1 Removing duplicate data in NumPy and Pandas

Both NumPy and Pandas offer easy ways of removing duplicate rows. Pandas offers a more powerful approach if you wish to remove rows that are partly duplicated.

1.1 NumPy

With numpy we use *np.unique()* to remove duplicate rows or columns (use the argument =axis=0 for unique rows or *axis=1* for unique columns).

```
import numpy as np

array = np.array([[1,2,3,4],
                  [1,2,3,4],
                  [5,6,7,8],
                  [1,2,3,4],
                  [3,3,3,3],
                  [5,6,7,8]])

unique = np.unique(array, axis=0)
print (unique)

OUT:

[[1 2 3 4]
 [3 3 3 3]
 [5 6 7 8]]
```

We can return the index values of the kept rows with the argument *return_index=True* (the argument *return_inverse=True* would return the discarded rows):

```
array = np.array([[1,2,3,4],
                  [1,2,3,4],
                  [5,6,7,8],
                  [1,2,3,4],
                  [3,3,3,3],
                  [5,6,7,8]])

unique, index = np.unique(array, axis=0, return_index=True)
print ('Unique rows:')
print (unique)
print ('\nIndex of kept rows:')
print (index)

OUT:

Unique rows:
[[1 2 3 4]
 [3 3 3 3]
 [5 6 7 8]]

Index of kept rows:
[0 4 2]
```

We can also count the number of times a row is repeated with the argument *return_counts=True*:

```
array = np.array([[1,2,3,4],
                  [1,2,3,4],
                  [5,6,7,8],
                  [1,2,3,4],
```

```
                  [3,3,3,3],
                  [5,6,7,8]])

unique, index, count = np.unique(array, axis=0,
                         return_index=True,
                         return_counts=True)

print ('Unique rows:')
print (unique)
print ('\nIndex of kept rows:')
print (index)
print ('\nCount of duplicate rows')
print (count)

OUT:

Unique rows:
[[1 2 3 4]
 [3 3 3 3]
 [5 6 7 8]]

Index of kept rows:
[0 4 2]

Count of duplicate rows
[3 1 2]
```

1.2 Pandas

With Pandas we use *drop_duplicates*.

```
import pandas as pd
df = pd.DataFrame()

names = ['Gandolf','Gimli','Frodo', 'Gimli', 'Gimli']
types = ['Wizard','Dwarf','Hobbit', 'Dwarf', 'Dwarf']
magic = [10, 1, 4, 1, 3]
aggression = [7, 10, 2, 10, 2]
stealth = [8, 2, 5, 2, 5]


df['names'] = names
df['type'] = types
df['magic_power'] = magic
df['aggression'] = aggression
df['stealth'] = stealth
```

Let's remove duplicated rows:

```
df_copy = df.copy() # we'll work on a copy of the dataframe
df_copy.drop_duplicates(inplace=True)
print (df_copy)

OUT:

df_copy = df.copy() # we'll work on a copy of the dataframe

df_copy.drop_duplicates(subset=['names','type'], inplace=True)
```

```
print (df_copy)
```

```
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7        8
1    Gimli   Dwarf            1          10        2
2    Frodo  Hobbit            4           2        5
```

We can choose to keep the last entered row with the argument *keep='last'*:

```
df_copy = df.copy() # we'll work on a copy of the dataframe
df_copy.drop_duplicates(subset=['names','type'], inplace=True, keep='last')
print (df_copy)
```

```
OUT:
```

```
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7        8
2    Frodo  Hobbit            4           2        5
4    Gimli   Dwarf            3           2        5
```

We can also remove all duplicate rows by using the argument *keep=False*:

```
df_copy = df.copy() # we'll work on a copy of the dataframe
df_copy.drop_duplicates(subset=['names','type'], inplace=True, keep=False)
print (df_copy)
```

```
OUT:
```

```
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7        8
2    Frodo  Hobbit            4           2        5
```

More complicated logic for choosing which record to keep would best be performed using a groupby method.