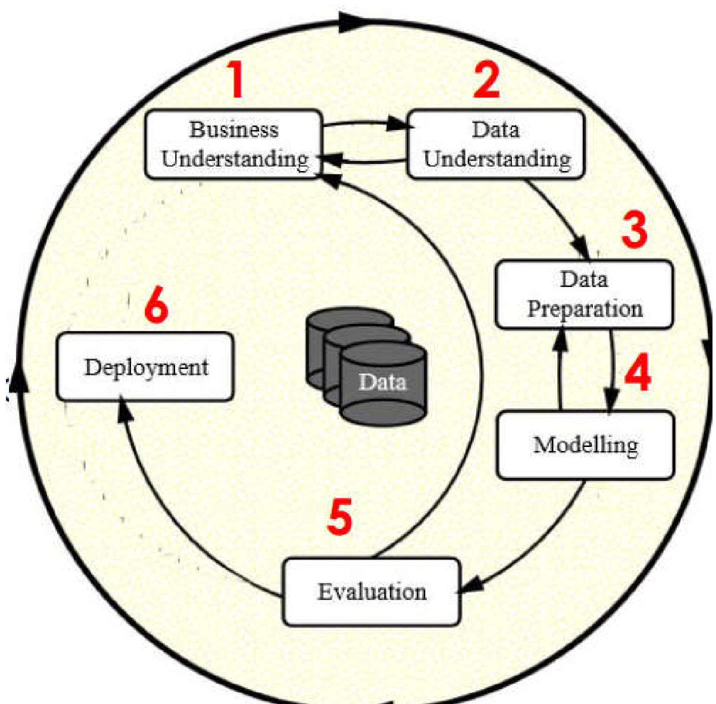
Data Mining Course

Data Cleaning

Prof. Chiraz Ben Abdelkader

September 26-27, 2019

Data Mining Pipeline



Step 4 – Data Preparation

- Data cleaning today
- Feature selection and engineering Next week

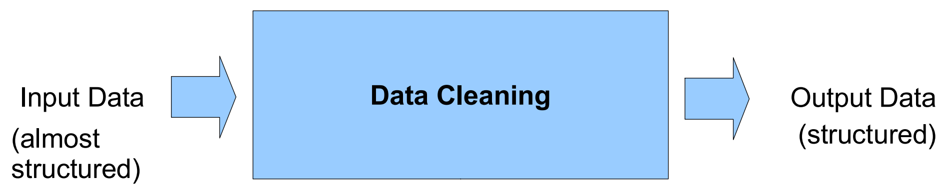
- This is one of the most important and time-consuming parts of the data mining pipeline.
- Why? Because:
  - Better data beats better models
  - Garbage-In Garbage-Out

Goal of Data Cleaning

- Raw data is almost never "clean", in ideal form for modeling
- Diagnose your data for noise/problems
- Reduce noise; improve quality of your data

Structured Data

- We assume the input data is **structured** or **almost structured**: tabular data where possibly some columns do not have the correct data type (numeric or categorical)
- One of the goals of data cleaning is to make sure the output data is structured.



Common Data Problems

Bad observations (rows)

- Duplicate observations
- Irrelevant observations

Common Data Problems

Bad attributes (columns)

- Wrong or confusing data type
- Redundant: represent same information
- Useless for data mining
- Contains too many missing values

Common Data Problems

Bad data values

- Missing values
- Extreme or unusual values (called *outliers*)
  - Very different than majority of data values
- Invalid or absurd values
  - With respect to the attribute's definition
  - Caused by data entry or measurement errors
- Inconsistent category names
  - for example: 'skateboard' and 'board' in the Travel_to_school variable from last week's data

Example 1

	name	sex	treatment a	treatment b
0	Daniel	male	-	42
1	John	male	12	31
2	Jane	female	24	27

What kind of problem does this data contain?

