Handling Missing Data and Imputing Values

Lesson Objectives

- After completing this lesson, you should be able to:
  - Drop records according to different criteria
  - Fill missing data according to different criteria
  - Drop duplicate records

DataFrame NA Functions

- The na method of DataFrames provides functionality for working with missing data
- Returns an instance of DataFrameNAFunctions
- The following methods are available:
  - drop, for dropping rows containing NaN or null values
  - fill, for replacing NaN or null values
  - replace, for replacing values matching specified keys

DataFrame with Missing Values

```
from numpy import NaN
```

```
from pyspark.sql.functions import UserDefinedFunction
from pyspark.sql.types import DoubleType

udf1 = UserDefinedFunction(lambda x: NaN if x > 0.5 else x, DoubleType())
udf2 = UserDefinedFunction(lambda x: NaN if x > 1.0 else x, DoubleType())
```

```
dfnan = df2.withColumn("nanUniform", udf1("uniform")) \
            .withColumn("nanNormal", udf2("normal")).drop("uniform") \
            .withColumnRenamed("nanUniform", "uniform").drop("normal") \
            .withColumnRenamed("nanNormal", "normal")
```

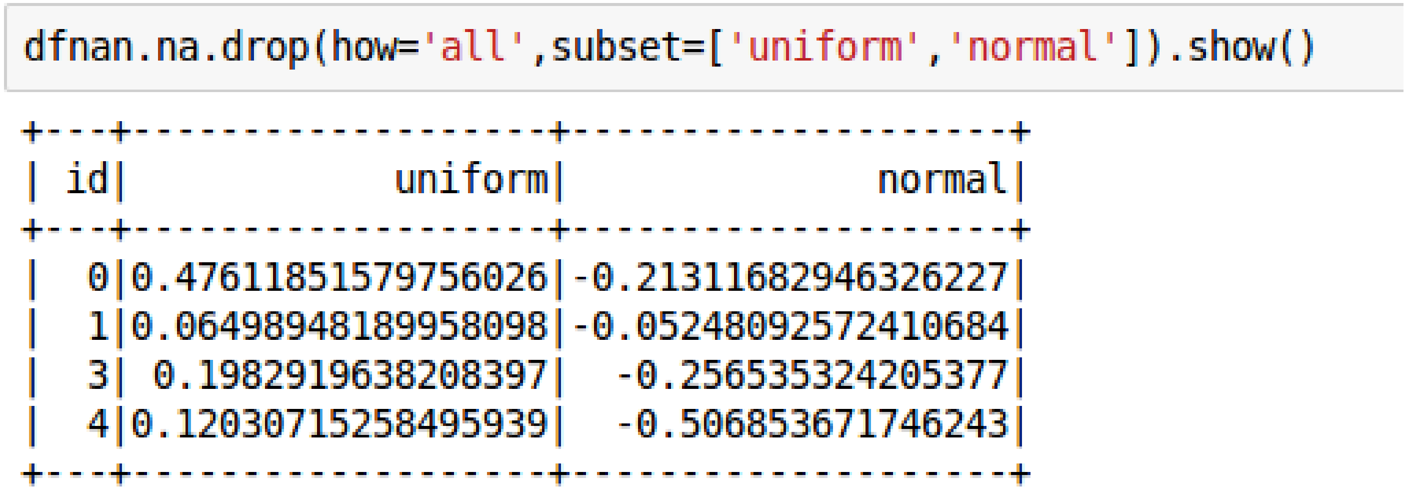
```
dfnan.show()
```

```
+---+-------------------+--------------------+
| id|            uniform|              normal|
+---+-------------------+--------------------+
|  0|0.47611851579756026|-0.21311682946326227|
|  1|0.06498948189958098|-0.05248092572410684|
|  2|                NaN|                 NaN|
|  3| 0.1982919638208397|  -0.256535324205377|
|  4|0.12030715258495939|  -0.506853671746243|
+---+-------------------+--------------------+
```

DataFrame NA Functions - drop

- drop is used for dropping rows containing NaN or null values according to a criteria
- Several implementations available:
  - drop(thresh, subset)
  - drop(thresh)
  - drop(how, subset)
  - drop(subset)
  - drop(how)
  - drop()
- cols is a List of column names
- how should be equal any or all

