1 Working with ordinal and categorical data

Some data sets may have ordinal data, which are descriptions with a natural order, such as small, medium large. There may also be categorical data which has no obvious order like green, blue, red. We'll usually want to convert both of these into numbers for use by machine learning models.

Let's look at an example:

```
import pandas as pd

colour = ['green', 'green', 'red', 'blue', 'green', 'red','red']
size = ['small', 'small', 'large', 'medium', 'medium','x large', 'x small']

df = pd.DataFrame()
df['colour'] = colour
df['size'] = size

print (df)

OUT:

  colour     size
0  green    small
1  green    small
2    red    large
3   blue   medium
4  green   medium
5    red  x large
6    red  x small
```

1.1 Working with ordinal data

One of our columns is obviously ordinal data: size has a natural order to it. We can convert this text to a number by mapping a dictionary to the column. We will create a new column (size_number) which replaces the text with a number.

```
# Define mapping dictionary:

size_classes = {'x small': 1,
                'small': 2,
                'medium': 3,
                'large': 4,
                'x large': 5}

# Map to dataframe and put results in a new column:

df['size_number'] = df['size'].map(size_classes)

# Display th new dataframe:

print (df)

OUT:

  colour     size  size_number
0  green    small            2
1  green    small            2
2    red    large            4
3   blue   medium            3
```

```
4  green   medium            3
5    red  x large            5
6    red  x small            1
```

1.2 Working with categorical data

There is no obvious sensible mapping of colour to a number. So in this case we create an extra column for each colour and put a one in the relevant column. For this we use pandas *get_dummies method.*

```
colours_df = pd.get_dummies(df['colour'])

print (colours_df)

OUT:

   blue  green  red
0     0      1    0
1     0      1    0
2     0      0    1
3     1      0    0
4     0      1    0
5     0      0    1
6     0      0    1
```

We then combine the new dataframe with the original one, and we can delete the temporary one we made:

```
df = pd.concat([df, colours_df], axis=1, join='inner')

del colours_df

print (df)

OUT:

  colour     size  size_number  blue  green  red
0  green    small            2     0      1    0
1  green    small            2     0      1    0
2    red    large            4     0      0    1
3   blue   medium            3     1      0    0
4  green   medium            3     0      1    0
5    red  x large            5     0      0    1
6    red  x small            1     0      0    1
```

1.3 Selecting just our new columns

At the moment we have both the original data and the transformed data. For use in the model we would just keep the new columns. Here we'll use the pandas *loc* method to select column slices from size_number onwards:

```
df1 = (df.loc[:,'size_number':])

print (df1)

OUT:

   size_number  blue  green  red
0            2     0      1    0
1            2     0      1    0
```

```
2              4      0       0    1
3              3      1       0    0
4              3      0       1    0
5              5      0       0    1
6              1      0       0    1
```