Data preprocessing

Avijit Bose
Assistant Professor
Department of Computer science & engineering
MCKVIE,Liluah,Howrah-711204

Why Data preprocessing is required?

- Real world data has got several problems

Incomplete:- Many times it has been seen that real life data are incomplete in the sense that they have missing attribute values, missing certain attributes of importance and noisy which means have many outliers and inconsistent which means it consists of discrepancies in order of names.

Steps in data preprocessing

- Data cleaning:-

(a) Data cleaning, also called data cleansing or scrubbing.

- Fill in missing values, smooth noisy data, identify or remove the outliers, and resolve inconsistencies.
- Data cleaning is required because source systems contain “dirty data” that must be cleaned.

Steps in data cleaning

- (a) Parsing
- Parsing locates and identifies individual data elements in the source files and then isolates these data elements in the target files.
- Example includes parsing the first, middle and the last name.

Steps in data cleaning

(b)Correcting

- Correct parsed individual data components using sophisticated data algorithms and secondary data sources.
- Example includes replacing a vanity address and adding a zip code.

Steps in data cleaning

- (c) Standardizing
- Standardizing applies conversion routines to transform data into its preferred and consistent format using both standard and custom business rules.
- Examples include adding a pre name, replacing a nickname.

Steps in data cleaning

- (d) Matching
- Searching and matching records within and across the parsed, corrected and standardized data based on predefined business rules to eliminate duplications.
- Examples include identifying similar names and addresses.

Steps in data cleaning

- (e) consolidating
- Analyzing and identifying relationships between matched records and consolidating/merging them into one representation.

Steps in data cleaning

- Data cleansing must deal with many types of possible errors:

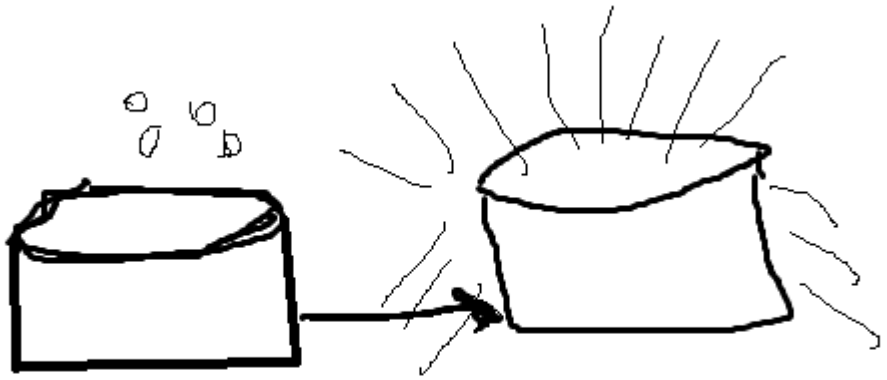
These include missing data and incorrect data at one source.

Steps in data cleaning

Data staging

- Accumulates data from asynchronous sources.
- At a predefined cutoff time, data in the staging file is transformed and loaded to the warehouse.
- There is usually no end user access to the staging file.
- An operational data store may be used for data staging.

Data staging diagram



Data Cleaning

Data Integration & Transformation

- Data integration:-Combines data from multiple sources into a coherent data store e.g. data warehouse.

- Sources may include multiple databases, data cubes or data files.

Issues in data integration:

- Schema integration:

- Integrate metadata from different sources.
- Entity identification problem: identify real world entities from multiple data sources, e.g. A cust-id=B.cust#.

Detecting and resolving data value conflicts:

- For the same real world entity, attribute values from different sources are different.
- Possible reasons: different representations, different scales.

Redundant data occur often when integration of multiple databases:

- The same attribute may have different names in different databases.

Issues in data integration:

- Data Transformation: Transformation process deals with rectifying any inconsistency (if any).
- One of the most common transformation issues is ‘Attribute Naming Inconsistency’. It is common for the given data element to be referred to by different data names in different databases.
- Eg Employee Name may be EMP_NAME in one database, ENAME in the other.
- Thus one set of Data Names are picked and used consistently in the data warehouse.
- Once all the data elements have right names, they must be converted to common formats.

Issues in data integration:

- Data Reduction:

- Obtains reduced representation in volume but produces the same or similar analytical results.

Need for data reduction:

- Reducing the number of attributes
- Reducing the number of attribute values
- Reducing the number of tuples

Issues in data integration:

- Discretization and Concept Hierarchy Generation(or summarization):
- Discretization: Reduce the number of values for a given continuous attribute by divide the range of a continuous attribute into intervals.
- Interval labels can then be used to replace actual data values.
- Concept Hierarchies: Reduce the data by collecting and replacing low level concepts(such as numeric values for the attribute age)by higher level concepts(such as young, middle-aged or senior).