Dropping Rows With How Argument



```
dfnan.na.drop(how='all',subset=['uniform','normal']).show()
```

```
+---+-------------------+--------------------+
| id|            uniform|              normal|
+---+-------------------+--------------------+
|  0|0.47611851579756026|-0.21311682946326227|
|  1|0.06498948189958098|-0.05248092572410684|
|  3| 0.1982919638208397|  -0.256535324205377|
|  4|0.12030715258495939|   -0.50685367174643|
+---+-------------------+--------------------+
```

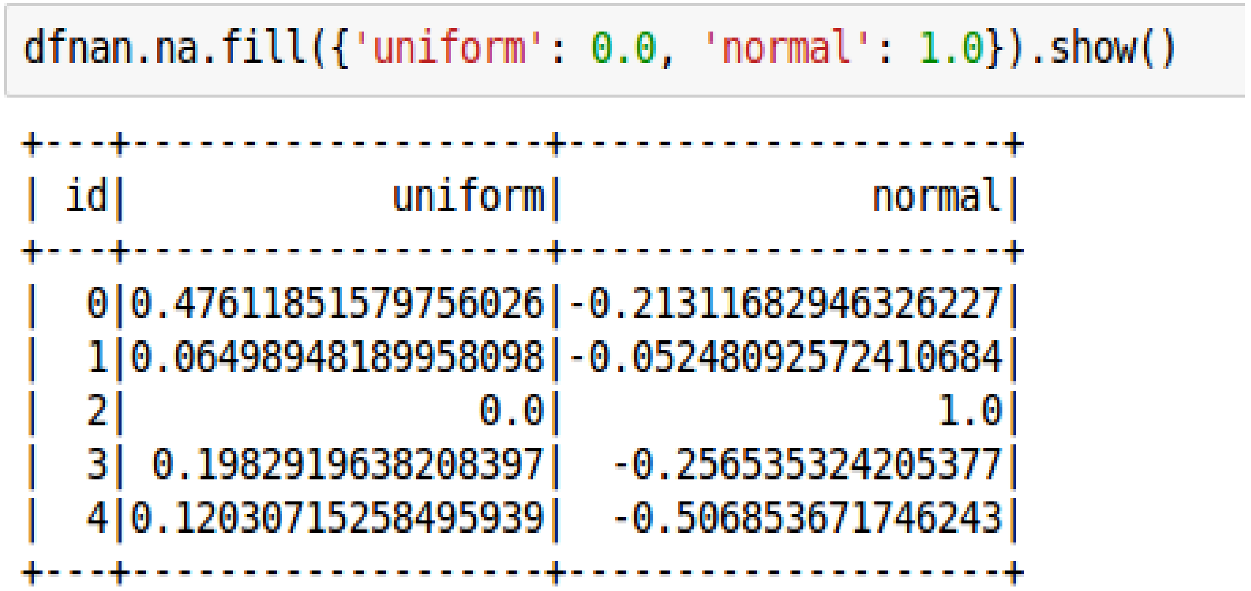
DataFrame NA Functions - fill

- fill is used for replacing NaN or null values according to a criteria
- Several implementations available:
  - fill(dict)
  - fill(value, cols)
  - fill(value)

DataFrame NA Functions - fill

- Returns a new DataFrame that replaces null or NaN values in numeric columns with value.
- dict is a mapping of column names to default values
- value is either a String or Double
- subset is a List of column names