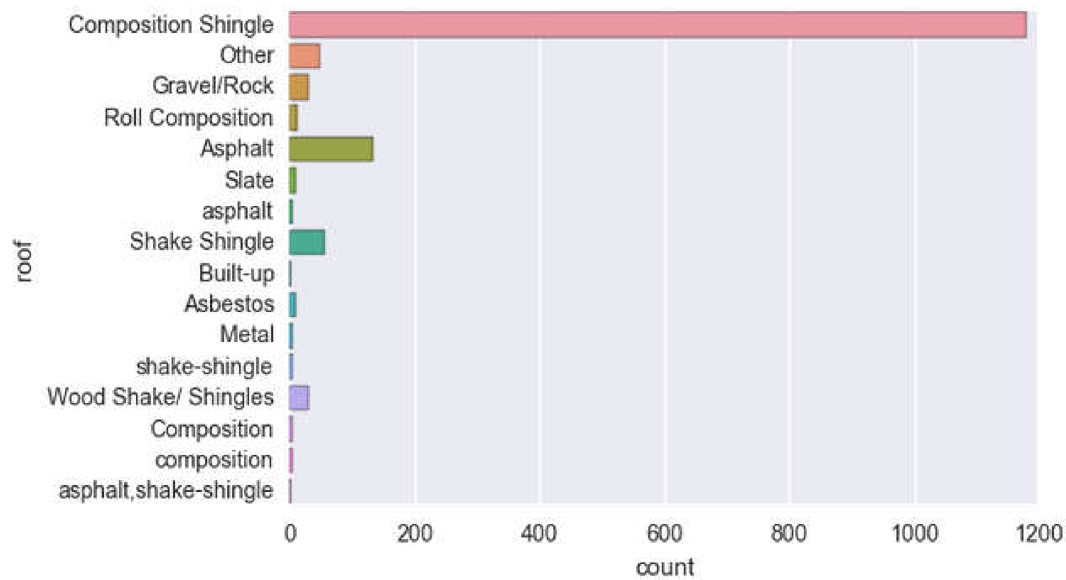
Example 1 (cont.)

	name	sex	treatment a	treatment b
0	Daniel	male	-	42
1	John	male	12	31
2	Jane	female	24	27

```
In [1]: print(df.dtypes)
name            object
sex             object
treatment a     object
treatment b      int64
dtype: object
```

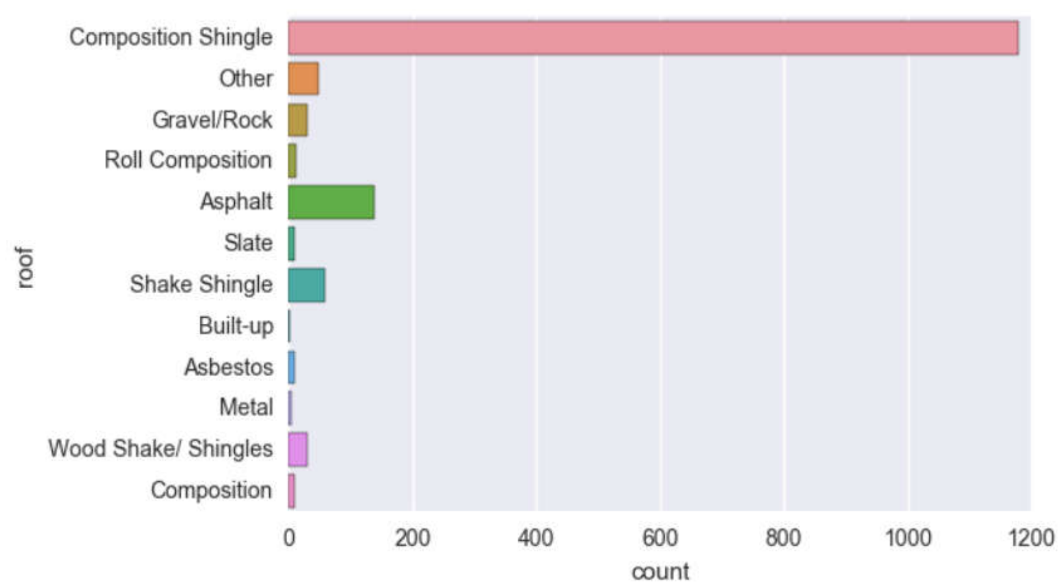
Python reads the 3rd column as string instead of numeric

Example



What kind of problem does this data contain?

