simply label them as 'Missing'!

- You're essentially adding a *new class* for the feature.
- This tells the algorithm that the value was missing.
- This also gets around the technical requirement for no missing values.

Missing numeric data

For missing *numeric* data, you should **flag and fill** the values.

1. Flag the observation with an indicator variable of missingness.
2. Then, fill the original missing value with 0 just to meet the technical requirement of no missing values.

By using this technique of flagging and filling, you are essentially **allowing the algorithm to estimate the optimal constant for missingness**, instead of just filling it in with the mean.

Outliers, Outliers are innocent until proven guilty, you must have a good reason for removing an outlier

Duplicate data

Irrelevant data

For instance, you can check for **typos** or **inconsistent capitalization**

Mis labelled classes (CS and computer science combine kar)

Get rid of extra spaces- “welcome home”

DOCUMENTATION – DATA CLEANING

Read Data

df = pd.read_csv("Name.csv")

COLUMN OPERATIONS

Check heading

df.columns

Replacing spaces between with underscore and removing spaces before & after

df.columns = df.columns.str.strip().str.replace(' ', '_')

Drop columns/row

df.drop([col1,col2,..]/ rownumber , inplace=True, axis=1/not needed)

Rename columns

df.rename(columns = {‘n1’:’new1’,’n2’:’new2’,…}, inplace=True)

Rearrange columns

df=df[[“ “,” “, “ “]]

Create unique column

```
a=[]
for i in range(len(df)):
    a.append(i)
df[‘Index’]=a
```

DESCRIBE

df.describe()

df.info()

df[column name]

Return column name

df.loc[index]

Whatever index we have set, it finds the one mentioned and returns those rows

For both above things, if we want multiple put 2 brackets and write q1,q2,q3,…

df.iloc[number]

actual row number

df.loc[rows,columns]

df.loc[[],[]] for multiple rows and columns

df.loc[df[col]==”a”, []] rows with a in “col” and gives all columns mentioned in []

Setting Index

df.set_index("QUANTITYORDERED",inplace=True)

df.loc[30]

df.reset_index(inplace = True)

Unique values

df.col.unique()

Number of unique values

df.col.nunique()

Check if all values have no decimals, integer

for a in df[col]:

```
    if(a%1 != 0):
            print(a)
```

Split columns

```
new= df[“col”].str.strip(“-“,n=1,expand=”True”)
df[n1]=new[0]
df[n2]=new[1]
```

Convert to string

```
df[col].astype(str)
```

Check if each combo is unique (eg :- Place and PIN code)

```
df[new]=df[col]+df[col]
df.new.unique()
```

Replace values/part of values in column

```
df.col.replace({‘-‘:’/’},regex=True(for part of value),inplace=True)
```

Convert col to data time format

```
df.col=pd.to_datetime(df.col,format='%Y/%m/%d')
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html

Detect duplicates

```
columns= df.columns
data.duplicated(columns).sum()
```

Remove duplicates

```
Df= df.drop_duplicates(columns, keep=’first’)
```

Check the format of a given column

```
import re
rex = re.compile("^[A-Z]{2}-[0-9]{4}-[0-9]{6}$")
for a in df["Order_ID"]:
   if rex.match(a) is None:
      print("True")
```

https://docs.python.org/3.3/howto/regex.html

Merge dataframes

Final=pd.merge(df1,df2, on=(“”,””))

Number of null values in each column

df.isnull().sum()

Deleting null values

Df.dropna(axis= , how=””, thresh= , subset= , inplace=)

Axis= 0 for rows and 1 for columns

How= “any” if any value in row/column is null, then it drops, “all” if all values in row/column are null, then it drops

Thresh= threshold value, if 2, then drops if there are >=2 values

Subset= checks only those rows/columns

Inplace= True for replacing in that place, false otherwise

https://www.journaldev.com/33492/pandas-dropna-drop-null-na-values-from-dataframe

df= df[df.col != " "] for specific column null value removal

Count of each unique value

df.colname.value_counts()

Add columns and create new column

Sum=df[“ “]+df[“ “]

Df[“newcol”] = sum

Compare columns

(Df[col]).equals(df[col])

Returns true if same

Different values in one column vs other / one array vs other

np.setdiff1d(b,df4['customer_id'])

WOW FUNCTION

pd.melt(df,id_vars=[“”,””],value_vars[“”,””,””],var_name=””, value_name=””)

takes all other columns into one column.

new_df = df["Year_span"].str.split("-", n=1, expand = True).rename({0: 'Start_Year', 1: 'Finish_Year'}, axis=1).replace([None], [np.nan]).fillna(method='ffill', axis=1)

df = pd.concat((df, new_df), axis=1)

OUTLIERS

def outlier(datacolumn):

```
sorted(datacolumn)

Q1,Q3 = np.percentile(datacolumn , [25,75])

IQR = Q3 - Q1

lower_range = Q1 - (1.5 * IQR)

upper_range = Q3 + (1.5 * IQR)

print (lower_range,upper_range)

plt.hist(datacolumn)

plt.show()
```