Filling Missing Data With Column Defaults



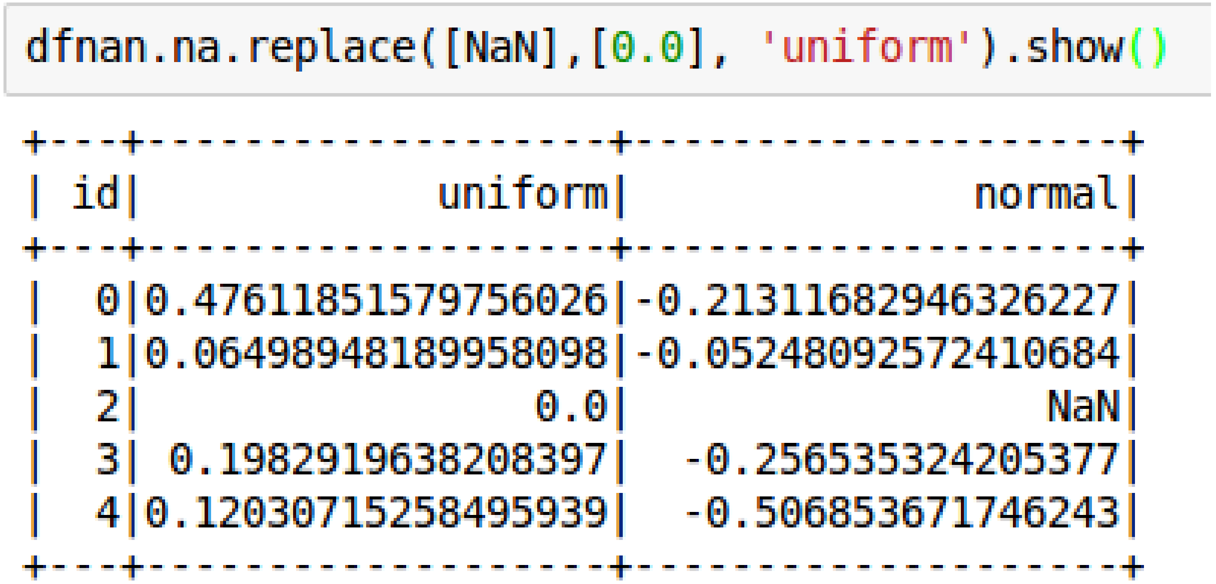
```
dfnan.na.fill({'uniform': 0.0, 'normal': 1.0}).show()
```

```
+---+-------------------+--------------------+
| id|            uniform|              normal|
+---+-------------------+--------------------+
|  0|0.47611851579756026|-0.21311682946326227|
|  1|0.06498948189958098|-0.05248092572410684|
|  2|                0.0|                 1.0|
|  3| 0.1982919638208397|  -0.256535324205377|
|  4|0.12030715258495939|  -0.506853671746243|
+---+-------------------+--------------------+
```

DataFrame NA Functions - replace

- replace is used for replacing values matching specified keys
- cols argument may be a single column name or an array
- replacement argument is a map:
  - key is the value to be matched
  - value is the replacement value itself

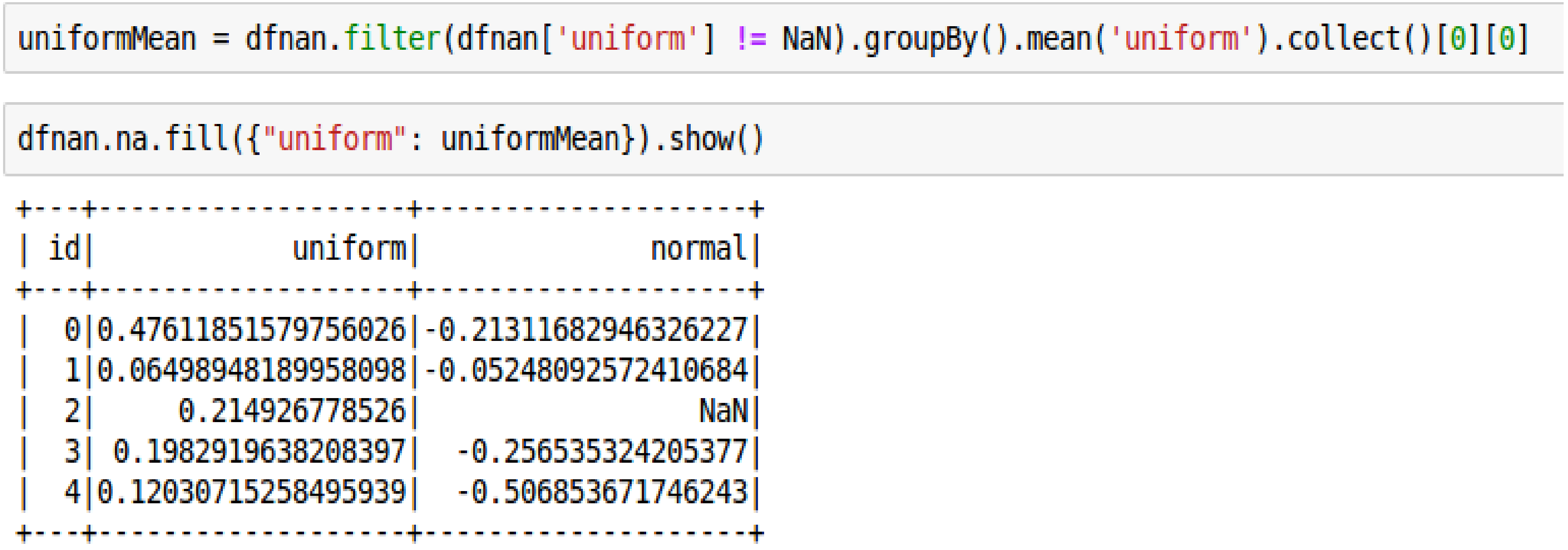
Replacing Values in a DataFrame



```
dfnan.na.replace([NaN],[0.0], 'uniform').show()
```

```
+---+--------------------+--------------------+
| id|             uniform|              normal|
+---+--------------------+--------------------+
|  0|0.4761185157975602 6|-0.2131168294632622 7|
```

