Lisbon School
of Economics
& Management
Universidade de Lisboa

Carlos J. Costa

DATA PREPARATION

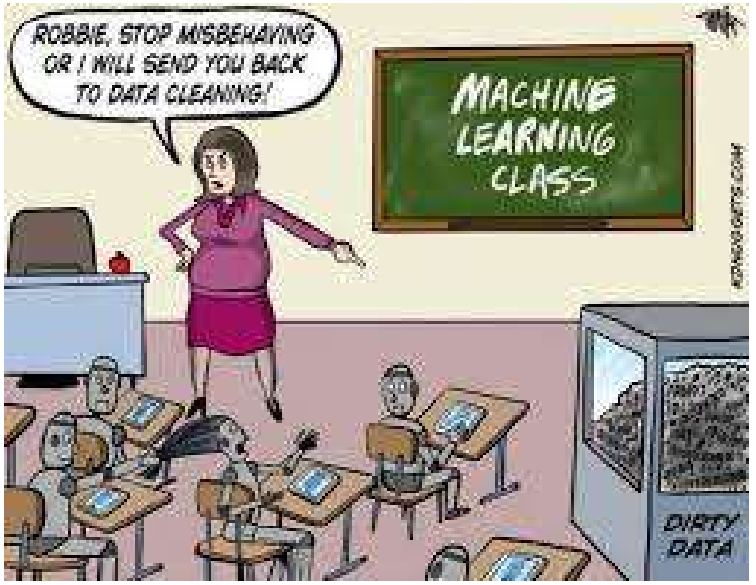
(2021)

Data preparation

- Missing values
- Normalization and Standardization
- Dummification
- Data balancing

