RFormula

Lesson Objectives

- After completing this lesson, you should be able to:
  - Understand why `RFormula` exist
  - Understand `RFormula` notation
  - Understand the transformations `RFormula` produces on linear models
  - Use `RFormula` on a real world example

RFormula

- In modelling we often define a variable (criterion) as a function of other variables (predictors)

- The pseudocode for this (very close to more traditional math notation) could be:

```
Variable Y = variable A … AND variable Z
```

RFormula

But wait! The = symbol is not really appropriate because it's not representing the idea well. Left and right sides are not equal (we wish! Then we could do perfect predictions!). We want a symbol that represents the idea that '**Y is a function of A…Z**)'. If we were scribbling on paper, a good candidate for **such symbol could be '~'**

```
Variable ~ variable A … AND variable Z
```

RFormula: symbols

But wait! The AND symbol is not really appropriate because there are different ways variables A…Z can be combined to explain Y. One simple way could be just adding each variable

```
Variable ~ variable A + variable B … + variable Z
```

Note that this is not the vanilla ‘+’ symbol; in the context of a formula ‘+’ has a different meaning to standard ‘+’. It means ‘add A as a predictor in this model’

RFormula: ways variables can be combined

~	As a function of	y ~ a
+	Add the variable to the model	y ~ a + b + c
:	Interaction	y ~ a + b + c + a:b
*	Factor crossing	**y ~ a * b is interpreted as y ~ a + b + a:b.**

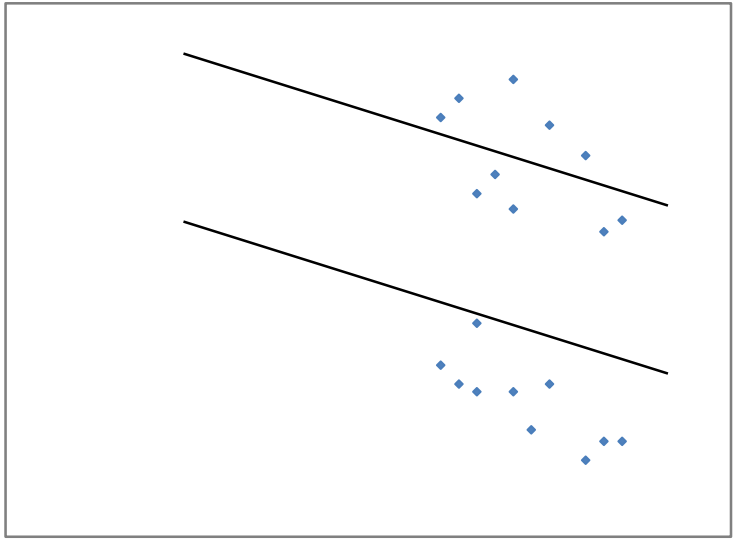
RFormula: internals

- An `RFormula` object produces a vector column of features and a double column of labels.

- String input columns will be one-hot encoded (remember lesson 3.2.2, categorical features), and numeric columns will be cast to doubles

RFormula: internals

1	40	1.5	Winter	**0**
2	100	1.6	Summer	**1**
3	60	1.3	Winter	**0**
4	120	1.4	Summer	**1**



Summer:
OH = 1

Winter:
OH = 0

RFormula, real data example

```
from pyspark.ml.feature import RFormula
formula = RFormula().setFormula(" ViolentCrimesPerPop ~ householdsize + racepctblack + racePctWhite ") \
                    .setFeaturesCol("features") \
                    .setLabelCol("label")
```

```
output = formula.fit(crimes).transform(crimes)
```

```
output.select("features", "label").show(3)
```

```
+----------------+-----+
|        features|label|
+----------------+-----+
| [0.33,0.02,0.9]|  0.2|
|[0.16,0.12,0.74]| 0.67|
|[0.42,0.49,0.56]| 0.43|
+----------------+-----+
only showing top 3 rows
```

RFormula: advantages

- You can see how `RFormula` is a very handy shorthand to write quite elaborated models.

- It started in the R world used only on linear models. With time, other models (example, random forests) started using formula notation to specify models.

- `RFormula` simplifies the creation of ML pipelines by providing a concise way of expressing complex feature transformations

RFormula: disadvantages

- Interactions are a concept from the classical multiple linear regression world, and they may produce weird results if you apply them on other models (particularly non-lineal models!)

- Connected to concepts from statistics (interactions, multiple linear regression) that are not well-known among engineers

Lesson Summary

- Having completed this lesson, you should be able to:
  - Understand the RFormula interface to model fitting
  - Fit a model using the formula interface