Replacing Values in a DataFrame



```
uniformMean = dfnan.filter(dfnan['uniform'] != NaN).groupBy().mean('uniform').collect()[0][0]
```

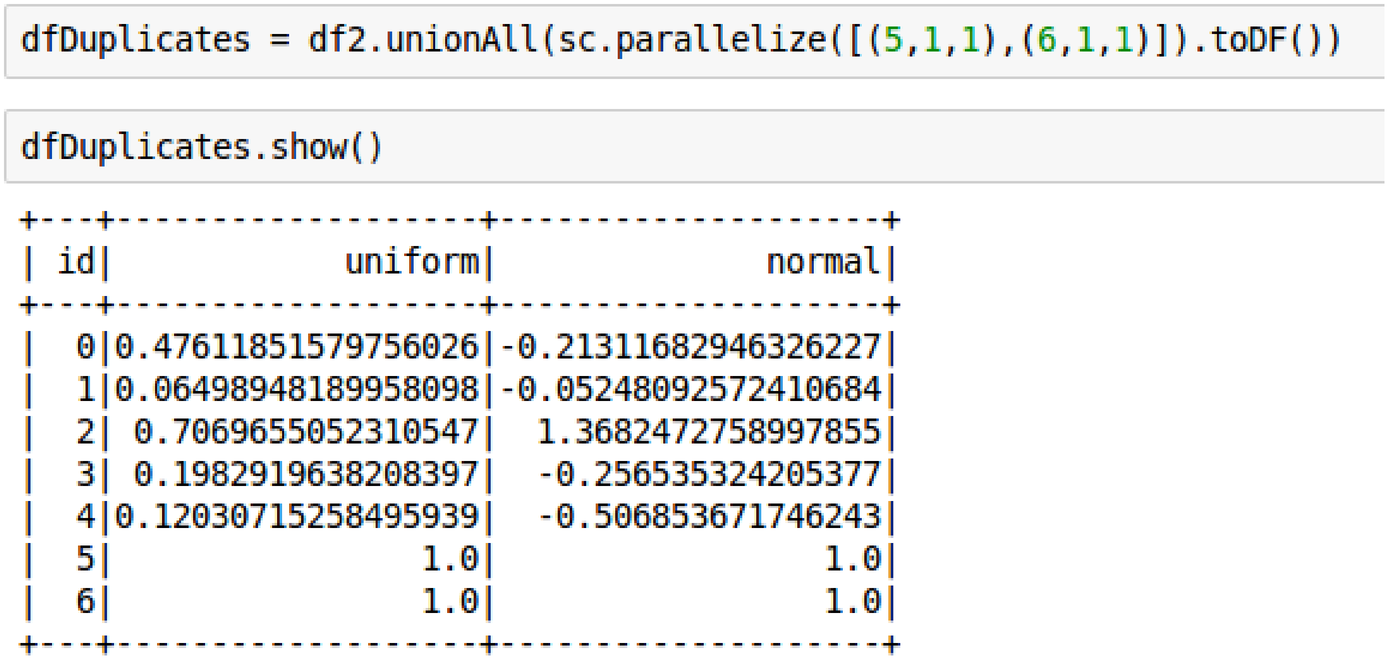
```
dfnan.na.fill({"uniform": uniformMean}).show()
```

```
+---+-------------------+--------------------+
| id|            uniform|              normal|
+---+-------------------+--------------------+
|  0|0.47611851579756026|-0.21311682946326227|
|  1|0.06498948189958098|-0.05248092572410684|
|  2|     0.214926778526|                 NaN|
|  3| 0.1982919638208397|  -0.256535324205377|
|  4|0.12030715258495939|  -0.506853671746243|
+---+-------------------+--------------------+
```