Example (cont.)



After fixing the problem ...

About Outliers

- Outliers may or may not be legitimate (valid) values
- They are problematic for many predictive modeling methods (lead to low quality models).
  - For example, linear regression models are more sensitive to outliers than decision trees.
- However, outliers are useful in anomaly detection

Missing Values

- Missing values can be represented explicitly or implicitly in the data
  - Explicitly: the value is absent (no value)
  - Implicitly: a special value is used to indicate the missing value
    - Numeric data: for example use 0 or -1
    - Categorical data: for example use "unknown" or "N/A" or "other"

How to Detect Data Problems?

- Read description of data (documentation)
- Data exploration and visualization

Typical Data Cleaning Tasks

- Remove duplicate observations (rows)

- Data type conversions of variables
- Remove useless variables
- Remove redundant variables

- Handling missing values
- Handling outliers and unusual values
- Fixing inconsistent category names

Data Type Conversions

- Convert to the right data type

- Common conversions:
  - Integer to float
  - String to numeric
  - Numeric to categorical
  - String to categorical

Data Type Conversions

- Useful Python functions for data type conversions:
  - **astype** method in Pandas
  - **to_numeric** function in Pandas
  - Regular expressions (**re** module)

String to Numeric

	name	sex	treatment a	treatment b
0	Daniel	male	-	42
1	John	male	12	31
2	Jane	female	24	27

```
In [1]: print(df.dtypes)
name            object
sex             object
treatment a     object
treatment b      int64
dtype: object
```

```
In [5]: df['treatment a'] = pd.to_numeric(df['treatment a'],
   ...:                                   errors='coerce')

In [6]: df.dtypes
Out[6]:
name              object
sex             category
treatment a      float64
treatment b       object
dtype: object
```

String to Numeric

- The **to_numeric** function will fail if the input value contains non-numeric characters

	Job #	Doc #	Borough	Initial Cost	Total Est. Fee
0	121577873	2	MANHATTAN	$75000.00	$986.00
1	520129502	1	STATEN ISLAND	$0.00	$1144.00
2	121601560	1	MANHATTAN	$30000.00	$522.50
3	121601203	1	MANHATTAN	$1500.00	$225.00
4	121601338	1	MANHATTAN	$19500.00	$389.50

String to Numeric

- In this case, we need to use string manipulation functions and **regular expressions**

Example text		**Regular expression**
$17	$12345678901	\$\d*
$17.00	$12345678901.42	\$\d* \.\d*
$17.89	$12345678901.24	\$\d*\.\d{2}

Converting to Categorical

- Categorical variables in Python are represented either as ***str*** or as ***category***
- The *category* data type is more efficient in terms of memory
  - It is like enumeration type in C/C++

Dealing with bad values

- 3 main approaches for dealing with all types of bad values (missing, invalid, outliers):
  1) Impute value, i.e. replace with a "proper" value
     - For numerical attributes, usually use *mean* or *median*
     - For categorical attribute, use mode (most frequent value)
  2) Remove entire observation (row)
  3) Keep the missing value.
     - For categorical attributes, just create a special category for missing values
     - For numerical attributes, this is only possible with certain models that tolerate missing values, e.g. decision trees.
       - but this is not the case for linear regression for example.

