Data Cleaning

Matthew Edwards

14/09/2021

Motivation

The most time consuming stages of a data science project are data cleaning and exporatory data analysis (EDA). It is often stated that approximately 80% of the time is consumed by these two stages. This percentage will increase as your skill level increases, since these stages are less effected by skill level. These two stages are particularly important for:

- Reliability
- Robustness
- Accuracy

The main difference between good and bad data scientists (with equal skill levels) is the quality of the data cleaning and EDA stages.

Introduction

Clean data are data that can be used for exploratory data analysis (EDA). **Data cleaning** is the process of transforming raw data into clean data. The process of data cleaning involves five steps:

1. Column names
2. Missing values
3. Column types
4. Deduplicstion
5. Quality checking

Raw Data

```python
import pandas as pd
data = pd.read_csv("data.csv")
print(data)
```

```
##          A           B   C  D
## 0     male  18/06/1988  74  C
## 1   female  29/08/1993  92  A
## 2   female  29/08/1993  92  A
## 3   female  18/03/1997  -1  B
## 4       na  16/01/2001  63  A
```

Column Names

The first stage of data cleaning is renaming the column names. This will simplify column selection. Column names should not include spaces or special characters. The three most popular column name formats are:

- `snake_case`
- `kebab-case`
- `PascalCase`

`snake_case` is recommended since it is more human readable.

Column Names

```python
names = {
  'A': 'gender',
  'B': 'dob',
  'C': 'score',
  'D': 'grade',
}
data.rename(mapper=names, axis=1, inplace=True)
print(data.columns)
```

```
## Index(['gender', 'dob', 'score', 'grade'], dtype='object
```

Missing Values

Missingness is often represented inconsistently. For example the integer `-1` and the string “na” are commonly used to represent missing numeric and categorical values, respectively.

Missingness is also sometimes respresented with an empty string. This can introduce ambiguity when empty strings are used for other purposes. In `pandas` missing values should be represented with the `numpy` special value `nan` (there are some rare exceptions).

Missing Values

```
import numpy as np
missing = {'na': np.nan, -1: np.nan}
data.replace(missing, inplace=True)
print(data.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 5 entries, 0 to 4
## Data columns (total 4 columns):
##  #   Column  Non-Null Count  Dtype
## ---  ------  --------------  -----
##  0   gender  4 non-null      object
##  1   dob     5 non-null      object
##  2   score   4 non-null      float64
##  3   grade   5 non-null      object
## dtypes: float64(1), object(3)
## memory usage: 288.0+ bytes
## None
```

Column Types

The most common `pandas` data types are:

- `int64`
- `float64`
- `datatime64`
- `category`

The use of `boolean` and `string` data types should be avoided. `category` data types are preferred instead since they include information about categories and ordering.

Column Types

```
from pandas.api.types import CategoricalDtype
gender_cat = CategoricalDtype(["male", "female"], ordered=F
grade_cat = CategoricalDtype(["A", "B", "C", "D"], ordered=
types = {
  'gender': gender_cat,
  'dob': 'datetime64',
  'score': 'float64',
  'grade': grade_cat,
}
data = data.astype(types)
print(data.dtypes)
```

```
## gender          category
## dob       datetime64[ns]
## score            float64
## grade           category
## dtype: object
```

Deduplication

Deduplication is the process of removing identical rows from a table. Two rows are identical if the corresponding values in each column are equal.

Deduplication

```
data.drop_duplicates(inplace=True)
print(data)
```

```
##    gender        dob  score grade
## 0    male 1988-06-18   74.0     C
## 1  female 1993-08-29   92.0     A
## 3  female 1997-03-18    NaN     B
## 4     NaN 2001-01-16   63.0     A
```

Quality Check

Quality checking is the process of testing if the columns and relationships between columns satisfy certain expectations. For example testing if all values in a column are non-negative or between two bounds.

Quality checking improves the reliability of project outputs (e.g. predictive model) and robustness to upstream changes in the data pipeline.

Quality Check

```python
def check(row):
  A = row['grade'] == "A" and 90 < row['score'] <= 100
  B = row['grade'] == "B" and 80 < row['score'] <= 90
  C = row['grade'] == "C" and 70 < row['score'] <= 80
  D = row['grade'] == "D" and  0 < row['score'] <= 70
  return A or B or C or D or np.isnan(row['score'])

print(data.apply(check, axis=1))
```

```
## 0     True
## 1     True
## 3     True
## 4    False
## dtype: bool
```

Advice

You should think of data cleaning as transforming data **without loss of information**. Data contains important information that you should not lose at this stage. In fact, data cleaning should increase the information content of data (e.g. column names and types). With only some exceptions:

1. **Data Cleaning**: lossless data transformations
2. **Modelling**: lossy data transformations

Lossy data transformations during the modelling stage (i.e. pre-processing) are evaluated with validation data on the task.