Removing unwanted data from data frame

- We will use here several approaches on how to drop rows from the dataframe based on certain condition applied on a column. Retain all those rows for which the applied condition on the given column evaluates to True.
- Solution:- We will use vectorization to filter out such rows from the dataset which satisfy the applied condition.

Method

- import pandas as pd

```
 # Read the csv file and construct the
 # dataframe
df = pd.read_csv('nba.csv')
 # Visualize the dataframe
print(df.head(15)
# Print the shape of the dataframe
print(df.shape)
```

Removing unwanted data from data frame

- In this data frame, currently, we are having 458 rows and 9 columns. Let’s use vectorization operation to filter out all those rows which satisfy the given condition.
- # Filter all rows for which the player's
- # age is greater than or equal to 25

```
df_filtered = df[df['Age'] >= 25]
# Print the new dataframe
print(df_filtered.head(15)
# Print the shape of the dataframe
print(df_filtered.shape)
```

Removing unwanted data from data frame

- We can use the DataFrame.drop() function to drop such rows which does not satisfy the given condition.

```
# importing pandas as pd
 import pandas as pd
# Read the csv file and construct the
# dataframe
df = pd.read_csv('nba.csv')

 # First filter out those rows which
# does not contain any data
df = df.dropna(how = 'all')

 # Filter all rows for which the player's
 # age is greater than or equal to 25
 df.drop(df[df['Age'] < 25].index, inplace = True)

 # Print the modified dataframe
 print(df.head(15))

 # Print the shape of the dataframe
  print(df.shape)
```

Removing unwanted data from data frame

- we are going to see several examples of how to drop rows from the dataframe based on certain conditions applied on a column.

- Pandas provide data analysts a way to delete and filter data frame using dataframe.drop() method. We can use this method to drop such rows that do not satisfy the given conditions.

Removing unwanted data from data frame

```
 # import pandas library
 import pandas as pd
# dictionary with list object in values
details = {
'Name' : ['Ankit', 'Aishwarya', 'Shaurya',
      'Shivangi', 'Priya', 'Swapnil'],
  'Age' : [23, 21, 22, 21, 24, 25],
 'University' : ['BHU', 'JNU', 'DU', 'BHU',
                 'Geu', 'Geu'],
  }

# creating a Dataframe object
df = pd.DataFrame(details, columns = ['Name', 'Age',
                                      'University'],
           index = ['a', 'b', 'c', 'd', 'e',
                    'f'])

df
```

Delete rows based on condition on a column.

```
 # import pandas library
  import pandas as pd

  # dictionary with list object in values
details = {
    'Name' : ['Ankit', 'Aishwarya', 'Shaurya',
          'Shivangi', 'Priya', 'Swapnil'],
   'Age' : [23, 21, 22, 21, 24, 25],
   'University' : ['BHU', 'JNU', 'DU', 'BHU',
                'Geu', 'Geu'],
}
```

Delete rows based on condition on a column

- ```
  # creating a Dataframe object
  df = pd.DataFrame(details, columns = ['Name', 'Age',
                                        'University'],
                    index = ['a', 'b', 'c', 'd', 'e', 'f'])

 # get names of indexes for which
  # column Age has value 21
  index_names = df[df['Age'] == 21].index

   # drop these row indexes
   # from dataFrame
 df.drop(index_names, inplace = True)

df
```
```

Delete rows based on multiple conditions on a column.

```
# import pandas library
 import pandas as pd

 # dictionary with list object in values
  details = {
    'Name' : ['Ankit', 'Aishwarya', 'Shaurya',
          'Shivangi', 'Priya', 'Swapnil'],
     'Age' : [23, 21, 22, 21, 24, 25],
     'University' : ['BHU', 'JNU', 'DU', 'BHU',
               'Geu', 'Geu'],
   }

 # creating a Dataframe object
 df = pd.DataFrame(details, columns = ['Name', 'Age',
                              'University'],
            index = ['a', 'b', 'c', 'd', 'e', 'f'])
```

Delete rows based on multiple conditions on a column.

```
  # get names of indexes for which column Age has value >= 21
   # and <= 23
   index_names = df[ (df['Age'] >= 21) & (df['Age'] <= 23)].index
   # drop these given row
   # indexes from dataFrame
   df.drop(index_names, inplace = True)
    df
```

Delete rows based on multiple conditions on different columns.