Dealing with bad values

- Useful Pandas methods:
  - notna, notnull
  - dropna
  - fillna

Summary

- Data cleaning approaches are heuristic, subjective, and depend on <u>your</u> data
  - **Heuristic**: hopefully will reduce noise but do not guarantee eliminating it
  - **Subjective**: not clear which approach is the best – just use your common sense and intuition
  - **Depend on your data**: there is no absolute perfect approach for all data; the best approach always depends on your data
    - should experiment with different approaches until you get it right ...

Reference

- https://elitedatascience.com/data-cleaning

Examples from TP2

Converting to Categorical

- ***seqno*** variable

```
In [4]:  1  df.head()
Out[4]:
```

	idate	imonth	iday	iyear	dispcode	seqno	ladult	numadult	nummen	numwor
0	4302013	4	30	2013	1100	2013009711	NaN	1.0	0.0	
1	4242013	4	24	2013	1100	2013003472	NaN	1.0	1.0	
2	10232013	10	23	2013	1100	2013006428	NaN	1.0	0.0	
3	1192013	1	19	2013	1100	2013000091	NaN	1.0	0.0	
4	12052013	12	5	2013	1100	2013004518	NaN	1.0	0.0	

5 rows × 91 columns

```
In [22]:  1  df.seqno.dtypes
Out[22]: dtype('int64')
```

```
In [11]: df.seqno = df.seqno.astype(str)

In [12]: df.seqno.head()

Out[12]: 0    2013009711
         1    2013003472
         2    2013006428
         3    2013000091
         4    2013004518
         Name: seqno, dtype: object
```


```
In [11]: df.seqno = df.seqno.astype(str)

In [12]: df.seqno.head()

Out[12]: 0    2013009711
         1    2013003472
         2    2013006428
         3    2013000091
         4    2013004518
         Name: seqno, dtype: object
```


```
In [13]: df.seqno = df.seqno.astype('category')

In [14]: df.seqno.head()

Out[14]: 0    2013009711
         1    2013003472
         2    2013006428
         3    2013000091
         4    2013004518
         Name: seqno, dtype: category
```

Implicit Missing Values

- ***medcost*** variable

medcost: Could Not See Dr. Because Of Cost

Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?

Value	Value Label	Frequency	Percent
1	Yes	60,104	12.22
2	No	430,446	87.53
NA	Don't know/Not sure	933	0.19
NA	Refused	290	0.06
Total	491,773	100.00	

Variable type: categorical Missing values: 7, 9

```
In [15]: df.medcost.unique()
Out[15]: array([2, 1, 7, 9], dtype=int64)

In [16]: df.medcost.isnull().sum()
Out[16]: 0

In [17]: df.medcost.value_counts()
Out[17]: 2    87629
         1    12151
         7      161
         9       59
         Name: medcost, dtype: int64

In [20]: df.loc[(df.medcost==7)|(df.medcost==9),'medcost'] = np.nan

In [21]: df.medcost.isnull().sum()
Out[21]: 220
```

Confusing Data Type

- ***weight2*** variable

weight2: Reported Weight In Pounds

About how much do you weigh without shoes? (If respondent answers in metrics, put a 9 in the first column)[Round fractions up.]

Value	Value Label	Frequency	Percent	Cum
[50 - 0999]	Weight (pounds)	470,161	95.61	
[9000 - 9998]	Weight (kilograms)	918	0.19	
NA	Don't know/Not sure	6,746	1.37	
NA	Refused	12,760	2.59	
NA	[Missing]	1,188	0.24	
Total	491,773	100.00		

Variable type: continuous Missing values: 7777, 9999, BLANK Notes: Numbers from 9,000 to 9,9998 denote kilograms. Values be denote pounds. Refer to ** for another version of this question.

