Chapter 8: Tree Methods

Ex2: Consulting Project - SOLUTION

You've been hired by a dog food company to try to predict why some batches of their dog food are spoiling much quicker than intended! Unfortunately this Dog Food company hasn't upgraded to the latest machinery, meaning that the amounts of the five preservative chemicals they are using can vary a lot, but which is the chemical that has the strongest effect? The dog food company first mixes up a batch of preservative that contains 4 different preservative chemicals (A,B,C,D) and then is completed with a "filler" chemical. The food scientists beelive one of the A,B,C, or D preservatives is causing the problem, but need your help to figure out which one! Use Machine Learning with RF to find out which parameter had the most predicitive power, thus finding out which chemical causes the early spoiling! So create a model and then find out how you can decide which chemical is the problem!

- Pres_A : Percentage of preservative A in the mix
- Pres_B : Percentage of preservative B in the mix
- Pres_C : Percentage of preservative C in the mix
- Pres_D : Percentage of preservative D in the mix
- Spoiled: Label indicating whether or not the dog food batch was spoiled.

Think carefully about what this problem is really asking you to solve. While we will use Machine Learning to solve this, it won't be with your typical train/test split workflow.

In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
#Tree methods Example
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('dogfood').getOrCreate()
```

In [3]:
```
# Load training data
data = spark.read.csv('dog_food.csv',inferSchema=True,header=True)
```

In [4]:
```
data.count()
```

Out[4]:
```
490
```

In [5]:
```
data.printSchema()
```

```
root
 |-- A: integer (nullable = true)
 |-- B: integer (nullable = true)
 |-- C: double (nullable = true)
 |-- D: integer (nullable = true)
 |-- Spoiled: double (nullable = true)
```

In [6]:
```
data.head()
```

Out[6]:
```
Row(A=4, B=2, C=12.0, D=3, Spoiled=1.0)
```

In [7]:
```
data.describe().show()
```

```
+-------+------------------+------------------+------------------+------------------+------------------+
|summary|                 A|                 B|                 C|                 D|           Spoiled|
+-------+------------------+------------------+------------------+------------------+------------------+
|  count|               490|               490|               490|               490|               490|
|   mean|  5.53469387755102| 5.504081632653061| 9.126530612244897| 5.579591836734694| 0.2857142857142857|
| stddev|2.9515204234399057|2.8537966089662063|2.0555451971054275|2.8548369309982857|0.45221563164613465|
|    min|                 1|                 1|               5.0|                 1|               0.0|
|    max|                10|                10|              14.0|                10|               1.0|
+-------+------------------+------------------+------------------+------------------+------------------+
```

In [8]:
```
# Import VectorAssembler and Vectors
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

In [9]:
```
data.columns
```

Out[9]:
```
['A', 'B', 'C', 'D', 'Spoiled']
```

In [10]:
```
assembler = VectorAssembler(inputCols=['A', 'B', 'C', 'D'],
                            outputCol="features")
```

In [11]:
```
output = assembler.transform(data)
```

In [12]:
```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import DecisionTreeClassifier
```

In [13]:
```
rfc = DecisionTreeClassifier(labelCol='Spoiled',featuresCol='features')
```

In [14]:
```
output.printSchema()
```

```
root
 |-- A: integer (nullable = true)
 |-- B: integer (nullable = true)
 |-- C: double (nullable = true)
 |-- D: integer (nullable = true)
 |-- Spoiled: double (nullable = true)
 |-- features: vector (nullable = true)
```

In [15]:
```
final_data = output.select('features','Spoiled')
final_data.head()
```

Out[15]:
```
Row(features=DenseVector([4.0, 2.0, 12.0, 3.0]), Spoiled=1.0)
```

In [16]:
```
rfc_model = rfc.fit(final_data)
```

In [17]:
```
rfc_model.featureImportances
```

Out[17]:
```
SparseVector(4, {1: 0.0019, 2: 0.9832, 3: 0.0149})
```

Feature at index 2 (Chemical C) is by far the most important feature, meaning it is causing the early spoilage! This is a pretty interesting use of a machine learning model in an alternative way!