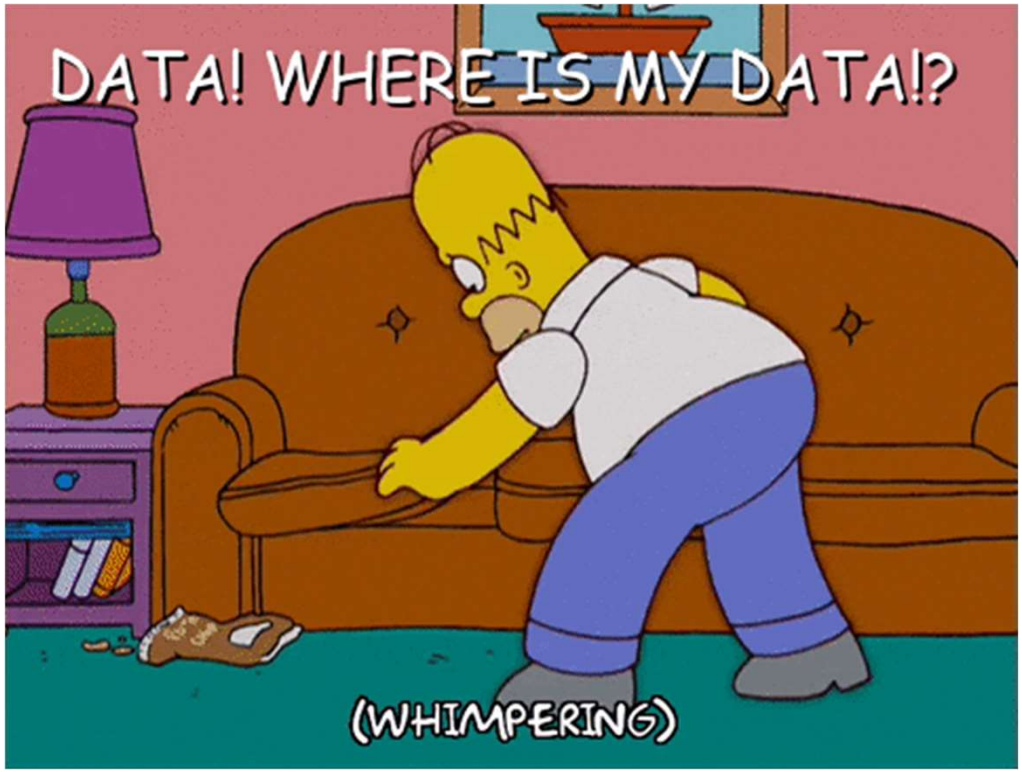
Data Preparation

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



Missing Values

```
fillna(),
replace()
interpolate()
```



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

```
df.fillna(method ='pad’)

df.fillna(method ='bfill’)

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

Missing Values

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

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

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

Missing Values

- `df.dropna()`
- `df.dropna(how = 'all')`
- `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 and Standardization

- **Normalization**

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Standardization**

$$X_{changed} = \frac{X - \mu}{\sigma}$$

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)
```

Dummification

- Our initial dataset df

- Function `get_dummies`

```
df
```

	Sex	First Quiz	Second Quiz	Third Quiz
0	f	100	60	40
1	f	90	60	50
2	m	55	75	50
3	m	95	99	50

```
df['f'],df['m']=pd.get_dummies(df['Sex'])
```

```
df
```

	Sex	First Quiz	Second Quiz	Third Quiz	f	m
0	f	100	60	40	f	m
1	f	90	60	50	f	m
2	m	55	75	50	f	m
3	m	95	99	50	f	m

Dummification

- Function get_dummies

```
gender=pd.get_dummies(df['Sex'])
gender
```

	f	m
0	1	0
1	1	0
2	0	1
3	0	1

- Join two dataframes

```
df=df.join(gender)
```

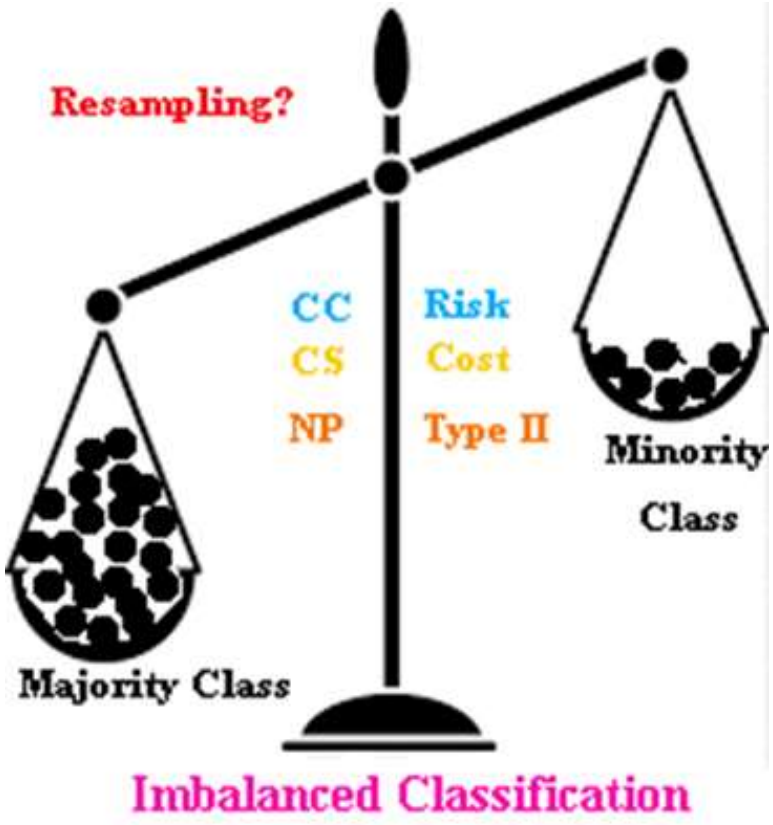
```
df
```

	Sex	First Quiz	Second Quiz	Third Quiz	f	m
0	f	100	60	40	1	0
1	f	90	60	50	1	0
2	m	55	75	50	0	1
3	m	95	99	50	0	1