Confusing Data Type

In [18]:
```
df.weight2.describe()
```

Out[18]:
```
count     99760.000000
mean        563.349830
std        1844.279295
min          50.000000
25%         145.000000
50%         174.000000
75%         205.000000
max        9999.000000
Name: weight2, dtype: float64
```

In [19]:
```
idx1 = (df.weight2>=50)&(df.weight2<1000)
idx1.sum()
```

Out[19]: 95498

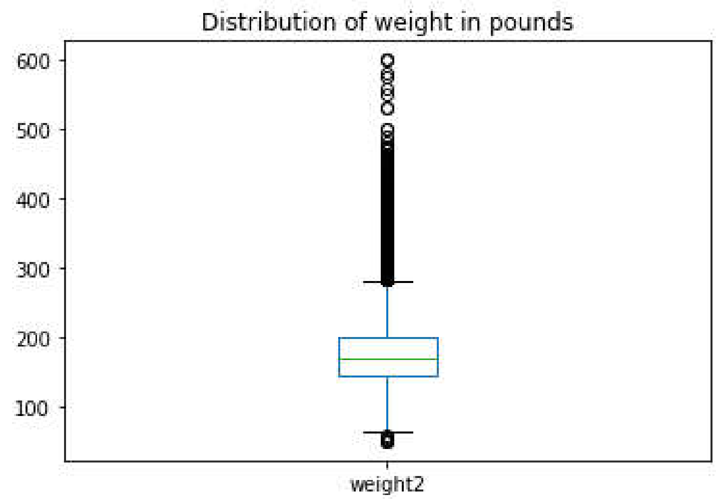
In [20]:
```
idx2 = (df.weight2>=9000)&(df.weight2<9999)
idx2.sum()
```

Out[20]: 178

In [25]:

```
df.loc[idx1,'weight2'].plot.box()
plt.title('Distribution of weight in pounds')
```

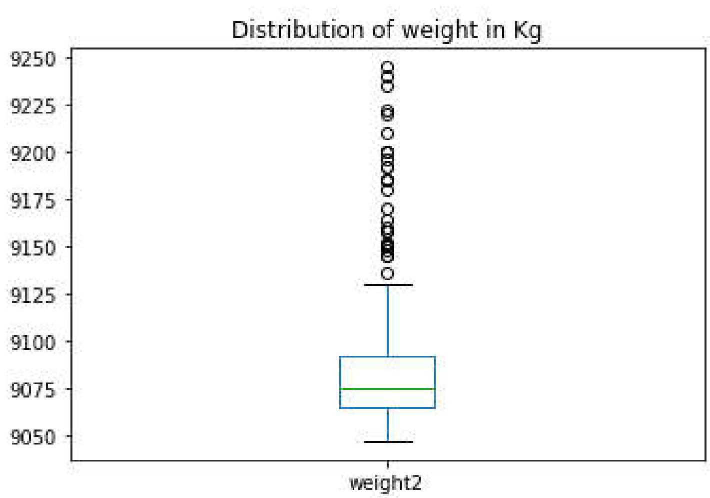
Out[25]: Text(0.5,1,'Distribution of weight in pounds')



In [24]:

```
df.loc[idx2,'weight2'].plot.box()
plt.title('Distribution of weight in Kg')
```

Out[24]: Text(0.5,1,'Distribution of weight in Kg')



Confusing Data Type

- Examples from TP2

alcday5: Days In Past 30 Had Alcoholic Beverage

During the past 30 days, how many days per week or per month did you have at least one drink of any alcoholic beverage beverage or liquor?

