LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT
UNIVERSIDADE DE LISBOA

Carlos J. Costa

DATA PREPARATION

Missing Values

- transform object variables to symbolic
- keep the numeric separated from the symbolic ones, to deal with them with the right tools.

Missing Values

- In order to fill null values in a datasets, use one of the following function :
  - fillna(),
  - replace()
  - interpolate()
- these function replace NaN values with some value of their own

Missing Values

```
# importing pandas as pd and numpy as np
import pandas as pd
import numpy as np

# dictionary of lists
dict = {'First Quiz':[100, 90, np.nan, 95],
        'Second Quiz': [60, 60, 75, np.nan],
        'Third Quiz':[np.nan, 50, 50, 50]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# filling missing value using fillna()
df.fillna(0)
```

Missing Values

- filling a missing value with previous ones

```
df.fillna(method ='pad')
```

- Filling null value with the next ones

```
df.fillna(method ='bfill')
```

- Fill all the null values in Gender column with “No Gender”

```
df["Gender"].fillna("No Gender", inplace = True)
```

Missing Values

- Fill all the null values with the mean

  df.fillna(df.mean(),inplace=True)

- Now we are going to replace the all Nan value in the data frame with -99 value.

  df.replace(to_replace = np.nan, value = -99)

- Using interpolate() function to fill the missing values using linear method.

  df.interpolate(method ='linear', limit_direction ='forward')

Missing Values

- **Dropping missing values using**
- **dropna() function drops null values from a dataframe**
- **this function drop Rows/Columns of datasets with Null values in different ways.**

```
df.dropna()
```

- **Dropping rows if all values in that row are missing**

```
df.dropna(how = 'all')
```

- **Dropping columns with at least 1 null value.**

```
df.dropna(axis = 1)
```

Missing Values

```
# importing pandas as pd and numpy as np
import pandas as pd
import numpy as np

# dictionary of lists
dict = {'First_Quiz':[100, 90, np.nan, 95],
        'Second_Quiz': [60, 60, 75, np.nan],
        'Third_Quiz':[np.nan, 50, 50, 50]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)


df

df.First_Quiz.fillna(df.First_Quiz.mean(),inplace=True)
```

Normalization

- Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0.
- It is generally useful for classification algorithms.
- It can only be applied to numerical data, without any missing value.

Normalization

```
dataNorm=((df-df.min())/(df.max()-df.min()))*10
```

Dummification

- The purpose of Dummification is creating dummy variables.

Dummification

```
# importing pandas as pd and numpy as np
import pandas as pd
import numpy as np

# dictionary of lists
dict = {'Sex':["f", "f", "m", "m"],
        'First Quiz':[100, 90, 55, 95],
        'Second Quiz': [60, 60, 75, 99],
        'Third Quiz':[40, 50, 50, 50]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)
```

Data balancing

- Data imbalance usually reflects an unequal distribution of classes within a dataset.
- For example, in a credit card fraud detection dataset, most of the credit card transactions are not fraud and a very few classes are fraud transactions.
- In the case of binary classification;
  - **positive** to the minority class
  - **negative** to the majority one.

Data balancing

- 2 different strategies:
  - undersampling
  - oversampling.
- with a huge dataset, and consequently a considerable number of positive records:
  - **Undersampling** strategy, keeping the positive records and sampling the negative ones to balance the final distribution

Data balancing

- in the presence of a small number of positive records:
  - oversampling, in order to create a larger set to support the training step.
- Among the different oversampling techniques, SMOTE is one of the most interesting ones.
- https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/