```
# import pandas library
 import pandas as pd

 # dictionary with list object in values
  details = {
  'Name' : ['Ankit', 'Aishwarya', 'Shaurya',
          'Shivangi', 'Priya', 'Swapnil'],
   'Age' : [23, 21, 22, 21, 24, 25],
   'University' : ['BHU', 'JNU', 'DU', 'BHU',
               'Geu', 'Geu'],
 }

# creating a Dataframe object
 df = pd.DataFrame(details, columns = ['Name', 'Age',
                                'University'],
            index = ['a', 'b', 'c', 'd', 'e', 'f'])
```

Delete rows based on multiple conditions on different columns.

```
# get names of indexes for which
# column Age has value >= 21
# and column University is BHU
index_names = df[ (df['Age'] >= 21) &
(df['University'] == 'BHU')].index

# drop these given row
# indexes from dataFrame
df.drop(index_names, inplace = True)
df
```

How to drop rows in Pandas Data Frame by index labels?

Let’s create a sample dataframe

```
# import pandas library
import pandas as pd

# dictionary with list object in values
details = {
   'Name' : ['Ankit', 'Aishwarya', 'Shaurya','Shivangi'],
   'Age' : [23, 21, 22,21],
   'University' : ['BHU', 'JNU', 'DU', 'BHU'],
}

# creating a Dataframe object
df = pd.DataFrame(details,columns = ['Name','Age','University'],
                  index = ['a', 'b', 'c', 'd'])

    df
```

How to drop rows in Pandas Data Frame by index labels?

- Delete a single Row in DataFrame by Row Index Label

```
import pandas as pd

# dictionary with list object in values
details = {
    'Name' : ['Ankit', 'Aishwarya', 'Shaurya', 'Shivangi'],
   'Age' : [23, 21, 22, 21],
   'University' : ['BHU', 'JNU', 'DU', 'BHU'],
}
```

How to drop rows in Pandas Data Frame by index labels?

- # creating a Dataframe object

```
df = pd.DataFrame(details, columns = ['Name',
'Age', 'University'],
                  index = ['a', 'b', 'c', 'd'])
# return a new dataframe by dropping a
# row 'c' from dataframe
update_df = df.drop('c')

update_df
```

How to drop rows in Pandas Data Frame by index labels?

- Delete Multiple Rows in DataFrame by Index Labels

```
# import pandas library
import pandas as pd

# dictionary with list object in values
  details = {
    'Name' : ['Ankit', 'Aishwarya', 'Shaurya', 'Shivangi'],
    'Age' : [23, 21, 22, 21],
    'University' : ['BHU', 'JNU', 'DU', 'BHU'],
}
```

How to drop rows in Pandas Data Frame by index labels?

- # creating a Dataframe object

```
df = pd.DataFrame(details, columns =
['Name', 'Age', 'University'],
                  index = ['a', 'b', 'c', 'd'])
# return a new dataframe by dropping a row
# 'b' & 'c' from dataframe
update_df = df.drop(['b', 'c'])
update_df
```

How to drop rows in Pandas Data Frame by index labels?

- Delete a Multiple Rows by Index Position in Data Frame

```
# import pandas library
import pandas as pd

# dictionary with list object in values
details = {
 'Name' : ['Ankit', 'Aishwarya', 'Shaurya', 'Shivangi'],
 'Age' : [23, 21, 22, 21],
 'University' : ['BHU', 'JNU', 'DU', 'BHU'],
}
```

How to drop rows in Pandas Data Frame by index labels?

```
# creating a Dataframe object
df = pd.DataFrame(details, columns = ['Name',
'Age', 'University'],
                  index = ['a', 'b', 'c', 'd'])

# return a new dataframe by dropping a row
# 'b' & 'c' from dataframe using their
# respective index position
update_df = df.drop([df.index[1], df.index[2]])
update_df
```

How to drop rows in Pandas Data Frame by index labels?

- Delete rows from dataFrame in Place

```
# import pandas library
 import pandas as pd
 # dictionary with list object in values
 details = {
  'Name' : ['Ankit', 'Aishwarya', 'Shaurya', 'Shivangi'],
       'Age' : [23, 21, 22, 21],
   'University' : ['BHU', 'JNU', 'DU', 'BHU'],
 }
```

How to drop rows in Pandas Data Frame by index labels?