Value	Value Label	Frequency	Percent
[101 - 199]	Days per week	63,144	12.84
[201 - 299]	Days in the past 30 days	172,368	35.05
NA	Don't know/Not sure	3,192	0.65
0	No drinks in past 30 days	236,617	48.12
NA	Refused	3,393	0.69
NA	[Missing]	13,059	2.66
Total	491,773	100.00	

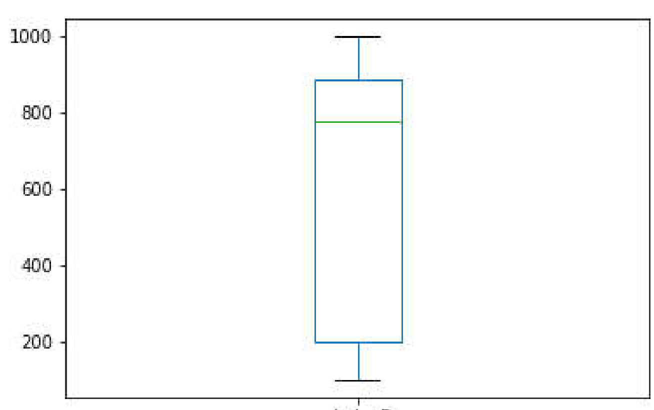
Variable type: continuous Missing values: 777, 999, BLANK Recoded to zero: 888 Notes: The first digit denotes days per v
(2). The remaining digits indicate the count of days. Refer to *drnkany5*, *drocdy3_*, and *_drnkdy4* for other versions of this q

```
In [28]:  1  df.alcday5.describe()

Out[28]: count    97351.000000
         mean       538.910848
         std        355.833731
         min        101.000000
         25%        202.000000
         50%        777.000000
         75%        888.000000
         max        999.000000
         Name: alcday5, dtype: float64
```

```
In [29]:  1  df.alcday5.plot.box()

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x225503a2a90>
```



In [30]:
```
idx1 = (df.alcday5 >=101)&(df.alcday5<200)
idx1.sum()
```
Out[30]: 12843

In [31]:
```
idx2 = (df.alcday5 >=201)&(df.alcday5<300)
idx2.sum()
```
Out[31]: 35152

In [35]:
```
# Number of EXPLICIT missing values

df.alcday5.isnull().sum()
```
Out[35]: 2649

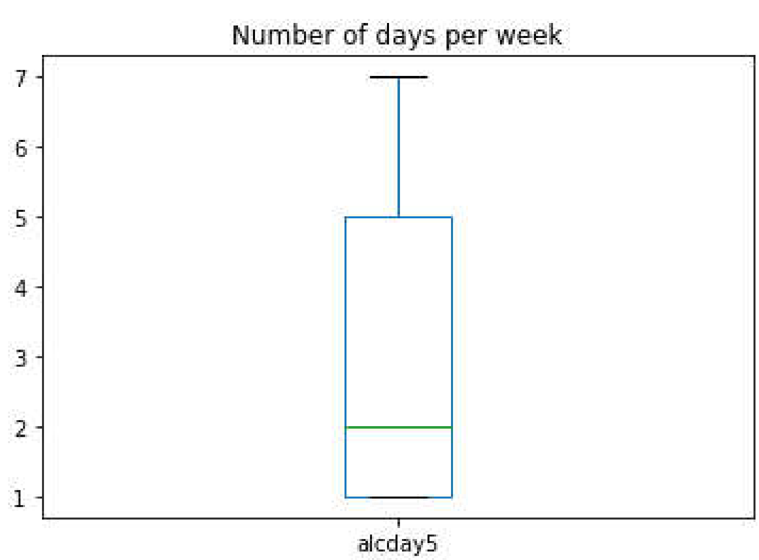
In [34]:
```
# Number of IMPLICIT missing values

idx3 = (df.alcday5==777)|(df.alcday5==999)
idx3.sum()
```
Out[34]: 1387

In [37]:
```
# Number of special endcoded ZERO values

(df.alcday5==888).sum()
```
Out[37]: 47969

In [39]:
```
(df.loc[idx1,'alcday5']-100).plot.box()
plt.title('Number of days per week')
```
Out[39]: Text(0.5,1,'Number of days per week')



In [40]:

```
(df.loc[idx2,'alcday5']-200).plot.box()
plt.title('Number of days per month')
```

Out[40]: `Text(0.5,1,'Number of days per month')`