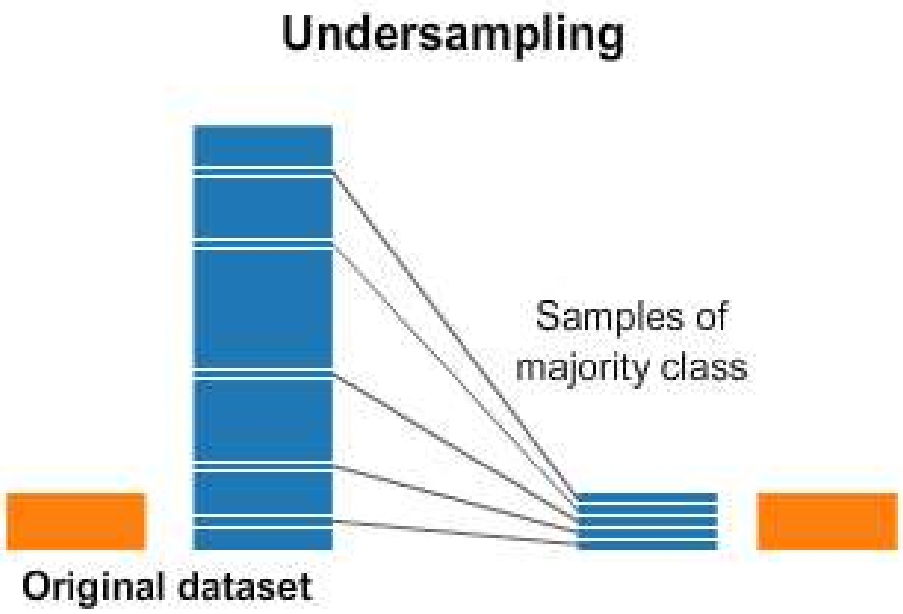
Data balancing

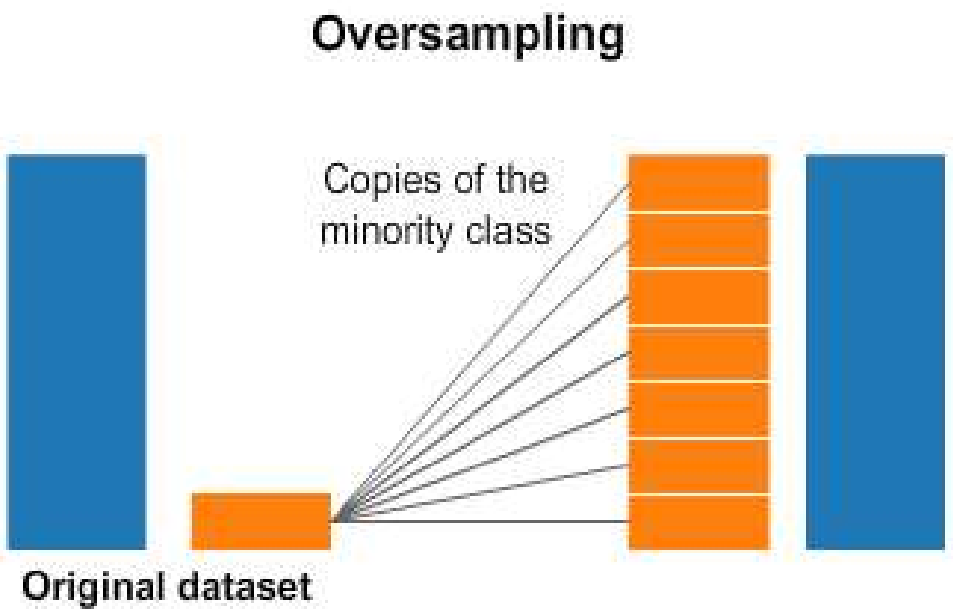
- Unequal distribution of classes within a dataset.
- Example: Credit card fraud