```
# creating a Dataframe object
df = pd.DataFrame(details, columns = ['Name', 'Age', 'University'],
                  index = ['a', 'b', 'c', 'd'])

# dropping a row 'c' & 'd' from actual dataframe
df.drop(['c', 'd'], inplace = True )

df
```

Python | Delete rows/columns from DataFrame using Pandas.drop()

- Dropping Rows by index label

```
# importing pandas module
import pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name" )

# dropping passed values
 data.drop(["Avery Bradley", "John Holland", "R.J. Hunter",
                    "R.J. Hunter"], inplace = True)

 # display
  data
```

Python | Delete rows/columns from DataFrame using Pandas.drop()

- Dropping columns with column name
- In his code, Passed columns are dropped using column names. axis parameter is kept 1 since 1 refers to columns.

Python | Delete rows/columns from DataFrame using Pandas.drop()

```
# importing pandas module
 import pandas as pd
 # making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name" )
 # dropping passed columns
data.drop(["Team", "Weight"], axis = 1, inplace = True)

 # display
 data
```

Python | Delete rows/columns from DataFrame using Pandas.drop()

- As shown in the output images, the new output doesn’t have the passed columns. Those values were dropped since axis was set equal to 1 and the changes were made in the original data frame since inplace was True.

How to drop one or multiple columns in Pandas Dataframe

- Let's Create a simple dataframe with dictionary of lists, say column names are A, B, C, D, E.

```
# Import pandas package
  import pandas as pd

   # create a dictionary with five fields each
     data = {
         'A':['A1', 'A2', 'A3', 'A4', 'A5'],
         'B':['B1', 'B2', 'B3', 'B4', 'B5'],
         'C':['C1', 'C2', 'C3', 'C4', 'C5'],
         'D':['D1', 'D2', 'D3', 'D4', 'D5'],
         'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

  # Convert the dictionary into DataFrame
   df = pd.DataFrame(data)

    df
```

How to drop one or multiple columns in Pandas Dataframe

- Drop Columns from a Dataframe using drop() method.

```
# Import pandas package
 import pandas as pd

 # create a dictionary with five fields each
    data = {
        'A':['A1', 'A2', 'A3', 'A4', 'A5'],
       'B':['B1', 'B2', 'B3', 'B4', 'B5'],
      'C':['C1', 'C2', 'C3', 'C4', 'C5'],
       'D':['D1', 'D2', 'D3', 'D4', 'D5'],
       'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

 # Convert the dictionary into DataFrame
 df = pd.DataFrame(data)

 # Remove column name 'A'
   df.drop(['A'], axis = 1)
```

How to drop one or multiple columns in Pandas Dataframe

Remove specific multiple columns.

```
# Import pandas package
import pandas as pd

# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

      # Remove two columns name is 'C' and 'D'
      df.drop(['C', 'D'], axis = 1)

    # df.drop(columns =['C', 'D'])
```

How to drop one or multiple columns in Pandas Dataframe

Remove columns as based on column index.

```
# Import pandas package
import pandas as pd

# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Remove three columns as index base
df.drop(df.columns[[0, 4, 2]], axis = 1, inplace = True)

df
```

Remove all columns between a specific column to another columns.

```
# Import pandas package
import pandas as pd
# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Remove all columns between column index 1 to 3
df.drop(df.iloc[:, 1:3], inplace = True, axis = 1)

df
```

Drop Columns from a Dataframe using ix() and drop() method.

Remove all columns between a specific column name to another columns name.

```
# Import pandas package
import pandas as pd

# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Remove all columns between column name 'B' to 'D'
df.drop(df.ix[:, 'B':'D'].columns, axis = 1)
```

Drop Columns from a Dataframe using loc[] and drop() method.

Remove all columns between a specific column name to another columns name.

```
# Import pandas package
import pandas as pd

# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Remove all columns between column name 'B' to 'D'
df.drop(df.loc[:, 'B':'D'].columns, axis = 1)
```

Different loc() and iloc() is iloc() exclude last column range element.

Drop Columns from a Dataframe by iterative way.

Remove all columns between a specific column name to another columns name.

```
# Import pandas package
import pandas as pd

# create a dictionary with five fields each
data = {
   'A':['A1', 'A2', 'A3', 'A4', 'A5'],
   'B':['B1', 'B2', 'B3', 'B4', 'B5'],
   'C':['C1', 'C2', 'C3', 'C4', 'C5'],
   'D':['D1', 'D2', 'D3', 'D4', 'D5'],
   'E':['E1', 'E2', 'E3', 'E4', 'E5'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
for col in df.columns:
   if 'A' in col:
      del df[col]

df
```