Duplicates

- dropDuplicates is a DataFrame method
- Used to remove duplicate rows
- May specify a subset of columns to check for duplicates

Dropping Duplicate Rows

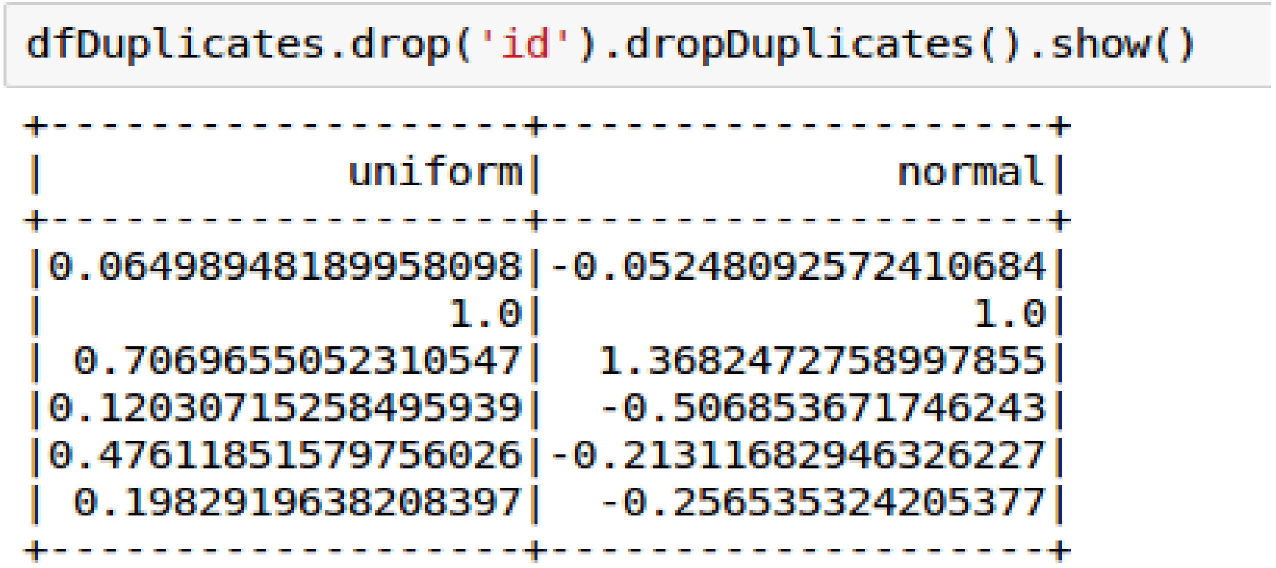


```
dfDuplicates = df2.unionAll(sc.parallelize([(5,1,1),(6,1,1)]).toDF())
```

```
dfDuplicates.show()
```

```
+---+-------------------+--------------------+
| id|            uniform|              normal|
+---+-------------------+--------------------+
|  0|0.47611851579756026|-0.21311682946326227|
|  1|0.06498948189958098|-0.05248092572410684|
|  2| 0.7069655052310547|  1.3682472758997855|
|  3| 0.1982919638208397| -0.2565353242053770|
|  4|0.12030715258495939| -0.5068536717462430|
|  5|                1.0|                 1.0|
|  6|                1.0|                 1.0|
+---+-------------------+--------------------+
```

Dropping Duplicate Rows



```
dfDuplicates.drop('id').dropDuplicates().show()
```

```
+-------------------+--------------------+
|            uniform|              normal|
+-------------------+--------------------+
|0.06498948189958098|-0.05248092572410684|
|                1.0|                 1.0|
| 0.7069655052310547|  1.3682472758997855|
|0.12030715258495939|  -0.506853671746243|
|0.47611851579756026|-0.21311682946326227|
| 0.1982919638208397|  -0.256535324205377|
+-------------------+--------------------+
```

Lesson Summary

- Having completed this lesson, you should be able to:
  - Drop records according to different criteria
  - Fill missing data according to different criteria
  - Drop duplicate records