Preprocessing 101

Outline

1. Golden Rules
2. Everything should be numbers
   - Categorical variable
   - Date
3. NaN ?
4. Feature Engineering
5. One concrete example

Golden Rules

Rule 1

Don't reinvent the wheel

1. Always, always, **always** google first
2. StackOverflow has the best answers

> "You'll be lucky if you have a single original idea in your entire life" (my dad)

Rule 2

NEVER loop over your data in Numpy / Pandas

```
country_empty=[]
for i in range(0,len(train)):
    if(train['COUNTRY'][i] is np.nan):
        country_empty.append(train['FISRT_APP_COUNTRY'][i])
```

:-(

```
train['FISRT_APP_COUNTRY'][train['COUNTRY'].isnull()]
```

:-)

Codebase

```
import numpy as np
import pandas as pd
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
target = train['TARGET_COLUMN']
train.drop('TARGET_COLUMN', axis=1, inplace=True)
piv_train = train.shape[0]

# Creating a DataFrame with train+test data
df_all = pd.concat((train, test), axis=0, ignore_index=True)
```

Everything should be numbers

Categorical variable

Gotta catch 'em all !

```
numerics = ['int16', 'int32', 'int64',
            'float16', 'float32', 'float64']
non_numeric_columns = df_all.select_dtypes(exclude=numerics).columns
```

Now you can do different things ...

---

1. Label Encoding

column_before	column_after
foo	0
bar	1
baz	2
foo	0

```
from sklearn.preprocessing import LabelEncoder
df_all[non_numeric_columns] = df_all[non_numeric_columns]
                                .apply(LabelEncoder().fit_transform)
```

---

2. One Hot Encoding

column_before	foo	bar	baz
foo	1	0	0
bar	0	1	0
baz	0	0	1
foo	1	0	0

```
ohe_columns = non_numeric_columns
dummies = pd.concat(
    [pd.get_dummies(df[col], prefix=col) for col in ohe_columns],
    axis=1)
df_all.drop(ohe_columns, axis=1, inplace=True)
df_all = pd.concat((df_all, dummies), axis=1)
```

Date != Categorical variable

You can extract Day/Month/Year/Whatever ...

```
col = 'some_date_column'
# Let's assume this column has the format : 01/2016
regex = r'([\d]{2})/([\d]{4})'
df_all[[col + '_month', col + '_year']] = df_all[col].str.extract(regex)
df_all.drop(col, axis=1, inplace=True)
```

... Or transform it to timestamp !

```
import datetime
import calendar
def to_timestamp(date_string):
    date_format = "%m/%Y"
    if date_string is np.nan:
        return None
    to_datetime = datetime.datetime.strptime(date_string, date_format)
    return calendar.timegm(to_datetime.utctimetuple())
df_all[col] = df_all[col].apply(to_timestamp)
```

NaN ?

Data might not be a number (yet)... Checkout:

```
df_all.isnull().sum()
```

What can I do ?

Imputing

Replace each missing data in a column by the column's mean, median or most frequent term...

```
from sklearn.preprocessing import Imputer
my_strategy = 'mean' # or 'median' or 'most_frequent'
imp = Imputer(strategy=my_strategy)
df_all = pd.DataFrame(imp.fit_transform(df_all),
                      columns=df_all.columns,
                      index=df_all.index)
```

Fillna()

... Or fill with the last valid term

```
# Checkout other method for fillna() ;-)
df_all.fillna(method='ffill', inplace=True)
```

Tips

1. Be smart! Don't use only one method to replace NaNs...
2. Be careful. Label Encoding and One Hot Encoding can somehow 'remove' your NaN values.

Feature Engineering

There is no method here, just feelings. Plot your data for each feature to give you an intuition, find correlations, ...

Think about the problem at hand and be creative !

Scale your data

Some algorithms are dumb, help them.

1. Standardization = zero mean + unit variance

   `sklearn.preprocessing.StandardScaler`

2. Transform your data: Sometimes a good logarithmic transformation is all you need

   `sklearn.preprocessing.FunctionTransformer`

Create new features

1. Create a flag (0/1)

   Example: a company has 500+ employees

2. Create new categories

   Example: indicate the season of a date feature

[...]

Bonus: Preprocessing for my first soumission

Don't go down if you don't want to get spoiled

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, Imputer

# Import
test = pd.read_csv('test.csv', sep=';', na_values='(MISSING)')
train = pd.read_csv('train.csv', sep=';', na_values='(MISSING)')
target = train['VARIABLE_CIBLE']
train.drop('VARIABLE_CIBLE', axis=1, inplace=True)
piv_train = train.shape[0]

# Creating a DataFrame with train+test data
df_all = pd.concat((train, test), axis=0, ignore_index=True)
```

```
#################
# Preprocessing
#################
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
date_columns = ['PRIORITY_MONTH', 'FILING_MONTH', 'PUBLICATION_MONTH', 'BEGIN_MONTH']
non_numeric_columns = train.select_dtypes(exclude=numerics).columns.difference(date_columns)

# Label Encoding
df_all[non_numeric_columns] = df_all[non_numeric_columns].apply(LabelEncoder().fit_transform

# Just get the year
regex = r'/([\d]{4})'
for col in date_columns:
    df_all[col] = df_all[col].str.extract(regex)


# Replacing NaN values with mean
all_data = Imputer().fit_transform(df_all)

# Recreate train / test
train, test = all_data[:piv_train], all_data[piv_train:]

# Format target
target = target.replace(to_replace=['GRANTED', 'NOT GRANTED'], value=[1., 0.])
```