Data balancing

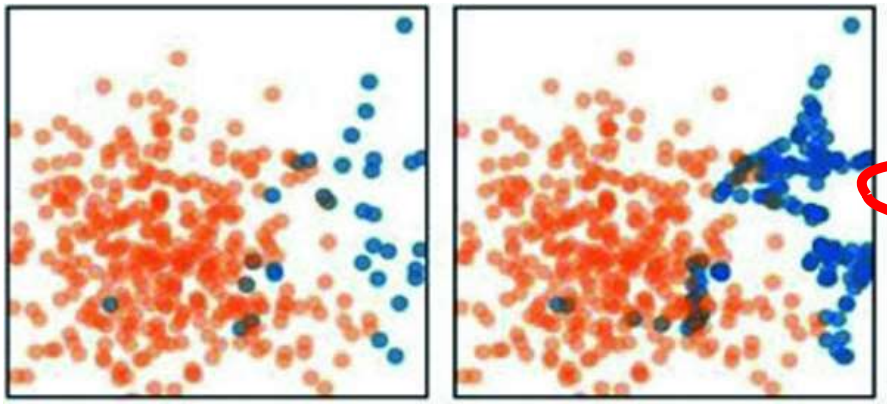


Data balancing





Data balancing



Original dataset Applying SMOTE

```
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import sklearn.metrics as metrics
import pandas as pd
```

```
# preprocessing, including standrdization or normalization
df=pd.read_csv('https://raw.githubusercontent.com/masterfloss/data/main/jogadores.csv', sep=";")
y=df['Ser Transferido']
X=df[['Idade','Altura','Minutos','Valor de Mercado']]
```

```
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

```
scaler = StandardScaler()
X_Trans=scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_Trans, y, test_size=0.20, random_state=42)
```

```
# object creation and model fit
model=KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred=model.predict(X_test)
metrics.accuracy_score(y_test, y_pred)
```

```
0.5641025641025641
```

```
sum(y)/len(y)
```

Data balancing

```
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import sklearn.metrics as metrics
import pandas as pd
```

```
# preprocessing, including standrdization or normalization
df=pd.read_csv('https://raw.githubusercontent.com/masterfloss/data/main/jogadores.csv', sep=";")
y=df['Ser Transferido']
X=df[['Idade','Altura','Minutos','Valor de Mercado']]
```

```
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

```
scaler = StandardScaler()
X_Trans=scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_Trans, y, test_size=0.20, random_state=42)
```

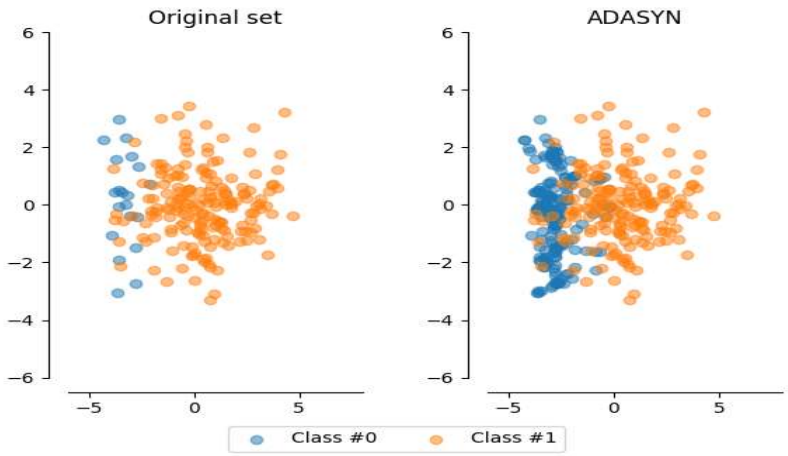
```
# object creation and model fit
model=KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred=model.predict(X_test)
metrics.accuracy_score(y_test, y_pred)
```

```
0.5641025641025641
```

```
sum(y)/len(y)
```



Summary

- Missing values
- Normalization and Standardization
- Dummification
- Data balancing